Ivana Ognjanoviq
Ramo Shendel

ALGORITMET DHE PROGRAMIMI

Teksti mësimor
për vitin e tretë ose të katërt të gjimnazit

Enti i Teksteve dhe i Mjeteve Mësimore
PODGORICË

IVANA OGNJANOVIQ RAMO SHENDEL

ALGORITMET DHE PROGRAMIMI

Teksti mësimor për vitin e tretë ose të katërt të gjimnazit

Enti i Teksteve dhe i Mjeteve Mësimore
PODGORICË
2015

doc. dr. Ivana Ognjanoviq • prof.dr. Ramo Shendel

ALGORITMET DHE PROGRAMIMI

teksti mësimor për vitin e tretë ose të katërt të gjimnazit

ALGORITMI I PROGRAMIRANJE

udžbenik za treći ili četvrti razred gimnazije

Botues:	Enti i Teksteve dhe i Mjeteve Mësimore – Podgoricë
Për botuesin:	Zoja Bojaniq-Laloviq
Kryeredaktor:	Radulle Novoviq
Redaktor përgjegjës dhe redaktor i botimit:	Llazo Lekoviq
Redaktor i botimit në shqip:	Dimitrov Popoviq
Recensentët:	dr. Predrag Stanishiq mr. Goran Shukoviq Ana Miranoviq Goran Zhivkoviq Natasha Gazivoda
Përkthyes:	Prof. dr. David Kalaj
Faqosja e tekstit në shqip:	Nikolla Knezheviq
Përgatitja teknike dhe dizajni:	STUDIO MOUSE – Podgoricë
Redaktor teknik:	Rajko Radulloviq

CIP – Каталогизација у публикацији
Национална библиотека Црне Горе, Цетиње

ISBN 978-86-303-1896-2
COBISS.CG-ID 27984144

Këshilli kombëtar i arsimit, me vendimin me nr. 16-4487 të datës 10. 09. 2013, e ka miratuar këtë tekst mësimor për përdorim në gjimnaze.

Përmbajtja

Legjenda e simboleve të përdorura në tekst

	interesante
	koncepti që përkufizohet
	shpjegim shtesë
	pyetje e vështirë
	pyetje të lehta për të cilat nuk ka zgjidhje në Përmbledhje
	pyetje/detyrë për të cilën ekziston zgjidhja në Përmbledhje
	më hollësisht në CD-në përcjellëse
	përmbajtja e skedarëve

PARATHËNIA

Zhvillimi dhe përdorimi i teknologjisë së komunikimit informativ (ICT) e ka transformuar shoqërinë moderne në "shoqëri informative" gjë që paraqet mbështetje në zhvillimin e tërësishëm shoqëror dhe ekonomik. Meritat për aplikacionin e madh dhe të gjerë të ICT i përkasin kryesisht lëmit të zhvillimit të softuerit. Në 60 vitet e fundit programuesit kanë arritur të përgjigjen në mënyrë konstante në sfidat e tregut aktual duke krijuar softuerë me cilësi të lartë me anë të të cilëve është përmirësuar dukshëm efikasiteti dhe efektiviteti i punës njerëzore. Si rrjedhim, edhe sot lëmi i zhvillimit të softuerit është shumë i rëndësishëm, dhe paraqet shtysë për zhvillimin e shumë lëmenjve të veprimtarisë njerëzore, në përputhje me parashikimin se programuesit në periudhën e ardhshme do të jenë ekspertë shumë të vlerësuar dhe të kërkuar.

Ky tekst është shkruar në përputhje me planin mësimor për nxënësit e klasës III dhe IV të shkollës së mesme. Qëllimi kryesor i tekstit është njohja e nxënësve me krijimin e algoritmeve (d.m.th. paraqitjen e zgjidhjes së ndonjë problemi, duke përkufizuar bashkësinë e aksioneve dhe renditjen e kryerjes së tyre) dhe zhvillim e zgjidhjeve përkatëse me softuer në gjuhën e programimit të orientuar në objekte Java.

Teksti është i ndarë në tri pjesë. Në pjesën e parë janë të përshkruara algoritmet, mënyra e krijimit të tyre dhe procedura e verifikimit të saktësisë. Pjesa e dytë e tekstit është pjesa qendrore në të cilën janë të përshkruara konceptet themelore të programimit të orientuar në objekte dhe elementet themelore të gjuhës programuese Java. Rëndësi të madhe për programim të suksesshëm në gjuhën e programimit Java ka njohja e klasave themelore të Javës dhe klasave për paraqitjen e rrjedhave të të dhënave dhe skedarëve, si dhe puna me komandat drejtuese, vargje dhe koleksione të të dhënave, duke përdorur teknikat e veçanta programuese, ndër të cilat veçohet rekursioni. Si pasojë, temave të përmendura u është përkushtuar një vëmendje e veçantë në këtë pjesë të tekstit. Pjesa e tretë u është dedikuar nxënësve të talentuar që dëshirojnë të zgjerojnë dituritë e tyre në lëmin e programimit, prandaj kapitujt e caktuar të tekstit dorëzohen si shtesë në CD-në përcjellëse. Fokusi i kësaj pjese të tekstit është zhvillimi i aplikacioneve grafike përdoruese duke zbatuar teknikat e avancuara të programimit. Është paraqitur mënyra e krijimit të formave themelore grafike të interfejsit përdorues dhe përcjellja e ngjarjeve rreth tyre. Zbatimi i teknikave të avancuara që përfshin sortimin (renditjen) dhe kërkimin e vargjeve, backtracking, programimi dinamik dhe puna me grafe, e zmadhon dukshëm kualitetin e programeve të zhvilluara.

Kontribut të veçantë në cilësinë e tekstit i jep padyshim një numër i madh i shembujve të zgjedhur me kujdes së bashku me zgjidhjen e tyre, si dhe detyrat e projektit që janë të përshtatshme për punë nëpër grupe në klasa, me çfarë inkurajohet bashkërendimi, komunikimi më kualitativ dhe bashkëpunimi midis nxënësve. Në fund të çdo kapitulli janë të shënuara pyetjet për kontrollin e njohurive dhe detyrat e pazgjidhura për punë individuale.

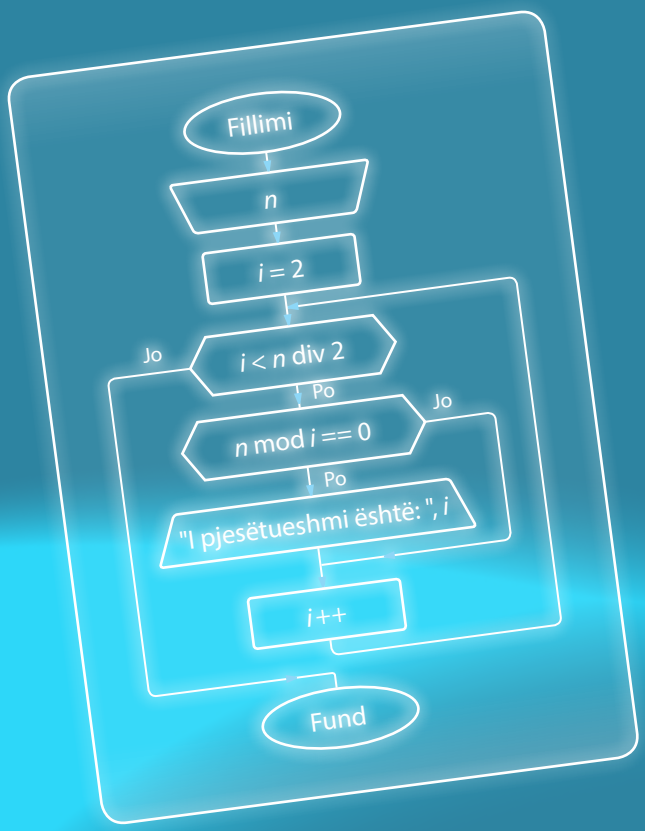
Në përgatitjen e tekstit është nisur nga pikësynimi që nxënësve duhet t'u përcillen në mënyrë pasive konceptet themelore të programimit të orientuar në objekte, mirëpo çfarë është edhe më e rëndësishme, edhe nga pikësynimi që është e domosdoshme që te ta në mënyrë aktive të zhvillohet ndjenja, siguria dhe besimi në aftësitë personale për programim, duke i motivuar ata që të pranojnë rekomandimet e përvetësuara të programimit dhe njëkohësisht të zhvillojnë stilin personal duke i inspiruar që në mënyrë permanente të konsultojnë literaturën dhe burimet e tjera të diturisë.

Tekstin mund ta përdorë edhe studenti i fakulteteve teknike në lëndët në lëmin e programimit.

Autorët

I.

TEORIA E ALGORITMEVE



Algoritmet përdoren çdo ditë në jetën e përditshme, dhe zakonisht nuk jemi as të vetëdishëm për këtë. Në informatikë dhe matematikë algoritmet zënë vend me rëndësi të posaçme, sepse nevojitet të përkufizohet një varg veprimesh me qëllim që të vihet deri te zgjidhja e kërkuar e ndonjë probleme.

Njohja e mirë e algoritmeve është e domosdoshme për secilin programues, duke marrë në konsiderim faktin që programi paraqet algoritmin e shkruar në një gjuhë programuese.

Në këtë kapitull jepen përgjigjet në pyetjet që vijojnë:

- Kush e ka shkruar algoritmin e parë?
- Si mund të paraqitet algoritmi?
- Cilat pjesë e përbëjnë algoritmin?
- Cilat lloje të algoritmeve ekzistojnë?

Gjithashtu, do të shkruani algoritmet tuaja të para, duke filluar nga të thjeshtat deri te të përbërat, që përbëjnë ciklet dhe degëzimet e kushtëzuara, në bazë të të cilave më vonë do të shkruani programe në gjuhën programuese Java.

Njohja me algoritme zakonisht fillon me tregimin mbi matematikanin, astronomin dhe gjeografin persian të shekullit IX **EL Horezmi** (emri i tij i plotë në transkriptim anglez është **Muhammad ibn Musa al-Khwarizmi**). Në vitin 852 ka shkruar librin në të cilin ka përshkruar veprimet për llogaritjen në sistemin numëror indian. Origjinali në gjuhën arabe nuk është ruajtur, kurse emri në përkthimin latin është "Algoritmi de numero Indorum" ("Al-Khwarizmi mbi numrat indianë"). Si rrjedhim, rregullat për të zgjidhur, në atë kohë kryesisht problemet matematike, në bazë të emrit "al Khawarizmi" janë quajtur ***algoritme***.

Çfarë është në të vërtetë algoritmi?

> Me nocionin ***algoritëm*** sot nënkuptojmë vargje të veprimeve të sakta të cilat hap nga një hap na çojnë në zgjidhjen e problemit.

Ato janë udhëzime aq të sakta që për kryerjen e tyre nuk nevojitet mësim apo mendim shtesë. Procesi i përpilimit të algoritmit në programim paraqet fazën më të rëndësishme, e cila nuk varet nga gjuha e programimit që programuesi e njeh dhe në të cilën do të shkruhet programi më vonë.

Algoritmin e parë për llogaritje e ka shkruar Ada Bajron në vitin 1842. Bëhet fjalë mbi algoritmin për llogaritjen e numrave të Bernulit në makinën analitike të Çarlls Bebixhit. Ajo makinë nuk ka funksionuar kurrë, mirëpo algoritmi i saj ka lënë gjurmë të thella. Sot programi i Adës konsiderohet si programi i parë kompjuterik, kurse për nder të Ada Bajronit, një nga gjuhët e programimit ka marrë emrin Ada.

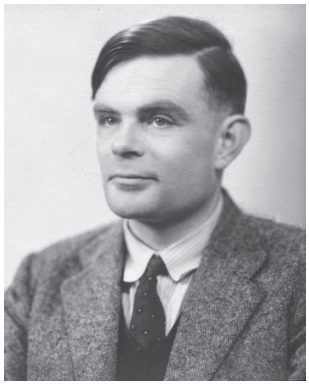
Avancim të rëndësishëm në formalizimin e futjes në përdorim të algoritmeve në matematikë dhe në logjikë më pas ka dhënë Alan Tjuringu, duke përkufizuar makinën e Tjuringut. Bëhet fjalë mbi një makinë imagjinare që mund të paraqesë të dhënat dhe në to të kryejë algoritmin. Edhe pse ka një strukturë të thjeshtë, kjo makinë është ekuivalente me të gjithë kompjuterët elektronikë dhe mekanikë.

Shumë autorë të teksteve, algoritmet i sqarojnë duke marrë shembuj recetash të disa gjellëve, si dhe në procedurat e përkufizuara në mënyrë koncize për përpunimin e disa produkteve. Ne do të shqyrtojmë kryesisht shembuj nga jeta e përditshme, të cilët kanë të bëjnë me kohën moderne informatike në të cilën jetojmë. Përpiquni të rikujtoheni, kur kini lexuar herën e fundit një udhëzim të hollësishëm. Ndoshta e keni pasur në dorë dhe nuk e keni lexuar, sepse ai udhëzim është për ju një rutinë, si p.sh. plotësimi i llogarisë së celularit nëpërmjet kuponit.

Ose, mund të veprojmë anasjelltas. A ju ka ndodhur në kohën e fundit situata e tillë që për të kryer një punë nuk keni pasur udhëzime të qarta ose nuk keni pasur aspak udhëzime?

Më herët keni mësuar procedurat (algoritmet) për mbledhjen dhe shumëzimin e numrave, algoritmin e Euklidit për përcaktimin e pjesëtuesit më të madh të përbashkët të dy numrave, algoritmin e Gausit për zgjidhjen e sistemit të ekuacioneve lineare dhe shumë shembuj të tjerë. Nga fizika keni mësuar si të njehsoni fuqinë e rrymës dhe rezistencën e përgjithshme në qarqet e rrymës me një drejtim dhe të rrymës alternative; nga kimia të sqaroni reaksionet kimike... Algoritmet janë prezentë në secilën fushë të shkencës, dhe në jetën e përditshme jemi vazhdimisht në kontakt me to, dhe shpesh veprojmë sipas algoritmeve, por pa vetëdije. Është mirë që të mendosh në mënyrë "algoritmike", pavarësisht se a merreni me programim apo jo.

Muhamed bin Musa al Horezmi (*Muhammed ibn Musa al Khowarizmi*) në një pullë postare e cila shënon përafërsisht 1200 vite nga lindja e tij.



Një nga teoricienët më të rëndësishëm të informatikës moderne angleze është **Alan Tjuring** (1912–1954). Gjatë luftës së dytë botërore është marrë me "Enigmën" gjermane, aparatin të cilin ushtria gjermane e përdori për kodimin dhe dekodimin e mesazheve. Deri në fund të Luftës së Dytë Botërore Tjuringu ka zhvilluar një proces me të cilin të gjitha mesazhet kanë mundur të dekodohen. Gjatë atyre hulumtimeve është shfaqur makina e Tjuringut.

1.1. Algoritmet

Sot, numrin më të madh të detyrave njeriu e zgjidh nëpërmjet kompjuterit. Të gjithë punët që bën kompjuteri kryhen në mënyrë progresive, hap pas hapi, në kohë të fundme. Çdo hap është precizuar saktësisht, si dhe kalimi në hapin e ardhshëm.

Çfarë është algoritmi?

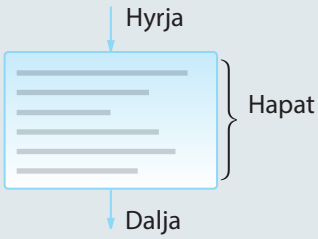


Figura 1.1.
Figura e thjeshtëzuar e algoritmit

Algoritmi paraqet një bashkësi veprimesh që kryhen me renditje të përcaktuar paraprakisht, me qëllim që nga të dhënat hyrëse të njehsohen rezultatet e kërkuara.

Në procesin e programimit, bashkësia e veprimeve është përkufizuar sipas mundësive të kompjuterit, gjegjësisht komandave të gjuhës programuese që përdoret, kurse renditja e kryerjes së veprimeve jepet nëpërmjet strukturave algoritmike (programuese).

Gjatë shënimit të programit, fillimisht duhet të jetë e qartë se çfarë kërkohet nga programi. Sikurse edhe te zgjidhja e detyrave, në çdo fushë tjetër, problemi duhet të përcaktohet dhe përkufizohet në mënyrë të qartë. Duke kuptuar problemin që zgjidhet, fillimisht bëhet skica e përafërt, në bazë të së cilës përpunohet algoritmi në detaje. Mbas përpilimit të algoritmit, vijon shënimi i programit që paraqet paraqitjen e algoritmit nëpërmjet të elementeve të një gjuhe programuese të caktuar.

Programi është algoritëm i shkruar në një gjuhë programimi të caktuar.

Krijimi i algoritmit fillon me zbërthimin e problemeve komplekse në probleme më të vogla dhe përkufizimin e sistemit të qartë të rregullave me të cilat madhësitë e njohura (hyrëse) të detyrës transformohen deri te rezultatet e kërkuara.

Që ndonjë problem të zgjidhet në atë mënyrë, procesin e zgjidhjes duhet ta definojmë sipas disa fazave: Ato janë:

1. Kuptimi i problemit
2. Krijimi i modelit
3. Krijimi i algoritmit
4. Përpunimi i një shembulli test dhe verifikimi i vërtetësisë së algoritmit
5. Implementimi i algoritmit (d.m.th. shënimi i programit)
6. Testimi i programit
7. Përpilimi i dokumenteve

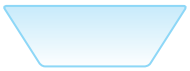
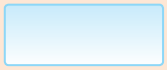
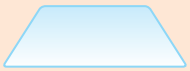
Kuptimi i ngushtë i termit programim përfshin vetëm hapat 5 dhe 6.

Teoria e algoritmeve është një fushë e pavarur që përkufizon modele abstrakte për zgjedhjen e problemave pavarësisht nga gjuha e programimit. Në mënyrë të ngjashme si te disiplinat e tjera matematikore, hulumtohen ligjshmëritë dhe principet e algoritmeve, dhe jo implementimet konkrete.

Ekzistojnë mundësi të ndryshme të paraqitjes së algoritmeve, për të cilat do të flitet më shumë në vazhdim.

1.2. Paraqitja e algoritmeve

Për paraqitjen e algoritmeve më shpesh përdoret paraqitja grafike në formë të një skeme e cila quhet **skema e algoritmeve** ose **diagrami i rrjedhës** (eng. *flowchart*). Në skemën algoritmike, secili veprim paraqitet me simbol të veçantë grafik. Simbolet grafike që përdoren për të bërë skemat algoritmike janë:

SIMBOLI GRAFIK	DOMETHËNIA
	Terminatori (përcakton fillimin (*start*) ose fundin (*end*) të algoritmit.
	Futja e të dhënave (përcakton madhësitë hyrëse të algoritmit)
	Përcakton përpunimin e të dhënave
	Hapi i kushtëzuar algoritmik
	Nxjerrja e të dhënave (përcakton madhësitë dalëse të algoritmit)
	Vazhdimi i algoritmit
	Fund i ciklit
	Lidhja e hapave algoritmikë
	Linja e rrjedhës (Linja lidhëse)

Rikujtoni se të dhënat dhe udhëzimet (programet) gjenden në memorien punuese të kompjuterit (RAM). Procesori kryen udhëzimet e programit, një nga një, në bazë të cilave merr të dhënat nga memoria, i përpunon dhe rezultatet e përpunimit i kthen mbrapa në memorie në vende të parapara, prej nga i dërgon në njërën prej njësive dalëse – monitor, printer, mikrofon...

Për programimin është veçanërisht e rëndësishme të kuptohet veprimi me ***ndryshore (variabla)*. Ndryshoret mundësojnë ruajtjen e të dhënave në memorien operative.** Pjesët e memories në të cilat ruhen të dhënat kanë adresën e vet dhe në procesin e krijimit të algoritmit dhe programimit iu jepen emrat. Gjatë përpunimit të algoritmeve, ndryshoreve u jepen emrat (simbolet) sipas dëshirës, duke pasur parasysh që të mund të shënohen në kuadër të skemës algoritmike. Në gjuhët e programimit, emrat zakonisht formohen nga shkronjat e alfabetit latin dhe disa simbole të tjera (më vonë në pjesën mbi gjuhën e programimit Java jepen edhe rregullat për emërtimin e ndryshoreve në atë gjuhë programimi).

Me ndihmën e ***operatorëve*** mund të kryhen operacione të ndryshme në të dhënat. Në vazhdim janë të dhënë operatorët që përdoren më shpesh, me shembuj që i përshkruajnë ata hollësisht.

Mënyra më e shpeshtë për paraqitjen e algoritmeve është ajo grafike – nëpërmjet skemave algoritmike, që i korrespondon faktit se 80 % të informatave njeriu i merr në mënyrë vizuale.

Përveç paraqitjes grafike, shpeshherë përdoren edhe këto dy mënyra:
– **gjuha natyrore** (Që algoritmi i cili kumtohet me gjuhë natyrore, të jetë preciz dhe me mjaft i detajuar, duhet të kihet parasysh që ekspozimi të jetë i qartë e jo konfuz, që ka shumë rëndësi gjatë përkufizimit të renditjes së veprimeve që duhet të kryhen).
– **gjuha pseudo** (Gjuha pseudo është kombinimi joformal i gjuhës natyrore dhe një gjuhe të imagjinuar programuese, prandaj zbatimi i saj nënkupton shënimin e algoritmit në një formë që i përngjan një gjuhe programuese).

Çfarë është ndryshorja?

Ndryshorja në matematikë nuk paraqet asnjë vlerë të veçantë. Megjithatë, në gjuhët e programimit termi "ndryshore" përdoret për diçka që ka një kohëzgjatje të caktuar dhe vlera e të cilës në momentin e caktuar është konstante dhe mbi të cilën mund të kryhen operacione të caktuara.
Shembuj ndryshoresh:

```
sipërfaqja = 2.75
x = 2
dita = "e hënë"
```

1.2.1. Operatorët aritmetikë

Operatorët +, -, *, /, *div* i *mod* quhen operatorët aritmetikë. Tabela 1.1. përmban shembuj zbatimesh të këtyre operatorëve. Operatori / realizon pjesëtimin në bashkësinë e numrave realë, përderisa operatorët *div* dhe *mod* janë të caktuar për punë me numra të plotë: rezultati i operatorit *div* është herësi i plotë, ndërsa rezultati i *mod* është mbetja gjatë pjesëtimit të numrave të plotë.

Operatorët aritmetikë!

Tabela 1.1. Operatorët aritmetikë

Operatorët	Veprimi	Shembulli	Përshkrimi
+	Mbledhja	5 + 2 = 7	Mbledhja e dy numrave
–	Zbritja	5 – 2 = 3	Zbritja e dy numrave
*	Shumëzimi	5 * 2 = 15	Shumëzimi i dy numrave
/	Pjesëtimi	11,9394 / 2,2 = 5,427 11 / 2 = 5,5	Pjesëtimi i dy numrave realë jep rezultatin numër real
		10 / 2 = 5,0	Pjesëtimi i dy numrave të plotë jep rezultatin numër real, nëse përdoret operatori i pjesëtimit për numra realë.
div	Pjesëtimi i plotë (div)	10 div 2 = 5 11 div 2 = 5	Në qoftë se përdoret operatori i pjesëtimit të numrave të plotë, rezultati do të jetë numër i plotë madje edhe atëherë kur i plotpjesëtueshmi nuk plotpjesëtohet me pjesëtuesin.
mod	Mbetja e pjesëtimit të plotë (moduli)	10 mod 3 = 1	10 pjesëtuar me 3 është 3 dhe mbetja është 1
		8 mod 3 = 2	8 pjesëtuar me 3 është 2 dhe mbetja është 2
		8 mod 2 = 0	8 pjesëtuar me 2 është 4 dhe mbetja është 0

Kini kujdes në shembujt e mëposhtëm për operatorët **div** dhe **mod**:

(–7) div 2 = –3

(–7) div (–2) = 3

(–14) mod 3 = 1

(–10) mod 5 = 0

Operatori **mod** (*a* mod *b*) është i përkufizuar vetëm për ***b* > 0**.

Në praktikë këto veprime realizohen më shpesh vetëm mbi madhësitë jonegative.

Operatori *mod* zbatohet për shumë njehsime dhe vërtetime. Për shembull, me ndihmën e tij mund të vërtetohet se a është një numër i plotpjesëtueshëm me numrin tjetër (në qoftë se po, mbetja e numrit të plotë duhet të jetë 0), a është një numër, numër çift apo numër tek (numri çift gjatë pjesëtimit me 2 jep mbetjen 0, kurse numri tek jep mbetjen 1), etj.

Rikujtoni ***kongruencën sipas modulit*** nga matematika, d.m.th. shënimit 3 ≡ 1(mod 2)

Tabela 1.2. Shembuj zbatimesh gjatë pjesëtimit me numër të plotë (*mod*)

SHEMBULLI	DOMETHËNIA
a mod *b* = 0	numri *a* është i plotpjesëtueshëm me numrin *b*
a mod 7 = 0	numri *a* është i plotpjesëtueshëm me 7
a mod 2 = 0	numri *a* është çift
a mod 2 = 1	numri *a* është tek
197 mod 10 = 7	7 është shifra e njësheve e numrit 197

1.2.2. Operatorët e shoqërimit

Operatori tjetër me rëndësi është operatori = . Domethënia e këtij operatori në programim dallohet nga domethënia në matematikë dhe në programim zakonisht e quajmë **operator të shoqërimit**. Në gjuhët e programimit, rezultati i veprimit, që kryhet në anën e djathtë të barazimit i shoqërohet ndryshores në anën e majtë të barazimit. Për shembull, ndryshores B i shoqërohet vlera e ndryshores A të zmadhuar për 3 sipas shprehjes: $B = A + 3$.

Megjithatë, edhe pse është korrekt matematikisht, nuk lejohet të shkruhet: $3 + A = B$, sepse një shprehje e tillë do të kuptohej si shoqërim i vlerës së ndryshores B shprehjes $A + 3$, dhe ky veprim nuk mund të kryhet.

Në anën tjetër, është e lejueshme: $A = A + B$, ku ndryshores A i shoqërohet shuma e vlerave të ndryshoreve A dhe B.

Operatorët e shoqërimit!

Shembull 1.
Të plotësohet tabela me vlerat e ndryshoreve mbas veprimeve të kryera.

Ndryshoret	Shprehja	
$x = 2,\ y = 3$	$z = x + y$	$z =$
$x = 2,\ y = 3$	$x = x + y$	$x =$
$x = 5$	$x = x \bmod 3$	$x =$
$z = 0,\ y = 3$	$z = y + 1$	$z =$

1.2.3. Operatorët e rritjes dhe të zvogëlimit

Rritja e vlerës së madhësisë a për 1 mund të kryhet nëpërmjet shprehjes $a = a + 1$, ose duke përdorur operatorin e rritjes ++, në këtë mënyrë: a++.

Operatorët e rritjes dhe të zvogëlimit

Tabela 1.3. Operatorët e rritjes dhe zvogëlimit

Operatorët	Veprimi	Shembulli	Rezultati
++	rritja e vlerës për 1	$x = 5$ x++	vlera e re e ndryshores x është 6
––	zvogëlimi i vlerës për 1	$x = 5$ x––	vlera e re e ndryshores x është 4

Operatori i rritjes shpeshherë quhet **operatori i inkrementimit**, kurse operatori i zvogëlimit **operator i dekrementimit**.

Operatorët e rritjes/zvogëlimit janë unarë dhe mund të përdoren me anë të prefiksit (para ndryshores, p.sh. ++n) dhe me anë të postfiksit (mbas ndryshores, p.sh. n++). Në formën prefikse, fillimisht zmadhohet/zvogëlohet vlera e ndryshores, pastaj ajo vlerë i shoqërohet ndryshores dhe zbatohet në punën e mëtejshme. Në anën tjetër, në formën e postfiksit, fillimisht lexohet vlera e ndryshores, dhe fill mbas ajo vlerë do të ndryshohet.

Për shembull, në qoftë se vlera e ndryshores x është e barabartë me 3, rezultati i veprimit të shprehjes y = ++x, është siç vijon: së pari x rritet për 1 (pra, fiton vlerën 4), dhe pastaj vlera e fituar x i shoqërohet ndryshores y. Kjo shprehje është ekuivalente me bashkësinë e shprehjeve:

$x = x + 1;$
$y = x;$

Nëse është dhënë shprehja y = x++, së pari vlera e ndryshores x (3) i shoqërohet y, pastaj x rritet për 1 (x fitohet vlera 4). Kjo shprehje është ekuivalente me bashkësinë e shprehjeve:

$y = x;$
$x = x + 1;$

Shembull 2.
Të përcaktojmë vlerën e ndryshoreve x, y dhe z mbas kryerjes së veprimeve:

x = 1
y = 1
z = (x + (++y)) * 3.

Zgjidhja: Ndryshorja x përmban vlerën 1, y rritet në 2, dhe mbas merr pjesë në njehsimin e shprehjes: $z = (1 + 2) * 3$, ashtu që z fiton vlerën 9.

1.2.4. Operatorët relacionalë dhe logjikë

Operatorët relacionalë

Operatorët logjikë

Në algoritme shpeshherë përdoren **operatorët e krahasimit**, të cilët quhen edhe operatorë relacionalë, sepse pasqyrojnë raportet midis madhësive. Në tabelën 1.4. janë të përmendur operatorët e krahasimit. Operatori logjik që i përgjigjet radhitjes së barazimit shënohet me ==, me qëllim që të dallohet nga simboli = që simbolizon operatorin e shoqërimit. Operatorët e krahasimit (radhitjes) janë shënuar në Tabelën 1.4, kurse operatorët logjikë në Tabelën 1.5.

Tabela 1.4. Operatorët e krahasimit

OPERATORI	PËRSHKRIMI
>	më e madhe
<	më e vogël
==	radhitja e barazimit
>=	më e madhe ose barazi
<=	më e vogël ose barazi
!=	i ndryshëm

Tabela 1.5. Operatorët logjikë

OPERATORI	PËRSHKRIMI
NOT	Mohimi, JO
AND	Konjukti, DHE
OR	Dizjunki, OSE
XOR	Ose ekskluzive

Shembulli 3.
Plotësoni tabelën me vlerat logjike të shprehjes për vlerat e përmendura të ndryshoreve.

Ndryshoret	Shprehja	Vlerat
a = 2 b = 3	a == b	
	(a > b) AND (NOT (b < 3))	
	(a != b) OR (a < 3)	

1.2.5. Prioritetet dhe vetia shoqëruese e operatorëve

Në qoftë se përdoren më shumë operatorë në një shprehje, atëherë është e nevojshme t'u kushtohet kujdes prioriteteve (përparësive) të operatorëve.

Tabela 1.6. Lista e operatorëve sipas prioriteteve (përparësive):

Prioriteti	Operatorët	Lista e operatorëve
1	Operatorët unarë	++, −−, NOT
2	Shumëzimi dhe pjesëtimi	*, /, div, mod
3	Mbledhja dhe zbritja	+, −
4	Operatorët e radhitjes	<, >, <=, >=
5	Barazimet dhe mosbarazimet	==, !=
6	Edhe logjike	AND
7	Ose logjike	OR
8	Operatorët e shoqërimit të vlerave	=

Lista e operatorëve e renditur sipas prioriteteve të veprimeve nga më i madhi deri te më i vogli është paraqitur në Tabelën 1.6. Prioritetet e operatorëve mund të ndryshohen duke përdorur kllapat ( ). Nëse dy operatorë kanë të njëjtin prioritet, atëherë merret parasysh vetia shoqëruese, e cila mund të realizohet nga ana e majtë në të djathtën ose nga ana e djathtë në të majtën.

Në qoftë se vetia shoqëruese vlen "majtas", së pari kryhet veprimi i parë i shikuar nga ana e majtë, pastaj sipas radhës edhe veprimet e tjera, duke lëvizur në anën djathtë. Kjo domethënë se shprehja $a+b+c+d$ njehsohet sipas formulës në vazhdim $(((a+b)+c)+d)$, sepse operatori binar + shoqërohet nga ana e majtë (vetia shoqëruese e majtë).

Shembulli 4.
Plotësoni tabelën me vlerat e ndryshoreve mbas kryerjes së veprimeve.

Të ndryshueshmet	Shprehjet	Vlerat
$a = 5$ $b = 3$	$c = a \bmod 3 + b$	$c =$
$a = 3$ $b = 2$	$r = a / b$	$r =$
$a = 7$ $b = 4$	$x = 2 * b / a$	$x =$

Shembulli 5.
Përcakto vlerat e shprehjes së mëposhtme:

$x = 24 \bmod 5 * 6 \bmod 5$

Rretho përgjigjen e saktë:

a) 0 b) 1 c) 3 d) 4

$x = 17 \bmod 3 * 7 \bmod 3$

Rretho përgjigjen e saktë:

a) 1 b) 2 c) 4 d) 5

Në rastet kur shprehja përmban operatorin unar dhe atë binar, operatori unar ka gjithmonë prioritet më të madh.

Operatorët prefiksë unarë dhe operatori e shoqërimit shoqërohen nga e djathta në të majtën. Operatorët e tjerë shoqërohen nga ana a majtë në të djathtë.

Shikoni shembujt e mëposhtëm:

Shprehja	Rezultati	Sqarimi
$x = 3;$ $z = ++x + 2;$	$x = 4,\ z = 6$	operator prefiks
$x = 3;$ $z = x---2;$	$z = (x--)-2;$ $x = 2,\ z = 1$	operator prefiks
$y = -1 + 2$	$y = 1$	operatorët unarë kanë prioritet më të lartë sesa ata binarë
$m = x - y - z$	$m = (x - y) - z$	operatori – shoqërohet nga ana e majtë në të djathtë
$x = y = z$	$x = (y = z)$	operatori = shoqërohet nga e djathta në të majtë

Në shprehje komplekse nganjëherë nuk është lehtë të përcaktohet renditja e kryerjes së veprimeve, prandaj kllapat duhen përdorur edhe kur nuk janë të nevojshme, sepse përmirësojnë lexueshmërinë e kodit.

Shembulli 6.
Plotëso tabelën me vlerat e ndryshoreve mbas veprimeve të kryera

Ndryshorja	Shprehja	
$a = 5$ $b = 7$	$c = b++ + 3 * a$	$c =$
$a = 3$ $b = 2$	$r = a * 3 ---b$	$r =$
$a = 7$ $b = 4$	$x = a * b + a++$	$x =$

Pyetje dhe detyra për kontrollin

1. Çfarë do të jetë vlera e ndryshoreve m dhe n mbas kryerjes së operacioneve:

a) $m = 15$
$n = m + 1$
$m = m - 20$

b) $m = 5$
$n = 10$
$m = m \bmod n$
$n = n * m$

c) $m = 2{,}3$
$n = 8{,}9$
$m = m + n$
$m = n - m$
$n = n - m$

d) $m = 5$
$n = 55$
$t = m$
$m = n$
$n = t$

Puno vetë

1. a) Përcakto vlerën e ndryshoreve x, y, z mbas vargut të veprimeve:

a) $x = 3$
$z = x++ - 5$

b) $y = x = 3$
$z = ++x - y--$

2. b) Shkruani vargun sipas të cilit, ndryshores y i shoqërohet vlera sipas formulës:

$$y = \frac{x^2 - 2x + 1}{x^2 - 1},\ \text{për } x = 5.$$

1.3. Tipat e skemave algoritmike

Në vitin 1966 matematikanët Korado Bom dhe Gjuzepe Jakopini kanë vërtetuar se secili algoritëm mund të shprehet me ndihmën e sekuencave, vendimeve dhe përsëritjeve. Prandaj ekzistojnë tri skema elementare algoritmike:

1. Skemat lineare algoritmike
2. Skemat e degëzuara algoritmike
3. Skemat ciklike algoritmike

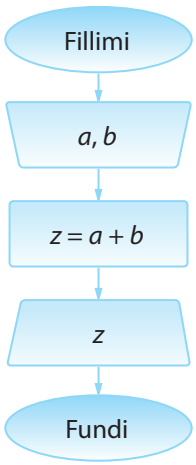
Me kombinimin e skemave elementare algoritmike fitohen ***skemat komplekse algoritmike***.

1.3.1. Skemat lineare algoritmike

Tek ***skemat lineare algoritmike*** hapi algoritmik kryhet saktësisht njëherë, d.m.th. mbas kryerjes së një hapi algoritmik kalojmë tek hapi algoritmik që pason menjëherë. Domethënë, hapat algoritmikë kryhen njëri mbas tjetrit, sipas renditjes së shënuar.

Skemat lineare algoritmike zakonisht shërbejnë për njehsime të ndërlikuara mbi vlerat hyrëse. Pasojnë disa shembuj, ku krahas algoritmit, me qëllim sqarimi, jepet edhe shënimi i tij tekstual. Shënimi tekstual nuk është i domosdoshëm gjatë shkruarjes së algoritmeve.

Algoritmi 1. Shkruani skemën algoritmike në të cilën lexohen dy numra dhe printohet shuma e tyre.



Fillimi i algoritmit.

Hyrje (leximi) e dy numrave ***a*** dhe ***b***.

Ndryshores ***z*** i shoqërohet rezultati i mbledhjes së numrave ***a*** dhe ***b***.

Shënimi i rezultatit të shoqëruar ndryshores ***z***.

Përfundimi i algoritmit.

Algoritmi 2. Shkruani skemën algoritmike në të cilën lexohen notat në provimin e parë dhe të dytë me shkrim nga programimi dhe shënohet nota mesatare.

Në qoftë se x dhe y paraqesin notat e provimit të parë me shkrim, gjegjësisht të dytë, nota mesatare llogaritet si mesi i tyre aritmetik, sipas formulës:

$$z = \frac{x + y}{2}.$$

Tani shkruajmë skemën algoritmike:

Skemat lineare algoritmike!

Shembull ilustrues i skemave lineare algoritmike paraqet procedura e hapjes së profilit në Facebook (duke supozuar që nuk gabohet gjatë shënimit të të dhënave në fushat e caktuara).

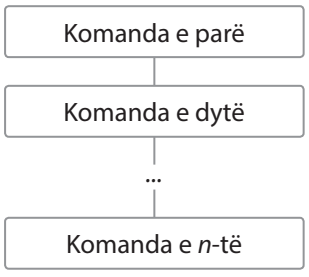
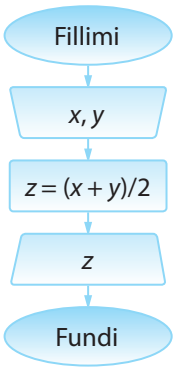


Figura 1.2.
Skema e thjeshtë lineare algoritmike



Notat në provimin e parë dhe të dytë me shkrim janë vlerat hyrëse, prandaj nevojitet të shkruhen vlerat e ndryshoreve ***x*** dhe ***y***.

Formula e shënuar përdoret për njehsimin e vlerës mesatare aritmetike (të ndryshoreve *x* dhe *y*) që i shoqërohet ndryshores ***z***.

Shënimi i rezultatit.

Gjatë zgjidhjes së shumë problemave, shpeshherë përdoren disa funksione standarde matematike, siç është ngritja në fuqi, nxjerrja e rrënjës, funksionet trigonometrike etj. Në tabelën 1.7. është paraqitur lista e funksioneve matematike me simbole që përdoren në disa gjuhë programuese siç është Basic dhe Pascal. Më vonë në Tekst, do të përmenden dhe do të përshkruhen në hollësi këto funksione në gjuhën e programimit Java.

Tabela 1.7. Lista e funksioneve matematike

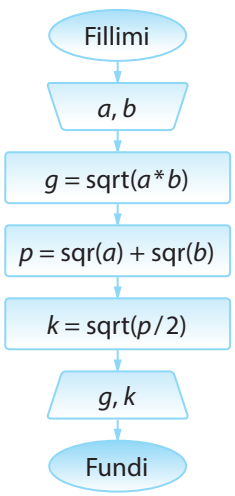
PËRSHKRIMI	SIMBOLI
Rrënja katrore	sqrt()
Ngritja në katror	sqr()
Ngritja në fuqi	^
Funksioni eksponencial	exp()
Funksionet trigonometrike: sin, cos, tg, ctg	sin(), cos(), tg(), ctg()
Vlera absolute	abs()

Algoritmi 3. Shkruani skemën algoritmike për njehsimin e mesit gjeometrik dhe mesit kuadratik të dy numrave pozitivë të dhënë *a* dhe *b*.

Nga matematika janë të njohura këto formula për njehsimin e mesit gjeometrik dhe kuadratik:

$$g = \sqrt{ab} \text{ dhe } k = \sqrt{\frac{a^2 + b^2}{2}}.$$

Tani shënojmë skemën algoritmike.



Leximi i vlerave të ndryshoreve ***a*** dhe ***b***.

Ndryshores ***g*** (mesit **g**jeometrik) i shoqërojmë vlerën $\sqrt{ab}$.

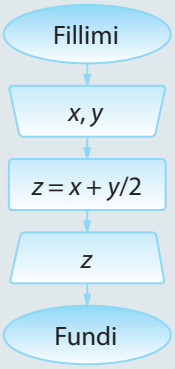
Meqenëse formula për njehsimi e mesit kuadratik është e ndërlikuar, vlera e tij mund të njehsohet me disa hapa. Prandaj përkufizohet ndryshorja ***p*** (ndryshorja ndihmëse) të cilës i shoqërohet vlera e numëruesit të thyesës nën rrënjë.

Në hapin vijues, ndryshores ***k*** (mesit kuadratik) i shoqërohet vlera: $\sqrt{\frac{p}{2}}$.

Shënimi i vlerave të njehsuara për mesin gjeometrik dhe kuadratik.

?

Është me rëndësi të theksohet se skema algoritmike e mëposhtme nuk jep zgjidhjen e saktë të problemit të definuar në Algoritmin 2. Pse?



Shembulli 7.
Shkruani skemën algoritmike nëpërmjet të cilës njehsohet perimetri, sipërfaqja dhe gjatësia e diagonales së drejtkëndëshit me gjatësi të brinjëve të dhëna.

Shembulli 8.
Shkruani skemën algoritmike që përcakton gjatësinë e fluturimit në minuta, në qoftë se janë të njohura terminët e fluturimit dhe të aterrimit të aeroplanit (koha e fluturimit dhe e aterrimit janë në të njëjtën ditë dhe janë dhënë në formatin 24 orësh: *h* orë, *m* minuta).

Shembulli 9.
Shkruani skemën algoritmike nëpërmjet të cilës njehsohet vlera e shprehjes

$$\frac{\sin(2(a+b-c))}{\cos(2a+b-c)}$$

për vlerat hyrëse të parametrave *a*, *b* dhe *c*.

Shembulli 10.
Shkruani skemën algoritmike me të cilën njehsohet perimetri dhe sipërfaqja e trekëndëshit, kulmet e të cilit janë përcaktuar me anë të koordinatave në rrafsh.

Gjatë zgjidhjes së problemave të ndryshme nga matematika, fizika, kimia, elektroteknika, përdoren konstante të shumta. Në tabelën 1.8. janë përfshirë vetëm disa nga ato konstante.

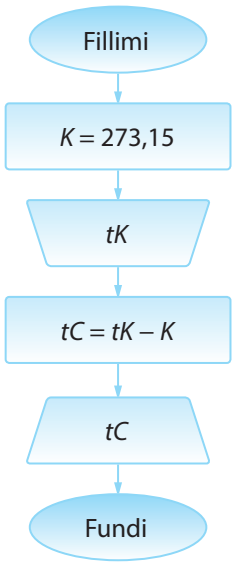
Simboli	Përshkrimi	Vlera
π	Numri i Ludolfit	3,14
g	Konstanta e gravitacionit (m/s^2)	9,81
v	Shpejtësia e zërit	340
B	bajti	8b

Tabela 1.8. Shembuj konstantash që përdoren shpesh

Shembulli 11.
Shkruani skemën algoritmike, me të cilën kryhet zbërthimi i sasisë së lëngut nga gallonët në litra në qoftë se 1 gallon = 4,54 litra.

Algoritmi 4. Shkruani skemën algoritmike me të cilën vlera e dhënë e temperaturës e shprehur në shkallën e Kelvinit zbërthehet në vlerën përkatëse në shkallën e Celsiusit.

Ndryshimi midis vlerës së temperaturës në shkallën e Kelvinit (°K) dhe shkallës së Celsiusit (°C) është 273,15, d.m.th. vlen formula °C = °K − K, ku K është konstanta, vlera e të cilës është 273,15.



Për dallim nga shembujt paraprak, këtu për herë të parë paraqitet konstanta ***K***, të cilës i shoqërohet vlera në fillim të algoritmit.

Konstanta është gjithmonë një numër i njëjtë dhe vlera e saj nuk ndryshon gjatë hapave algoritmikë.

Temperatura në shkallën e kelvinit shënohet me *tK*, sepse shenja *K* është rezervuar për vlerën e konstantes.

Vlera e temperaturës në Celsius i shoqërohet ndryshores *tC*.

Shënimi i rezultatit.

Vini re se, për ndryshim nga shembujt paraprak, në formulimin e problemit nuk është theksuar se cilat vlera merren si parametra hyrës të algoritmit. Qasja e njëjtë do të zbatohet edhe në vazhdim, d.m.th. nga ju pritet që mënyrë në të pavarur, në bazë të përshkrimit të problemit që zgjidhet, të përcaktoni se cilat janë parametrat hyrës.

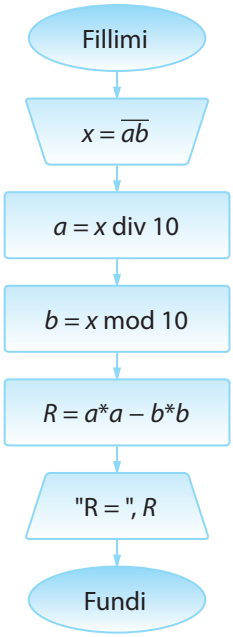
Që të zgjidhësh një detyrë duhet t'u përgjigjesh pyetjeve të mëposhtme:

1. Cilat madhësi janë të njohura dhe cilat nuk janë të njohura?
2. Cilat janë raportet e mundshme midis madhësive të njohura dhe të panjohura? Me fjalë të tjera, të zgjidhet vargu më i përshtatshëm i veprimeve mbi madhësitë e njohura me qëllim që të arrihet deri te rezultatet e dëshiruara.
3. Çfarë duhet të jetë rezultati?

Shembulli 12.
Shkruani skemën algoritmike, me të cilën njehsohet herësi i shifrës së parë dhe të fundit të numrit pesëshifror.

Algoritmi 5. Shkruani skemën algoritmike me të cilën njehsohet ndryshimi i katrorëve të shifrave të numrit dyshifror.

Për përpilimin e këtij algoritmi do të përdorim operatorët *mod* dhe *div*.



Hyrja e numrit dyshifror (p.sh. $x = 53$). Fakti se bëhet fjalë mbi një numër dyshifror, shënohet me $\overline{ab}$ (nëpërmjet kësaj nuk përkufizohen ndryshoret ***a*** dhe ***b***, por vetëm deklarohet që *a* dhe *b* paraqesin shifrat e dhjetësheve dhe njësheve të numrit *x*).

Tani përdoret ndryshorja ***a***, të cilës i shoqërohet vlera e shifrës së dhjetësheve ($a = 53$ div 10).

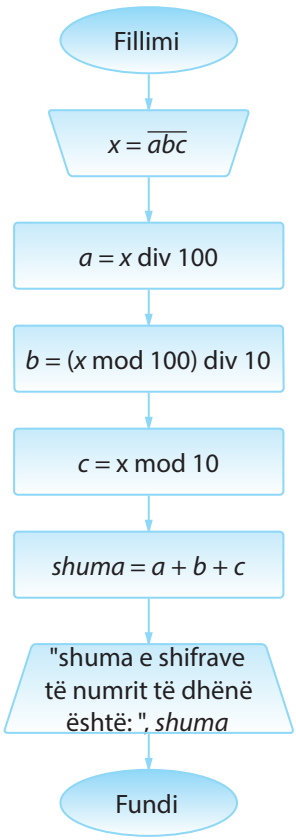
Ndryshores ***b*** i shoqërohet vlera e shifrës së njësheve ($b = 53$ mod 10).

Njehsimi i ndryshimit të katrorëve të shifrave *a* dhe *b* (***R*** = 5*5 − 3*3).

Shënimi i rezultatit ($R = 16$). (Nëse kemi dëshirë, mund të shtojmë tekst pranë rezultatit brenda " ", si në shembullin "R=", duke vazhduar me shënimin e ndryshores R të lidhur me presjen (,) me tekstin e përmendur, si në shembullin: "R = ", *R*.)

Algoritmi 6. Shkruani skemën algoritmike me të cilën njehsohet shuma e shifrave të numrit treshifror.

Në mënyrë të ngjashme si në shembullin paraprak, nevojitet, që me radhë të përcaktohen të gjitha shifrat e numrit treshifror të dhënë.



Futja e numrit treshifror ***x***.

Ndryshores ***a*** i shoqërohet vlera e shifrës së qindësheve, që përftohet si rezultat i pjesëtimit të plotë me 100.

Ndryshores ***b*** i shoqërohet vlera e shifrës së dhjetësheve të numrit $x = \overline{abc}$, që paraqet shifrën e dhjetësheve të numrit dyshifror $\overline{bc}$ (të përftuar sipas formulës x mod 10).

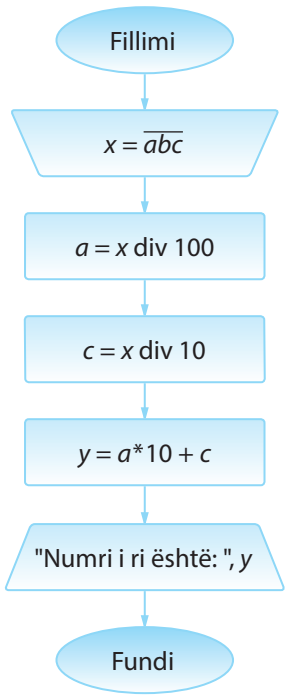
Ndryshores ***c*** i shoqërohet vlera e shifrës së njësheve, që paraqet mbetjen gjatë pjesëtimit me 10.

Njehsimi i shumës së shifrave të numrit të dhënë.

Shënimi i rezultatit.

Shembulli 13.
Shkruani skemën algoritmike nëpërmjet të cilës lexohet numri treshifror, dhe nga ai formohet numri treshifror me shifra në renditje të anasjelltë (p.sh. nga $x = abc$ formohet $y = cba$).

Algoritmi 7. Shkruani skemën algoritmike me të cilën lexohet numri treshifror dhe nga ai formohet numri dyshifror nga shifrat e dhjetësheve dhe njësheve të numrit të dhënë (p.sh. nga numri 345 të formohet numri 35: $y = 3*10 + 5$).



Hyrja e numrit treshifror ***x***.

Ndryshoreve ***a*** dhe ***c*** iu shoqërohen vlerat e shifrave të qindësheve dhe njësheve të numrit ***x***.

Formimi i numrit dyshifror ***y***.

Shënimi i rezultatit.

Shembulli 14.
Nxënësi ka filluar të punojë kuizin (pyetësorin) për kontrollin e njohurive mbi algoritmet në x orën, y minuta dhe z sekonda, dhe për përgjigje në pyetje ka shpenzuar q sekonda. Shkruani skemën algoritmike me të cilën përcaktohet koha kur nxënesi ka përfunduar përpunimin e kuizit.

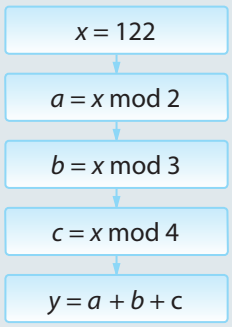


Figura 1.3.

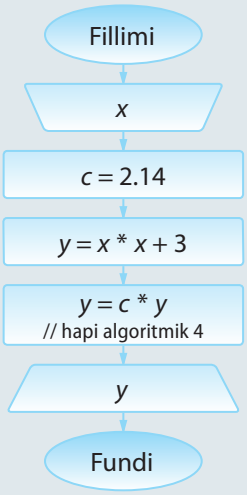


Figura 1.4.

Pyetje dhe detyra kontrolli

1. Kur përdoren skemat lineare algoritmike?

2. A duhet çdo skemë algoritmike të ketë vlerat hyrëse dhe dalëse?

3. Për pjesën e skemës algoritmike nga figura 1.3, vlera e ndryshores y mbas kryerjes së hapave algoritmik është:

a) 1 b) 2 c) 3 d) 4 e) Asnjë nga më lart.

4. Cili nga pohimet është i saktë për skemën algoritmike në figurën 1.4?

a) Vlera e ndryshores *y* varet nga vlera e ndryshores *x*.

b) Pavarësisht nga vlera e ndryshores *x*, vlera e ndryshores *y* është 2.14.

c) Në skemë ekziston gabimi në hapin algoritmik 4.

Puno vetë

1. Shkruani skemën algoritmike me ndihmën e të cilës:

- kapaciteti i memories i paraqitur në bajte zbërthehet në bite;
- zbërthen këndin e dhënë në radianë, në shkallë, duke pasur parasysh formulën

$$\alpha_{step} = \frac{\alpha_{rad} \cdot 180}{\pi};$$

- zbërthen temperaturën nga shkalla e Celsiusëve në shkallën Farenhajt, nëse formula përkatëse duket si mëposhtë:

Temperatura sipas Farenhajtit = (*temperatura sipas Celsiusit*) * 1,80 + 32;

- njehson vlerën e funksionit $f(x) = \frac{x^2 - 2x + 1}{x^2 - 1}$ sipas vlerës së argumentit hyrës *x*;
- i ndihmon një fizikani të ri që në bazë të gjatësisë së shkopit dhe të hijes së tij përcaktojë këndin të cilit e formojnë rrezet e diellit me sipërfaqen e tokës;
- * njehson vlerën e funksionit

$$y = \sqrt{\frac{e^{\frac{2x}{3}} + 1.5\sin 3x}{3\cos 2x - \frac{1+x}{2e^{-x}}}}$$

sipas vlerës së argumentit hyrës *x*.

1.3.2. Skemat algoritmike të degëzuara (të kushtëzuara)

Zgjidhja e shumicës së detyrave kërkon zbatimin e logjikës më komplekse, sesa ajo që ka qenë prezent në skemat algoritmike që kemi zgjidhur deri tani. Shpeshherë ndodh që gjatë kryerjes së hapave algoritmikë, në varësi të plotësimit të një kushti, sillet vendimi se cili hap i ardhshëm do të kryhet.

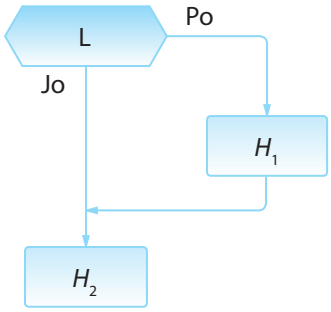
Skemat algoritmike të degëzuara (të kushtëzuara)

Në skema të tilla ekzistojnë blloqet e hapave algoritmikë të cilët për disa vlera të të dhënave hyrëse asnjëherë nuk kryhen, dhe skema e tillë quhet ***skema algoritmike e degëzuar (e kushtëzuar)***. Hapi algoritmik në të cilin në bazë të plotësimit të kushtit të përkufizuar sillet vendimi mbi hapin vijues algoritmik quhet ***hapi algoritmik i kushtëzuar*** ose, shkurtimisht, ***degëzimi***. *Një skemë lineare algoritmike e degëzuar (kushtëzuar) mund të ketë më shumë se një degëzim.*

Simboli grafik për degëzim është 

, ku në mes jepet kushti logjik L. Përgjigjet në kushtin logjik të vendosur mund të jenë **PO** dhe **JO**, gjegjësisht **T** – saktë (`true`) dhe **F** – pasaktë (`false`). Në varësi të plotësimit/mosplotësimit të kushtit të përcaktuar, ekzekutimi (kryerja) vazhdon në degën e shënuar me Po (True) / Jo (False).

a)



b)

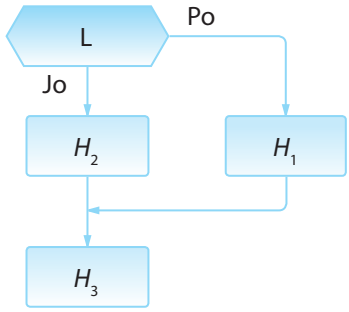


Figura 1.3. Shembuj të degëzimeve të njëfishta

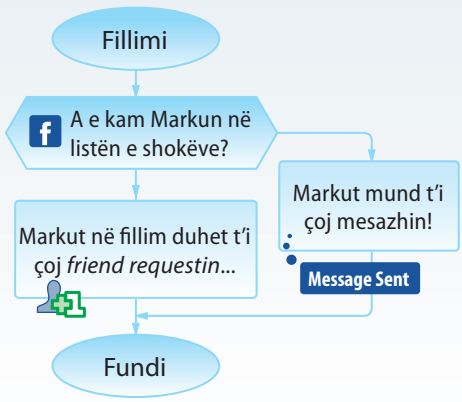
Në figurën 1.3.a) është dhënë një shembull degëzimi. Në qoftë se vlera e shprehjes logjike (pohimit) L është e saktë (`true`), atëherë kryhet hapi algoritmik H_1 dhe kalohet në hapin algoritmik vijues H_2; nëse vlera e shprehjes është e pasaktë (`false`), kalohet në hapin algoritmik vijues H_2.

Në figurën 1.3. b) është paraqitur gjithashtu një shembull degëzimi. Në qoftë se vlera e shprehjes logjike L është e saktë (`true`), kryhet hapi algoritmik H_1; nëse është e pasaktë (`false`), kryhet hapi algoritmik H_2. Meqenëse janë kryer hapat algoritmikë H_1 ose H_2, kalohet në hapin algoritmik H_3 që pason.

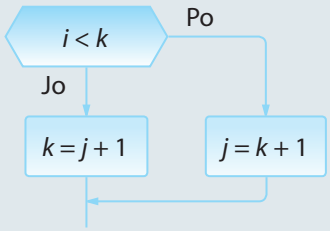
Algoritmi 8. Shkruani skemën algoritmike me të cilën dy numra të dhënë shënohen në renditje rënëse.

Për zgjidhjen e kësaj detyre nevojitet që në varësi të raportit të vlerave të numrave *a* dhe *b* të përcaktojmë vlerën më të madhe dhe vlerën më të vogël dhe në atë renditje të paraqiten. Që të realizojmë një plan të tillë, do të përdorim dy ndryshore ndihmëse *max* dhe *min* të cilave u shoqërojmë vlerat *a* ose *b*, në varësi nga raporti midis tyre.

Një shembull ilustrues i skemës së degëzuar algoritmike është dërgimi i shokut/shoqes së shkollës të një mesazhi nëpërmjet facebook-ut. Në fillim duhet të verifikojmë nëse atë shok/shoqe e kemi në facebook në listën e shokëve apo jo.

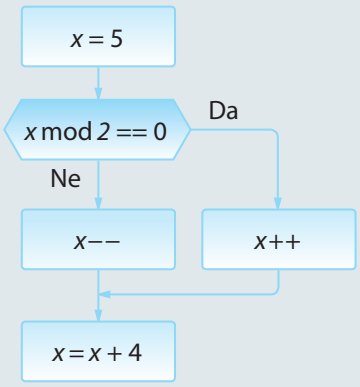


Shembulli 15.
Cilat vlera do t'i përftojnë ndryshoret *j* dhe *k* mbas kryerjes së degëzimeve për vlerat e dhëna të parametrave hyrës?

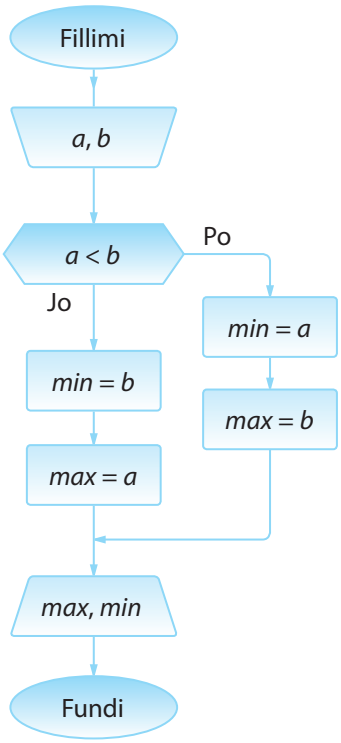


a) $j = 2,\ k = 3$
b) $j = 3,\ k = 3$
c) $j = 2,\ k = 2$

Shembulli 16.
Cilën vlerë do të përftojë ndryshorja *x* mbas kryerjes së hapave algoritmikë të mëposhtëm.



a) 8
b) 10
c) 0
d) 5



Në rast se $a < b$, atëherë ndryshores *min* i shoqërohet vlera *a*, kurse ndryshores *max* vlera *b*.

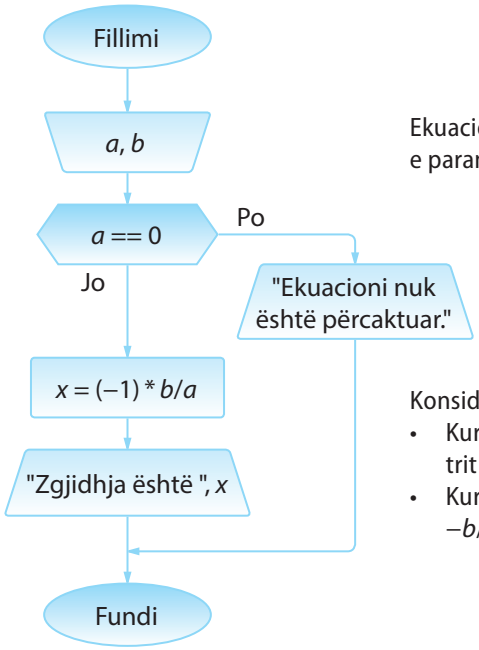
Në rastet e tjera, d.m.th kur $a = b$ ose $a > b$, ndryshores *min* i shoqërohet vlera *b*, kurse ndryshores *max* vlera *a*.

Vini re se kushti mund të lexohej edhe si *a* <= *b*, sepse në rastin e barazimit të vlerave *a* dhe *b*, edhe vlerat e *min* e *max* do të ishin të barabarta.

Paraqitja e vlerave në renditje rënëse.

A mund të shkruhet Algoritmi 8 pa futjen në përdorim të ndryshoreve shtesë *min* dhe *max*?

Algoritmi 9. Të shkruhet skema algoritmike për zgjidhjen e ekuacionit linear $ax + b = 0$, $a \neq b$.



Ekuacioni përcaktohet duke dhënë vlerat e parametrave *a* dhe *b*.

Konsiderojmë dy raste:
- Kur $a = 0$ (hyrje jokorrekte e parametrit *a*),
- Kur $a \neq 0$, atëherë zgjidhja është $-b/a$.

Shembulli 17.
Shkruani skemën algoritmike për njehsimin e vlerës *Z* në bazë të vlerave hyrëse *a* dhe *b*, sipas formulës:

$$Z = \begin{cases} a+b, a<b \\ a-b, a \geq b \end{cases}$$

Shembulli 18.
Të përpilohet skema algoritmike nëpërmjet të cilës gjenden shumat e vlerave pozitive dhe negative të numrave të dhënë *a*, *b* dhe *c*.

Shembulli 19.
Formo skemën algoritmike me anë të cilës për numrat e dhënë *x* dhe *y* njehsohet *z*, sipas formulës:

$$z = \frac{\min(x,y) + 0{,}5}{1 + \max^2(x,y)}.$$

Ky është shembull i skemës algoritmike në të cilën mesazhi dalës nuk është unik, por varet nga vlerat e parametrave hyrës dhe plotësimit të kushteve të përcaktuara.

Në dy shembujt e mësipërm kemi pasur një degëzim (të ashtuquajtur degëzim të njëfishtë). Në shembullin vijues, do të duhen më shumë degëzime, dhe do të tregojmë se skema algoritmike në rastin e përgjithshëm nuk është unike, por një detyrë e dhënë mund të zgjidhet me kombinime të ndryshme të dy apo më shumë degëzimeve.

Shembulli i kombinimit të më shumë degëzimeve (i ashtuquajtur ***degëzim i futur***) është paraqitur në figurën 1.4.

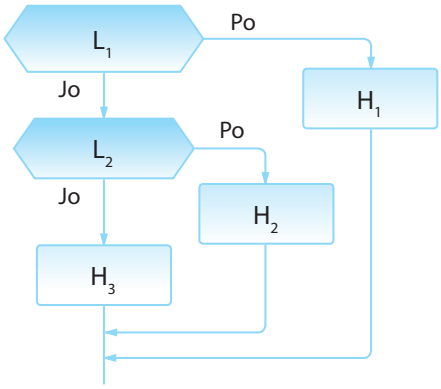


Figura 1.4. Shembull i degëzimit të futur.

Në qoftë se vlera e shprehjes logjike L_1 është e saktë (`true`), kryhet hapi algoritmik H_1, kurse nëse është e pasaktë (`false`), verifikohet vlera e shprehjes logjike L_2. Nëse shprehja logjike L_2 është e saktë (`true`), kryhet hapi algoritmik H_2, kurse në të kundërtën hapi algoritmik H_3. Meqenëse janë kryer hapat algoritmikë H_1 ose H_2 ose H_3, vazhdon ekzekutimi i skemës algoritmike.

Thellësia e futjes së degëzimit varet vetëm nga problemi që zgjidhet.

Shembulli 20.
Shkruani skemën algoritmike me të cilën verifikohet nëse është numri i dhënë numër i Armstrongut. Numri i Armstrongut është numri, shuma e kubeve të shifrave të të cilit është e barabartë me numrin e dhënë.

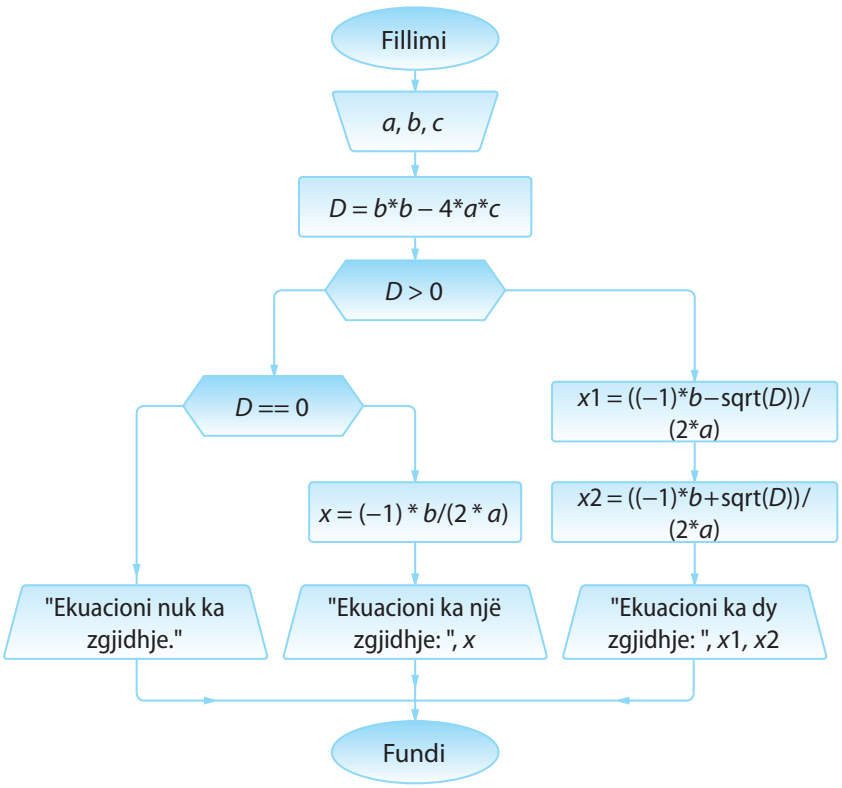
Shembulli 21.
Shkruani skemën algoritmike për njehsimin e dy ekuacioneve lineare me dy të panjohura:

$a_1x + b_1x = c_1$

$a_2x + b_2x = c_2$

Algoritmi 10. Të shkruhet skema algoritmike për zgjidhjen e ekuacionit kuadratik $ax^2 + bx + c = 0$, $a \neq 0$, në bashkësinë e numrave realë.

Dihet se në varësi të vlerës së determinantës $D = b^2 - 4ac$, ekuacioni kuadratik ka një zgjidhje, dy zgjidhje ose nuk ka zgjidhje midis numrave realë.



Ekuacioni kuadratik përcaktohet me anë të vlerave të koeficienteve ***a***, ***b*** dhe ***c***.

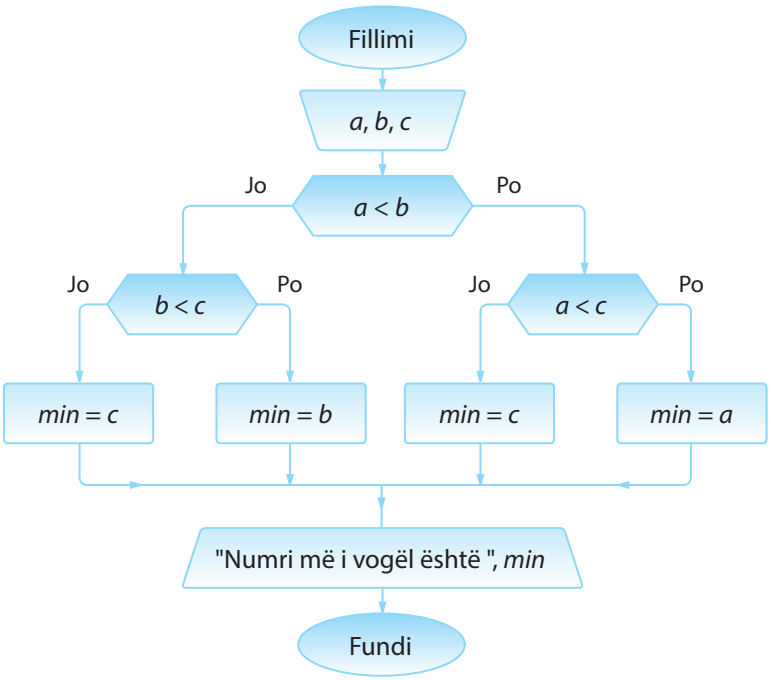
Ndryshores ***D*** i shoqërohet vlera e determinantes.

Verifikohet a është $D > 0$. Atëherë ekuacioni ka dy zgjidhje, vlerat e të cilave iu shoqërohen ndryshoreve x_1 dhe x_2.

Nëse nuk vlen $D > 0$, verifikohet nëse vlen $D = 0$. Në atë rast, ekuacioni ka një zgjidhje unike që i shoqërohet ndryshores ***x***.

Nëse nuk vlen $D > 0$ dhe nuk vlen $D = 0$, do-methënë $D < 0$ dhe ekuacioni nuk ka zgjidhje në bashkësinë e numrave realë.

Algoritmi 11. Të shkruhet skema algoritmike e cila për madhësi hyrëse ka tre numra dhe rezultati është numri më i vogël ndërmjet tyre.



Në fillim kontrollohet nëse është $a < b$. Nëse po, kandidatët për vlerën më të vogël janë numrat ***a*** dhe ***c***, të cilët duhet të renditen.

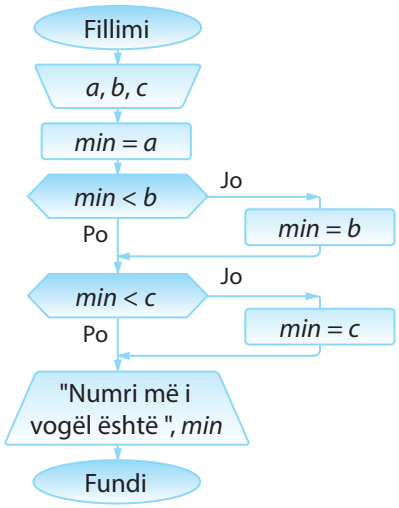
Në qoftë se $a < c$, atëherë a është numri më i vogël, vlera e të cilit i shoqërohet ndryshores *min*.

Nëse nuk vlen $a < c$, d.m.th. $c \le a$, kurse paraprakisht vlen $a < b$, vlera më e vogël është ***c***.

Në fund, nëse në degëzimin e parë nuk vlen $a < b$, krahasohen numrat b dhe c me qëllim që të përftohet vlera më e vogël, në mënyrë të njëjtë si në rastin paraprak.

Shembulli 22.
Shkruani skemën algoritmike e cila për madhësi hyrëse ka katër numra dhe rezultati është numri më i vogël ndërmjet tyre.

Mënyra II. Problema e dhënë mund të zgjidhet edhe në mënyrën e mëposhtme: numri i parë, d.m.th. numri a përkohësisht deklarohet si numri më i vogël. Duke e krahasuar, sipas radhës me numrat b dhe c, ndryshores *min* i shoqërohet në atë moment vlera minimale. Në fund, në ndryshoren *min* ndodhet vlera e numrit më të vogël nga lista e dhënë.



Ndryshores *min* i shoqërohet vlera e numrit të parë, d.m.th vlera e numrit ***a***. Pastaj bëhet kontrolli nëse numri ***a*** është minimal (në lidhje me numrin ***b***). Në qoftë se rezultati është afirmativ, kurrfarë ndryshimesh nuk nevojiten, kurse në të kundërtën, ndryshores ***min*** i shoqërohet vlera ***b*** si vlera në atë moment e numrit minimal (midis numrave a dhe b).

Tani kontrollohet nëse vlera e ***min*** është më e vogël mbas krahasimit me numrin ***c***. Në qoftë se jo, vlera e re e ndryshores ***min*** është ***c***.

Paraqitja e rezultatit, d.m.th. vlerës së ndryshores ***min***.

Shembulli i dhënë tregon se nuk ekziston zgjidhja unike algoritmike e problemit të caktuar. Për algoritme të tilla themi se janë ***ekuivalente***. Midis algoritmeve ekuivalente duhet të zgjidhet algoritmi që në mënyrë më efikase sjell deri te rezultati. Kriteret për zgjedhjen e algoritmit më efikas janë të ndryshme:

- Shpejtësia me e madhe e kryerjes së algoritmit;
- Angazhimi minimal i hapësirës në memorie;
- Struktura sa më e thjeshtë etj.

Mbi karakteristikat e tjera të skemave algoritmike do të bëhet fjalë në fund të këtij kapitulli.

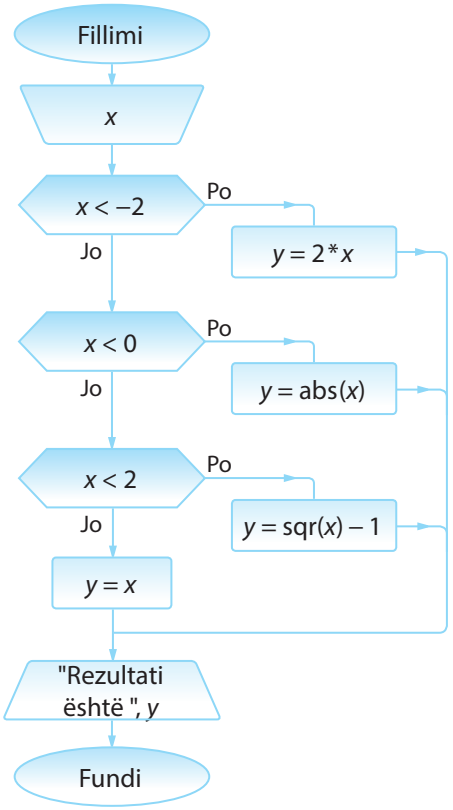
Shembulli 23.
Formo skemën algoritmike me të cilën për gjatësitë e dhënë a, b, c (nëse mund të ndërtohet trekëndëshi), njehsohet sipërfaqja e trekëndëshit sipas formulës së Heronit:

$$P = \sqrt{s(s-a)(s-b)(s-c)},$$

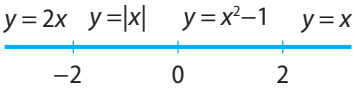
ku $s = (a+b+c)/2$.

Algoritmi 12. Të shkruhet skema algoritmike me të cilën për vlerën hyrëse x njehsohet vlera y sipas formulës:

$$y = \begin{cases} 2x, & x < -2 \\ |x|, & -2 \le x < 0 \\ x^2 - 1, & 0 \le x < 2 \\ x, & x \ge 2 \end{cases}$$

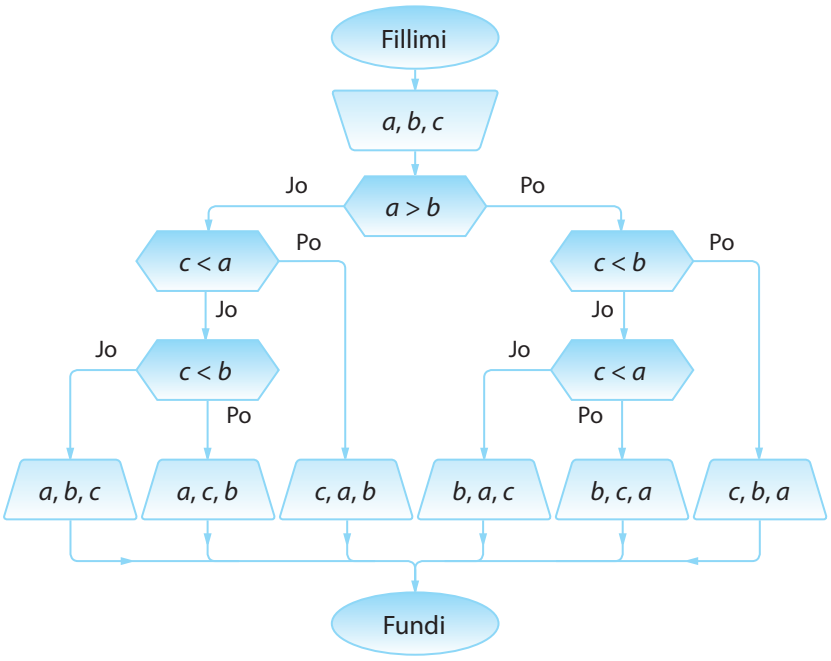


Sipas formulës së dhënë, boshti numerik ndahet në katër intervale dhe në secilin interval y njehsohet sipas nënformulës së caktuar:



Për vlerën e dhënë të parametrit x, kontrollohet sipas radhës përkatësia e secilit interval.

Algoritmi 13. Të shkruhet skema algoritmike për renditjen (radhitjen) e tre numrave realë në renditjen jorënëse:



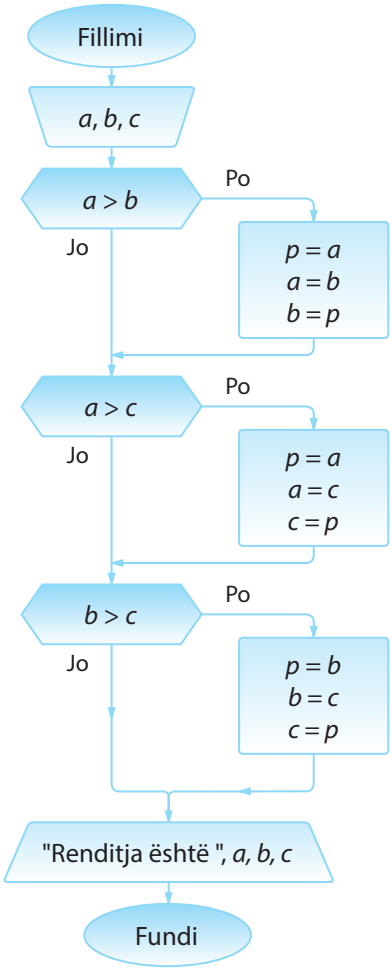
Nisemi nga hipoteza se vlerat e a, b dhe c kanë renditje të kërkuar dhe bëjmë korrigjime përkatëse.
Në qoftë se vlen $a > b$, domethënë se renditja e rregullt e këtyre elementeve është ***b a***. Tani duhet të përcaktojmë pozitën e elementit c në lidhje me ta.

Nëse vlen $c < b$, renditja është ***c b a***. Në të kundërtën kontrollohet kushti $c < a$. Në atë rast renditja jorënëse duket ***b c a***, kurse në të kundërtën ***b a c***.

Tani kthehemi në degëzimin e parë. Në qoftë se $a \le b$, atëherë pozicioni i elementit c përcaktohet në mënyrë të ngjashme si në shqyrtimin paraprak.

Shembulli 24.
Në shitore ndodhen paketimet e CD-ve nga k1 copë, k2 copë dhe k3 copë. Të përpilohet skema algoritmike me të cilën kontrollohet nëse blerësit, i cili ka porositur k copë CD mund t'i dorëzohen artikujt e kërkuar pa hapjen e paketimeve (translokimin nga to).

Vërë re se skema algoritmike për algoritmin 13 është ekuivalente me skemën e mëposhtme:



Në degëzimin e parë, në qoftë se $a > b$, duke futur në përdorim ndryshoren ***p*** kryhet ndryshimi i pozicioneve të elementeve ***a*** dhe ***b***. Në këtë mënyrë, mbas degëzimit të parë, numat *a* dhe *b* janë në renditjen korrekte ndërmjet tyre.

Me anë të degëzimit vijues, në të njëjtën mënyrë, në qoftë se nevojitet, kryhet zëvendësimi i vlerave ***a*** dhe ***c***, gjegjësisht ***b*** dhe ***c***.

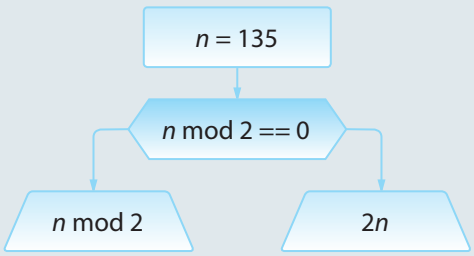


Figura 1.5.

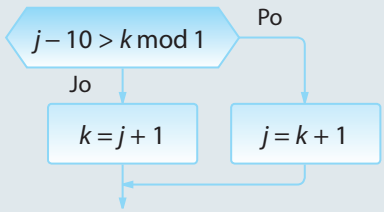


Figura 1.6.

Pyetje dhe detyra kontrolli

1. Kur zbatohen skemat lineare algoritmike?

2. A mund të përkufizohen kushtet e degëzimit si shprehje komplekse logjike, p.sh. **(*a* == 5) *and* (*b* < 3)** ose është e domosdoshme të zbatohen më shumë degëzime me shprehje të thjeshta logjike (në këtë shembull do të ishin dy degëzime me kushtet ***a*** == **5** dhe ***b*** < **3**)?

3. Cila vlerë do të paraqitet mbas kryerje së pjesës së skemës algoritmike të paraqitur në figurën 1.5.?

4. Cilat vlera do të kenë ndryshoret *j* dhe *k* mbas kryerjes së degëzimeve në figurën 1.6. për vlerat e dhëna të parametrave hyrës:

a) $j = 1,\ k = 3$

b) $j = 1,\ k = 2$

5. Cilën vlerë do të ketë ndryshorja x mbas kryerjes së hapave algoritmikë nga figura 1.7?

a) 8

b) 10

c) 0

d) 5

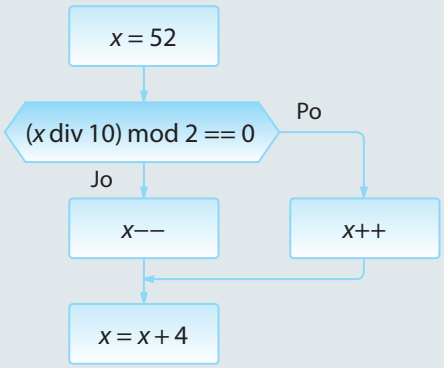


Figura 1.7.

Puno vetë

1. Përpilo skemën algoritmike me ndihmën e së cilës:

- përcaktohet vlera e numrit më të madh (më të vogël) ndër dy numra të dhënë a dhe b;
- për numrin e dhënë a përcaktohet nëse është pozitiv apo negativ. Në qoftë se është pozitiv, rezultati është rrënja e numrit të dhënë, kurse nëse është negativ, rezultati është vlera absolute e tij;
- shënon numrin më të madh nga tre numra të dhënë ose konstatohet se numrat janë të barabartë mes vete;
- radhiten katër numra të dhënë në renditje jorritëse;
- kontrollon nëse skedarët me kapacitete *k1*, *k2*, *k3* dhe *k4* mund të vendosen në dy memorie flash me kapacitete *K1* dhe *K2*;
- për pikën e dhënë (x, y) kontrollon nëse i përket ndonjërës nga drejtëzat e përcaktuara nga pikat $A(x_1, y_1)$, $B(x_2, y_2)$ dhe $C(x_3, y_3)$;
- • për numrat e dhënë x, y dhe z njehson vlerën $\max(z, \min(x, y))$.

1.3.3. Skemat ciklike algoritmike

Skemat ciklike algoritmike janë skemat në të cilat një apo më shumë hapa algoritmikë mund të kryhen më shumë se një herë gjatë kohës së kryerjes së algoritmit. Këta hapa formojnë ***ciklin***, dhe një kalim nëpër cikël quhet ***iteracion (përsëritje)***. Ndërmjet komandave që formojnë strukturën ciklike duhet të ekzistojë së paku një komandë që përkufizon kushtin e daljes nga cikli. Në qoftë se kushti është plotësuar, delet nga cikli, përndryshe cikli përsëritet. Kushti për dalje nga cikli quhet ***kriteri dalës nga cikli***.

Në qoftë se kriteri dalës është numri i përsëritjeve të ciklit, atëherë atë numër e qaujmë ***numëruesin e ciklit***. Në të kundërtën, në qoftë se bëhet fjalë për një kriter tjetër i cili duhet të plotësohet, atëherë flasim për ***ciklin iterativ (përsëritës)***.

1.3.3.1. Ciklet numëruese

Cikli numëruesi i komandës FOR

Më shpesh, në fillim të ciklit numërues, para kryerjes së tij, dihet numri i përsëritjeve (iteracioneve) të tij. Për shembull, kur nevojitet të shënohet mesazhi i njëjtë 100 herë, dihet se kjo mund të realizohet në 100 iteracione, duke shënuar mesazhin një herë në secilin iteracion. Madje edhe nëse numri i iteracioneve varet nga njëri prej parametrave ose ndryshoreve hyrëse, ai numër do të jetë i njohur në momentin kur cikli fillon të kryhet.

Për realizimin e cikleve numëruese përdoret komanda FOR.

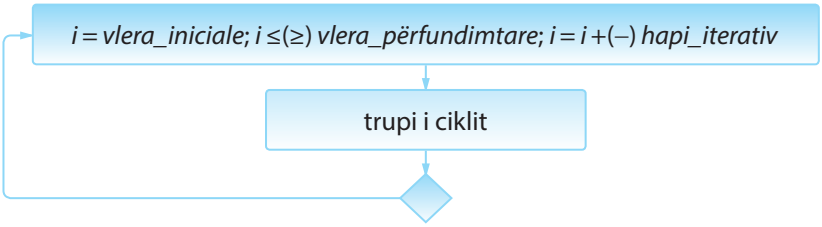


Figura 1.8. Forma e përgjithshme e komandës FOR

Forma e përgjithshme e komandës FOR është paraqitur në figurën 1.8. Me qëllim që gjatë kohës së kryerjes së skemës algoritmike të mundësohet kontrolli i numrit të ekzekutimeve, përkufizohet ndryshorja për kontrollin e ciklit, numëruesi *i*. Numëruesi në fillim inicializohet dhe kështu i shoqërohet **vlera iniciale (fillestare)**. Kushti me të cilin përcaktohet, nëse cikli do të përsëritet, përkufizohet në trajtën ***i <= vlera_përfundimtare*** ose ***i >= vlera_përfundimtare***, me çfarë verifikohet nëse vlera e ndryshores ***i*** ka mbërritur vlerën më të madhe (më të vogël) të paraparë. Shprehja iteracionale e trajtës ***i = i + hapi_iterativ*** ose ***i = i – hapi_iterativ*** përkufizon mënyrën në të cilën ndryshon vlera e numëruesit të ciklit. Këto tre parametra të komandës FOR ndahen me pikëpresje (;). ***Trupi i ciklit*** përbëhet nga një ose më shumë komanda.

Shembulli ilustrues i ciklit numërues është dërgimi i një urimi ditëlindjeje të njëjtë shokut në Facebook, 18 herë për ditëlindjen "jubilare" të 18-të.

Cikli kryhet gjithnjë gjersa kushti është i plotësuar. Në momentin kur kushti logjik bëhet i pasaktë (`false`), dilet nga cikli, kurse kryerja e programit vazhdon me komandën e parë mbas komandës FOR.

Renditja e kryerjes së komandave dhe verifikimi i kushteve në cikle (nga figura 1.7.) është si mëposhtë:

FILLIMI I CIKLIT
- kryhet komanda $i = vlera\_iniciale$

ITERACIONI
- kontrollohet kushti $i \leq (\geq) vlera\_përfundimtare$ (kushti vlen)
- kryhet trupi i ciklit
- kryhet komanda $i = i + (-) hapi\_iterativ$

ITERACIONI
- kontrollohet kushti $i \leq (\geq) vlera\_përfundimtare$ (kushti vlen)
- kryhet trupi i ciklit
- kryhet komanda $i = i + (-) hapi\_iterativ$

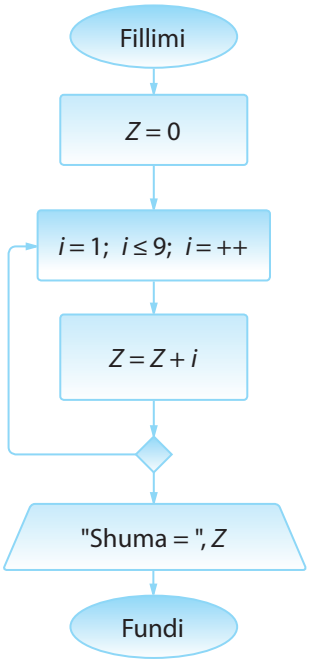
ITERACIONI
- kontrollohet kushti $i \leq (\geq) vlera\_përfundimtare$ (kushti vlen)
- kryhet trupi i ciklit
- kryhet komanda $i = i + (-) hapi\_iterativ$

...

ITERACION I FUNDIT
- kontrollohet kushti $i \leq (\geq) vlera\_përfundimtare$ (kushti NUK vlen – cikli ndërpritet)

FUNDI I CIKLIT.

Algoritmi 14. Të shkruhet skema algoritmike me të cilën njehsohet shuma e të gjithë numrave njëshifrorë.



Ndryshores ***Z*** i shoqërohet vlera fillestare 0 (***kur duhet të bëhet një mbledhje, ndryshores së rezervuar për "mbajtjen në mend" të rezultatit i jepen vlera fillestare 0, sepse 0 është elementi neutral për mbledhje***).

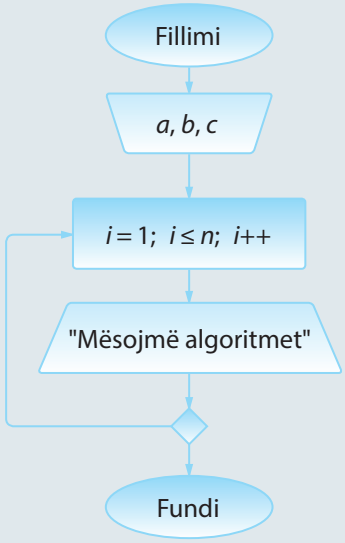
Në kuadrin e komandës FOR përkufizohet numëruesi ***i*** me vlera të mundshme nga 1 deri në 9 (të gjithë numrat njëshifrorë).

Në secilin iteracion (përsëritje) vlera e ndryshores ***Z*** zmadhohet për vlerën ***i***.

Rezultati i kërkuar është vlera e ndryshores ***Z***.

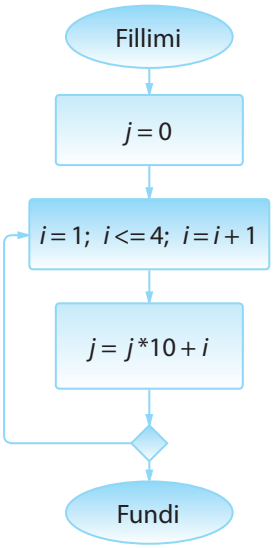
Të kemi kujdes se numri i iteracioneve të kryera është 4 ($= vlera\_nga\_kushti\_i\_ciklit - vlera\_iniciale + 1$). Gjatë shënimit të këtyre skemave algoritmike, shpeshherë gabohet, sepse cikli përsëritet një herë më shumë sesa nevojitet, prandaj duhet të kihet kujdes në përkufizimin e vlerës fillestare të numëruesit dhe kushtit për dalje nga cikli FOR.

Algoritmi 15.
Shkruani skemën algoritmike e cila e paraqet n herë mesazhin "Mësojmë algoritmet", ku $n > 0$ është vlera e cila jepet në hyrje.



Shembulli 25. Të përcaktohet vlera e madhësisë dalëse në skemën algoritmike të paraqitur në figurën e mëposhtme.

Renditjen e kryerjes së komandave në ciklin FOR do ta paraqesim hap nga një hap.



Zgjidhje:
Me komandën e parë vlera **0** i shoqërohet ndryshores *j*.

Fillimi
i ciklit: $i = \mathbf{1}$ ✓

Iteracioni i 1. $j = \mathbf{0}\cdot 10 + \mathbf{1} = \mathbf{1}$ $i = \mathbf{2}$ $i \le 4$ ✓

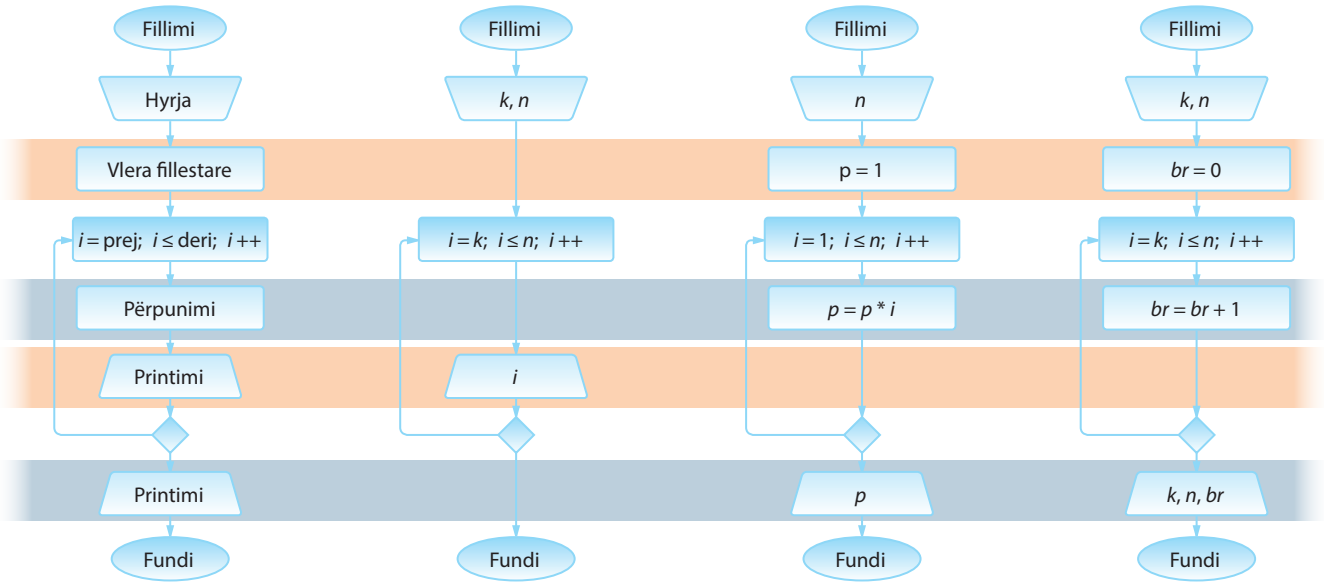
Iteracioni i 2. $j = \mathbf{1}\cdot 10 + \mathbf{2} = \mathbf{12}$ $i = \mathbf{3}$ $i \le 4$ ✓

Iteracioni i 3. $j = \mathbf{12}\cdot 10 + \mathbf{3} = \mathbf{123}$ $i = \mathbf{4}$ $i \le 4$ ✓

Iteracioni i 4. $j = \mathbf{123}\cdot 10 + \mathbf{4} = \mathbf{1234}$ $i = \mathbf{5}$ $i \le 4$ ×

Sipas analogjisë me shembullin e dhënë, mund të përpilojmë një varg të tërë skemash algoritmike në të cilat janë të ndryshme vlerat fillestare, përpunimet dhe shënimi i të dhënave, siç është paraqitur në figurën e mëposhtme.

Diagrami i përgjithshëm	Shënimi i numrave prej *k* deri në *n*	Prodhimi prej 1 deri në *n* (*n*!)	Numëruesi prej *k* deri në *n*



Shembulli 26.
Shkruani skemën algoritmike që për numrin e dhënë n njehson shumën:

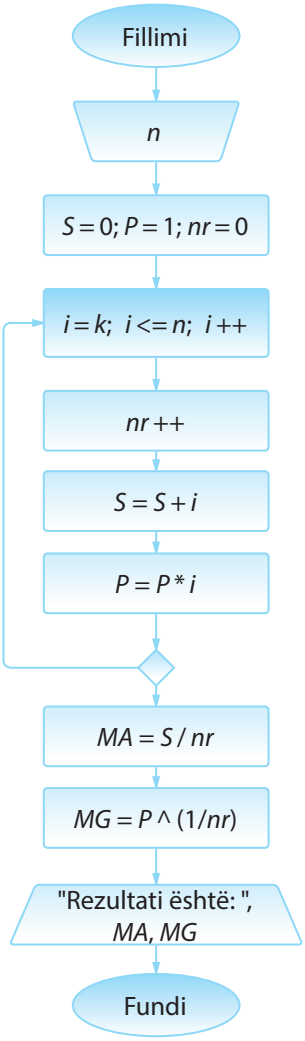
$S = 1! + 2! + \ldots + n!$

Shembulli 27.
Shkruani skemën algoritmike që përcakton vlerën mesatare të të gjithë numrave të plotë nga intervali i dhënë (a, b).

Shembulli 28.
Shkruani skemën algoritmike që për numrin e dhënë k, njehson prodhimin:

$P = k(k + 1)\ldots(2k - 1)2k.$

Algoritmi 16. Të shkruhet skema algoritmike për njehsimin e mesit aritmetik dhe gjeometrik të numrave të plotë nga intervali $[k, n]$, $k < n$.



Ndryshoret ndihmëse
S – shuma e numrave (vlera fillestare 0),
P – prodhimi i numrave (vlera fillestare 1),
nr – numri i përgjithshëm i numrave të plotë nga intervali $[k, n]$ (mund të njehsohet sipas formulës $nr = n - k + 1$).
Në ciklin FOR numëruesi i merr vlerën nga intervali $[k, n]$. Në secilin iteracion vlerat e ndryshoreve ndihmëse *br*, *S* dhe *P rinovohen*.

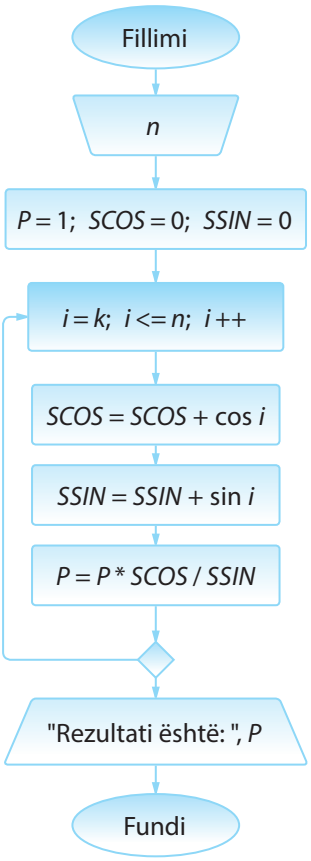
Mbas përfundimit të ciklit FOR, ndryshoret *MA* dhe *MG* marrin vlerat e meseve aritmetike dhe gjeometrike, sipas formulave:

$$MA = \frac{S}{nr}, \quad MG = \sqrt[nr]{P}$$

dhe shkruhet rezultati.

Algoritmi 17. Të shkruhet skema algoritmike që për numrin e dhënë natyror n e njehson shprehjen:

$$P=\frac{\cos 1}{\sin 1}\cdot\frac{\cos 1+\cos 2}{\sin 1+\sin 2}\cdot\ldots\cdot\frac{\cos 1+\ldots+\cos n}{\sin 1+\ldots+\sin n}.$$



Ndryshoret ndihmëse:
P – vlera e shprehjes së kërkuar (vlera fillestare 1),
SCOS dhe *SSIN* – shumat në numërues gjegjësisht në emërues të kufizave të veçanta të shprehjes (vlera fillestare 0).

Në ciklin FOR numëruesi *i* merr vlerën nga *1* deri në *n*, dhe mbas çdo iteracioni, rinovohen vlerat e ndryshoreve *SCOS*, *SSIN* dhe *P*.

Le të vëmë re vetitë e mëposhtme të ndryshoreve të përkufizuara:
Nëse me P_k, $SSIN_k$ i $SCOS_k$ shënojmë vlerat në iteracionin e *k*-të vlerat në iteracionin e, *k*+1 pasues janë dhënë sipas formulave:

$$SCOS_{k+1}=SCOS_k+\cos(k+1)$$
$$SSIN_{k+1}=SSIN_k+\sin(k+1)$$
$$P_{k+1}=P_k\cdot\frac{SCOS_{k+1}}{SSIN_{k+1}}.$$

Shembulli 29.
Shkruani skemën algoritmike që për numrin e dhënë *n* njehson shumën:

$$S=\frac{1!}{n}+\frac{2!}{n-1}+\frac{3!}{n-2}+\ldots+\frac{n!}{1}$$

Shembulli 30.
Shkruani skemën algoritmike me të cilën njehsohet vlera e shprehjes:

$$S=\sqrt{2+\sqrt{2+\ldots+\sqrt{2+\sqrt{2}}}}$$

ku rrënja zbatohet *n* herë.

Skemat algoritmike mund të përmbajnë një numër më të madh ciklesh FOR ashtu që ato mund të renditen njëra mbas tjetrës (figura 1.9. a.) ose të ndodhen njëra brenda tjetrës (figura 1.9. c).

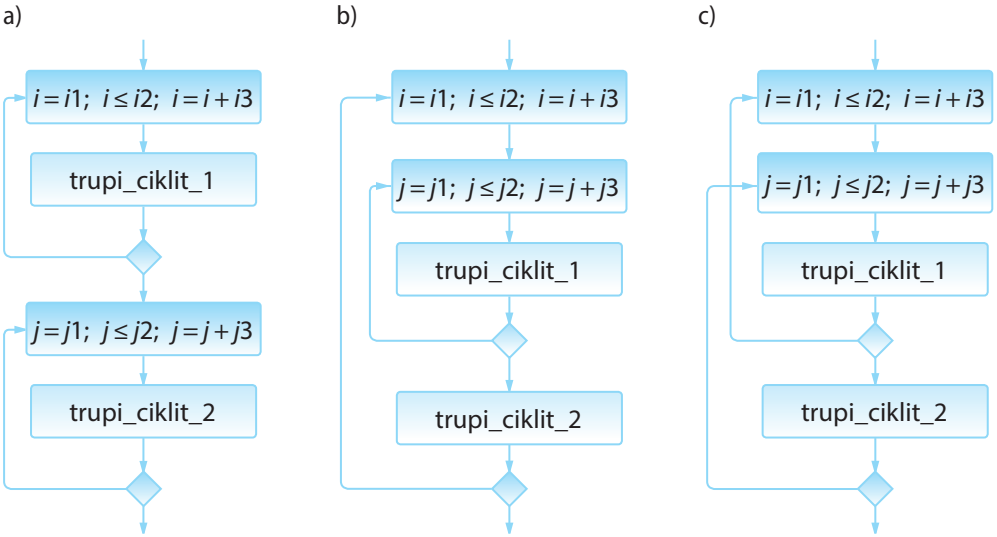
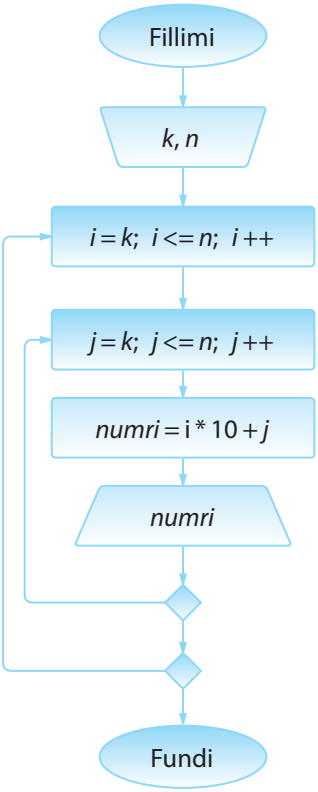


Figura 1.9. Kombinimet e lejueshme dhe të palejueshme të cikleve FOR.

Algoritmi 18. Të shkruhet skema algoritmike nëpërmjet të cilës për numrat e dhënë njëshifrorë k dhe n, $k<n$ paraqet të gjithë numrat dyshifrorë që mund të krijohen nga shifrat nga segmenti $[k, n]$. (P.sh. për $k=2$, $n=4$, të gjithë numrat dyshifrorë të formuar nga shifrat 2, 3 dhe 4 janë 22, 23, 24, 32, 33, 34, 42, 43, 44.)



Duke pasur parasysh se në pozicion të shifrës së parë dhe të dytë mund të jetë secili numër nga bashkësia e dhënë, na duhen dy cikle FOR.

Meqenëse për një vlerë të shifrës së parë (shiko në shembullin paraprak numrat që fillojnë me shifrën 2), shifra e dytë mund të marrë të gjitha vlerat nga segmenti $[k, n]$ (shiko në shembullin, vlerat e mundshme të shifrës së dytë janë 2, 3, 4, prandaj formohen numrat dyshifrorë 22, 23, 24), cikli që kontrollon vlerën e shifrës së dytë do të ndodhet brenda ciklit për kontrollimin e shifrës së parë.

Prandaj, përkufizojmë ciklin e "jashtëm" (me numëruesin ***i***), që na paraqet shifrën e dhjetësheve të numrit që formohet dhe ciklin e "brendshëm" (me numërues ***j***), që paraqet shifrën e njësheve. Gjatë secilit iteracion formojmë numrin dyshifror $\overline{ij}$ dhe paraqesim vlerën e tij.

1.3.3.2. Cikli iterativ

Për realizimin e ciklit iterativ përdoren komandat **WHILE** dhe **DO-WHILE**. Në këto komanda (cikle) nuk është paraprakisht e njohur sa iteracione do të kryhen. Prandaj nuk ka kuptim që të përdoret numëruesi i ciklit, sepse kushti për dalje nga cikli nuk varet nga vlera e numëruesit, por nga një kusht tjetër. Në bazë të paraqitjes ilustruese të të dy cikleve (të dhënë në figurën 1.10), mund të vihet re se dallimi kryesor midis tyre është në pozitat për kontrollin e kushtit të daljes nga cikli, çfarë do të sqarohet në vazhdim.

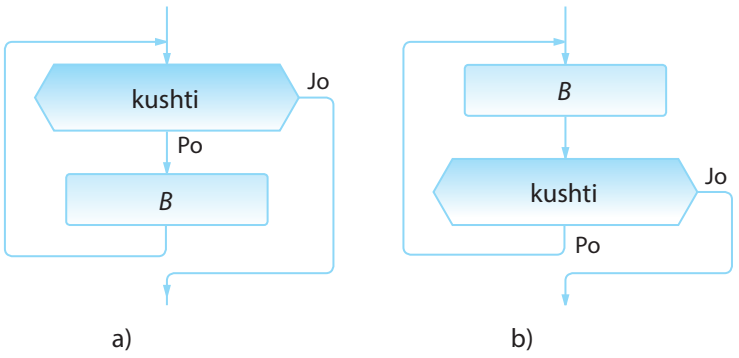


Figura 1.10. Forma e përgjithshme e ciklit a) WHILE, b) DO-WHILE

Forma e përgjithshme e ciklit **WHILE** është dhënë në figurën 1.10.a. Në këtë cikël, B është një apo më shumë komanda, gjersa kushti përkufizon kriterin me të cilin kontrollohet cikli dhe mund të jetë cilado shprehje e saktë logjike. Cikli përsëritet derisa kushti është plotësuar. Kur kushti nuk vlen (nuk plotësohet), dilet nga cikli dhe kalohet në komandën e parë mbas ciklit.

Renditja e kryerjes së komandave dhe kontrolli i kushtit në ciklit WHILE nga figura 1.10. a) është siç vijon:

FILLIMI I CIKLIT

ITERACIONI 1
- kontrollohet *kushti* (kushti i plotësuar)
- kryhet komanda *B*

ITERACIONI 2
- kontrollohet *kushti* (kushti i plotësuar)
- kryhet komanda *B*

ITERACIONI 3
- kontrollohet *kushti* (kushti i plotësuar)
- kryhet komanda *B*

...

ITERACIONI I FUNDIT
- kontrollohet *kushti* (kushti nuk është plotësuar – dilet nga cikli)

FUNDI I CIKLIT.

Mund të ndodhë që kushti nuk është plotësuar në fillim të ciklit. Në atë rast **nuk do të kryhet asnjë iteracion**.

Mirëpo, gjithashtu mund të ndodhë, që si rezultat gabimi, të shënohet kushti që plotësohet gjithmonë, pavarësisht nga vlerat e ndryshoreve. Efekti i një situate të tillë është që cikli WHILE do të përsëritet pandërprerë dhe do të bëhet **"cikli i pafundmë"**.

Mënyra e dytë e realizimit të ciklit iterativ është zbatimi i strukturës **DO-WHILE**, forma e përgjithshme e të cilit është paraqitur në figurën 1.10.b). Në këtë cikël, kushti kontrollohet në fund, që nënkupton se cikli duhet të kryhet së paku njëherë dhe kryhet gjithnjë derisa kushti është i plotësuar.

Vëmë re se cikli numërues (d.m.th. cikli **FOR**), mund të realizohet si cikël iterativ (p.sh. me ndihmën e ciklit **WHILE**) (figura 1.11.).

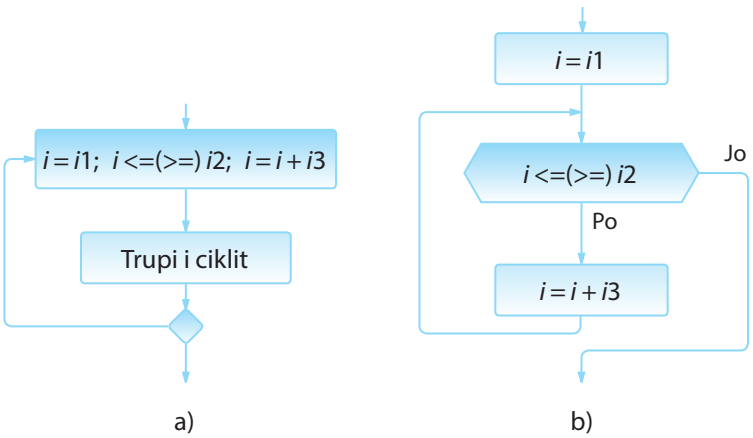


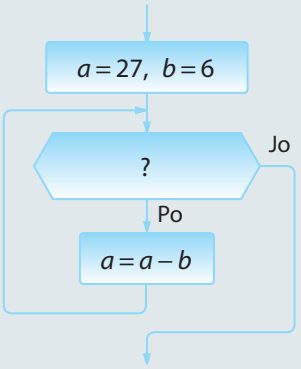
Figura 1.11. Paraqitja e ciklit numërues në formën e ciklit iterativ.

Shembull ilustrues i skemës ciklike algoritmike është përgjigjja në mesazhe nga inbox-i. Veprimi përsëritet gjithnjë derisa ka mesazhe të palexuara.

Shembulli 31.

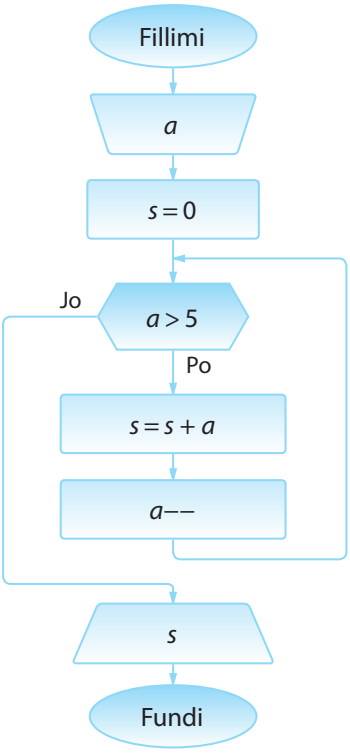
Cili nga kushtet e ofruara nevojitet në ciklin WHILE në mënyrë që ai të kryhet saktësisht 5 herë? Rrumbullako numrin pranë përgjigjes së saktë.

1. $b < 5$ 2. $a! = b$
3. $b < a$ 4. $a > 5$

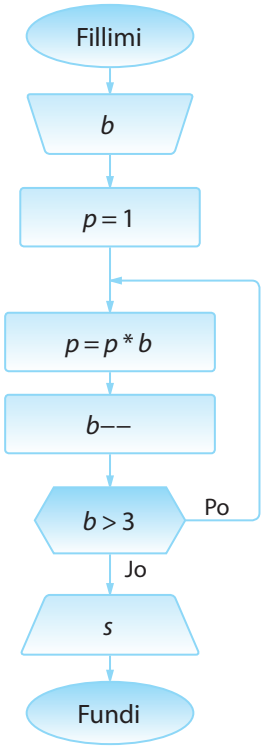


Shembulli 32. Të përcaktohen vlerat dalëse të skemave algoritmike të paraqitura në figurat për vlerat e parametrave hyrës $a = 9$ dhe $b = 6$.

Që të përgjigjemi në detyrën paraprake, do të tregojmë hap nga një hap, kryerjen e cikleve të paraqitura.

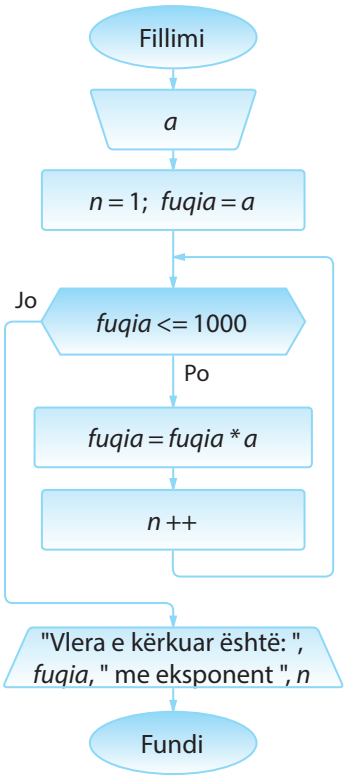


Fillimi i ciklit:	$a = 9$	$s = 0$	$a > 5$ ✓
Iteracioni 1:	$a = 8$	$s = 9$	$a > 5$ ✓
Iteracioni 2:	$a = 7$	$s = 17$	$a > 5$ ✓
Iteracioni 3:	$a = 6$	$s = 24$	$a > 5$ ✓
Iteracioni 4:	$a = 5$	$s = 30$	$a > 5$ ×
Rezultati:		$s = 30$	



Fillimi i ciklit:	$b = 6$	$p = 1$	
Iteracioni 1:	$b = 5$	$p = 0$	$b > 3$ ✓
Iteracioni 2:	$b = 4$	$p = -1$	$b > 3$ ✓
Iteracioni 3:	$b = 3$	$p = -2$	$b > 3$ ×
Rezultati:		$p = -3$	

Algoritmi 19. Të shkruhet skema algoritmike e cila për numrin e plotë a përcakton numrin më të vogël të trajtës a^n që është më i madh sesa numri 1000 dhe paraqet madhësitë dalëse n dhe a^n. Ky algoritëm, në të vërtetë duhet të shumëzojë numrin a me vetvete gjithnjë derisa të përftohet një numër më i madh sesa numri 1000.



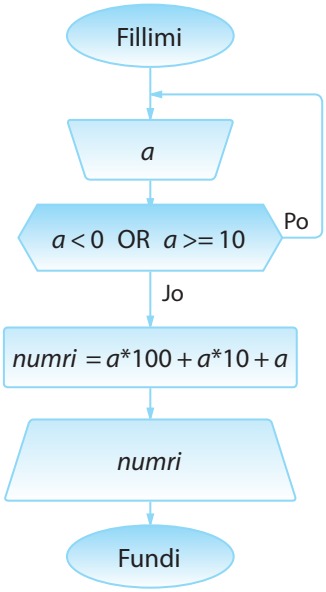
Hyrja e numrit të plotë a.

Ndryshorja n paraqet eksponentin, kurse *fuqiaA* vlerën momentale të shprehjes a^n.

Gjigjnjë derisa vlera e shprehjes *fuqia* ≤ 1000, zmadhohet eksponenti (n++) dhe njehsohet vlera e *fuqia* (*fuqia* = *fuqia* * a).

Shënimi i rezultatit.

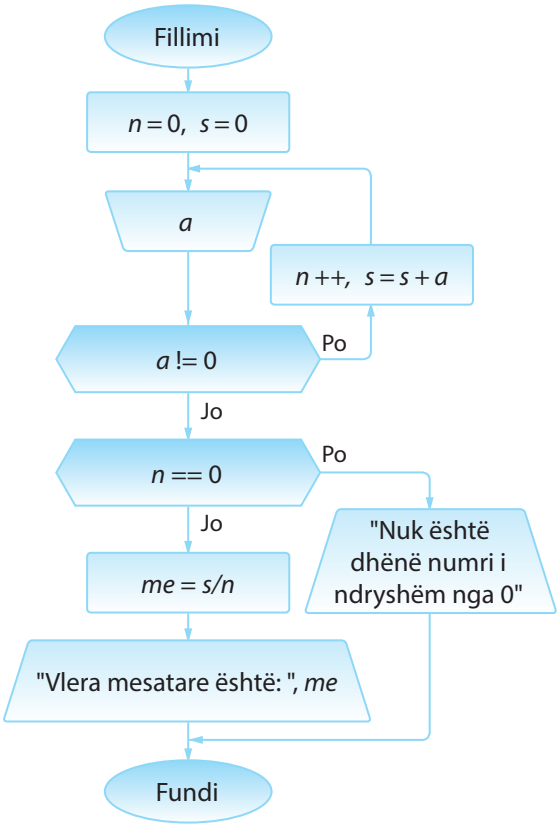
Algoritmi 20. Të shkruhet skema algoritmike nëpërmjet të cilës lexohet numri njëshifror dhe paraqitet numri treshifror i formuar vetëm me ndihmën e asaj shifre. Në qoftë se numri që paraqitet në hyrje është 0 ose numër jo njëshifror, duhet të jepet vlera e re dhe të kontrollohet, nëse ajo është korrekte apo jo. Kontrolli dhe hyrja e re bëhet deri te hyrja e parë korrekte.



Mbas hyrjes së numrit a kryhet kontrolli nëse është 0 apo numër shumëshifror. Në rast se po, përsëritet hyrja dhe kontrolli.

Mbas hyrjes së parë korrekte, njehsohet numri $x = \overline{aaa}$ dhe shënohet rezultati

Algoritmi 21. Të shkruhet skema algoritmike që llogarit vlerën mesatare të numrave të ndryshëm nga zeroja, numri i të cilëve është i panjohur. Duhet të mundësohet hyrja e numrave, dhe si fund hyrjeje të përdoret numri 0.



Ndryshoret ndihmëse:
n – numri i përgjithshëm i numrave të dhënë,
s – shuma e numrave të dhënë.
Vlera fillestare e të dy ndryshoreve është 0.

Mbas hyrjes së numrit a, kryhet kontrolli nëse është numër i ndryshëm nga zeroja. Në qoftë se po, rinovohen vlerat e ndryshoreve n dhe s.
Në të kundërtën, vlera mesatare e numrave të dhënë njehsohet sipas formulës $me = s/n$ dhe shënohet rezultati.

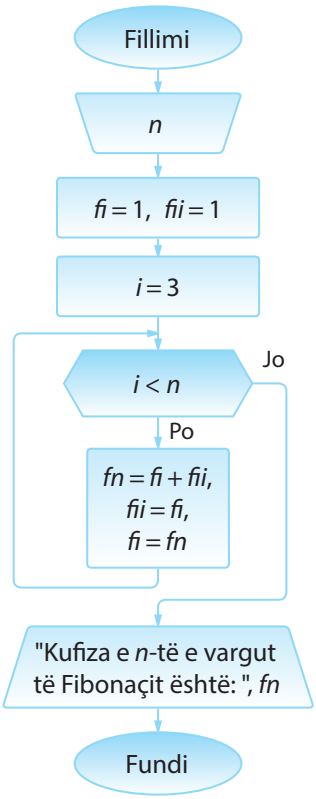
Pyetje: *A mund të zgjidhet detyra me ndihmën e komandave WHILE dhe DO-WHILE?*

Shembulli 33.
Shkruani skemën algoritmike me të cilën ndër numrat ...

0.5, 0.5 * 1.5, 0.5 * 1.5 * 2.5 ...

kërkohet numri i parë më i madh sesa numri i dhëna *a*.

Algoritmi 22. Të shkruhet skema algoritmike me të cilën njehsohet kufiza e *n*-ti të e vargut të Fibonaçit: $f_1 = 1,\ f_2 = 1,\ f_i = f_{i-1} + f_{i-2},\ i = 3, 4, 5...$



Ndryshoret ndihmëse: *fi* i *fii*, paraqesin numrin e parë dhe numrin e dytë të Fibonaçit. Vlerat fillestare i kanë të barabarta me 1. Më vonë në iteracionet e ciklit *WHILE*, ndryshoret e njëjta përdoren për paraqitjen, sipas radhës, të numrit të *i* -të dhe numrit (kufizës) të $i + 1$-të të Fibonaçit.

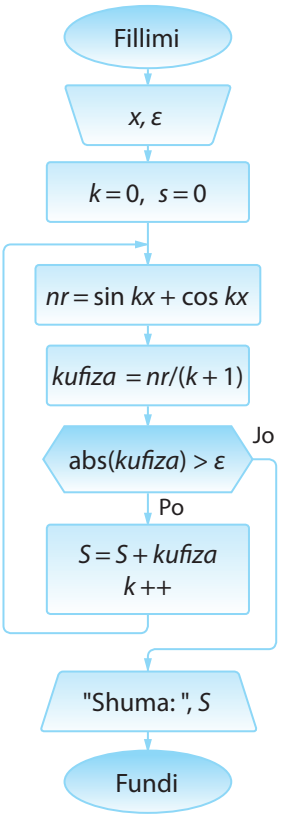
Ndryshorja ndihmëse *i* tregon se cilin nga numrat e Fibonaçit sipas radhës duhet ta njehsojmë. Meqenëse dy kufizat e para janë përkufizuar paraprakisht, vlera fillestare është 3.

Brenda ciklit, vlera e numrit të Fibonaçit që njehsohet në atë moment (numri i i-të i Fibonaçit) i shoqërohet ndryshores fn, pastaj rinovohen vlerat e ndryshoreve *fi* dhe *fii* me qëllim që ato të paraqesin vlerën e kufizës së $i - 1$-të (d.m.th. vlerën e fn) dhe e kufizës së *i*-të (d.m.th. vlerën e *fi*) të Fibonaçit në iteracionin pasues.

Algoritmi 23. Të shkruhet skema algoritmike me të cilën njehsohet shuma

$$\sum_{k=0}^{\infty} \frac{\sin x + \cos x}{k+1}$$

për vlerat e dhëna të x dhe saktësinë ε. Mbledhja përfundon kur kufiza e fundit është më e vogël se $\leq \varepsilon$ sipas vlerës absolute.



Ndryshoret ndihmëse:
k – numëruesi në shumën që duhet të njehsohet,
S – vlera e shumës dhe
nr – numëruesi i kufizave të shumës.

Vlera e mbledhësit të *k*-të i shoqërohet ndryshores kufiza, kurse kushti për përfundimin e llogaritjes është $|kufiza| \leq \varepsilon$.

Në çdo iteracion bëhet rinovimi i ndryshoreve *S* dhe *k*.

Pyetje dhe detyra kontrolli

1. Cilat lloje të skemave ciklike algoritmike ekzistojnë dhe cili është dallimi midis tyre?

2. Cilat nga pohimet e mëposhtme janë të sakta:

a) Cikli numërues mund të paraqitet nëpërmjet ciklit iterativ.

b) Cikli iterativ mund të paraqitet nëpërmjet ciklit numërues.

c) Ekzistojnë disa shembuj me cikle numëruese që mund të paraqiten nëpërmjet ciklit iterativ, mirëpo një gjë e tillë nuk vlen në përgjithësi.

3. Sa herë kryhen ciklet e paraqitura në figurën 1.12?

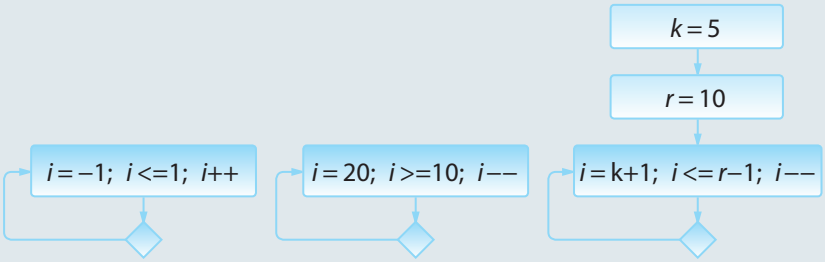


Figura 1.12.

4. Cila është vlera e ndryshores *s* në dalje nga skema algoritmike nga figura 1.13?

a) 30

b) 31

c) 21

d) 20

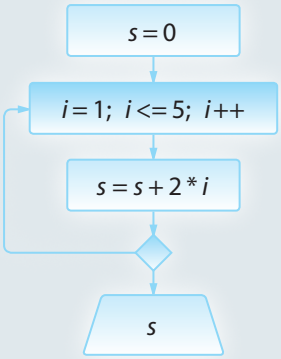


Figura 1.13.

5. Cila është vlera e ndryshores *s* në dalje nga skema algoritmike nga figura 1.14?

a) 10

b) 5

c) 16

d) 15

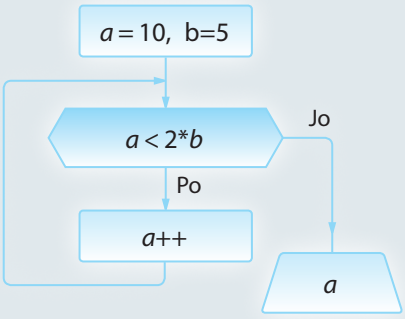


Figura 1.14.

6. Sa herë do të kryhet cikli nga figura 1.15? Cila është vlera e ndryshores *s* në dalje nga skema algoritmike?

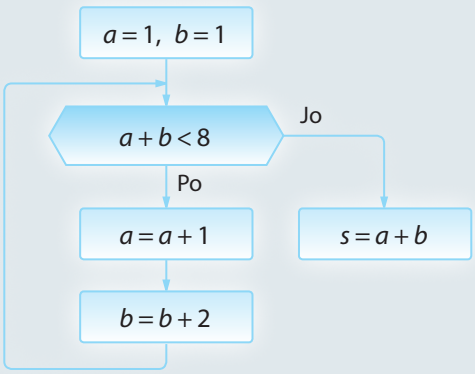


Figura 1.15.

Puno vetë

1. Shkruaje skemën algoritmike nëpërmjet të cilës:

- Për n numra të dhënë (n është dhënë), përcaktohet shuma e tyre;
- Njehsohet shuma $\sum_{k=1}^{n} \frac{1}{k}$, për numrin n të dhënë paraprakisht;
- Njehsohet shuma $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{100^2}$;
- Për numrin e dhënë të plotë përcaktohet:
  a) Shifra më e madhe e tij,
  b) Numri më i afërt që plotpjesëtohet me 5, por jo me 7,
  c) Numri që përftohet duke futur shifrën c në pozicionin k,
  d) Numri që përftohet duke hequr shifrën nga pozicioni i k-të i numrit;
- Përcaktohet vlera e shumës

$$S = \frac{1}{n+m} - \frac{1}{n+2m} + \frac{1}{n+3m} - \ldots + (-1)^{m+1} \frac{1}{n+m \cdot m}$$

për numrat e dhënë natyrorë m dhe n.

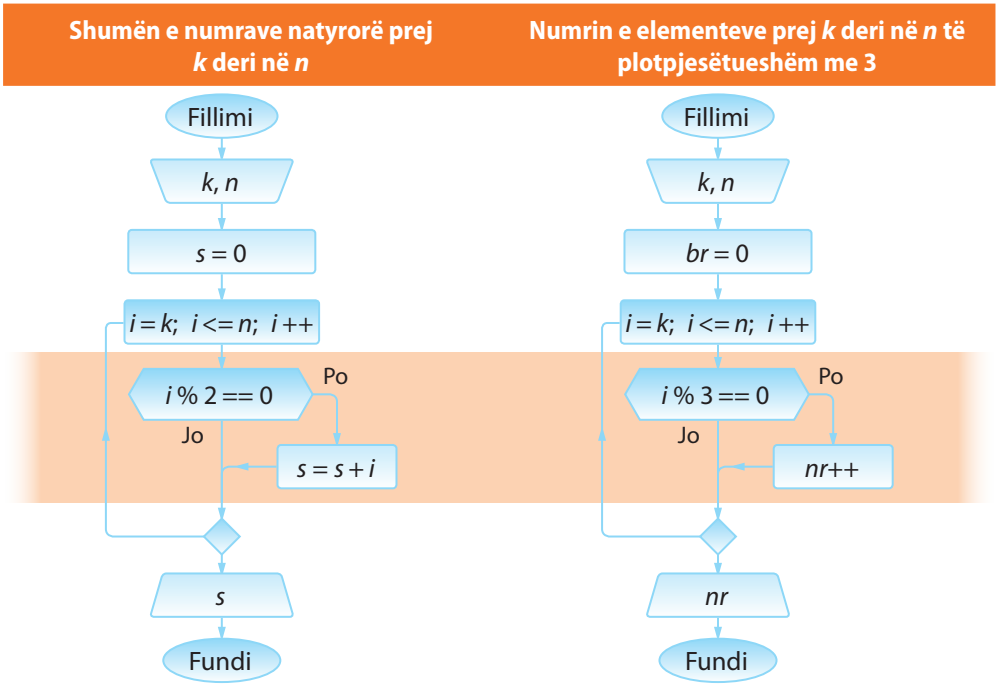
2. Shkruani skemën algoritmike e cila si parametër hyrës ka madhësinë e kamatës së shprehur në përqindje (p.sh. 9.5 %), të cilën banka e jep në nivelin vjetor. Duhet të njehsohet se për sa vite, mjetet e investuara do të dyfishohen, d.m.th. mbas sa vitesh investimi do të rritet në më shumë se 200 %. Hipoteza kryesore është që mjetet nga viti paraprak, së bashku me kamatën për atë vit do të investohen në vitin pasues, ashtu që do të llogaritet "kamata në kamatë".

1.3.4. Skemat komplekse algoritmike

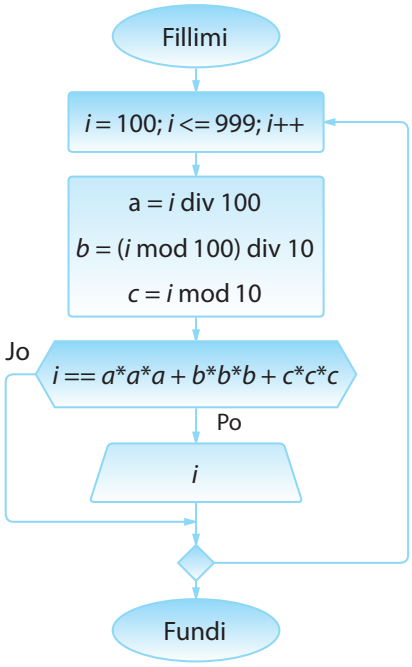
Jemi njohur me mënyrat për paraqitjen e sekuencave, degëzimeve dhe përsëritjeve në skemat algoritmike, si edhe me shembujt që kanë përmbajtur vetëm njërën prej tyre. Me kombinacione të ndryshme të këtyre elementeve përftohen ***skemat algoritmike komplekse***.

Në vazhdim jepen disa shembuj të zbatimit të tyre.

Algoritmi 24. Me kombinimin e ciklit FOR me degëzim të kushtëzuar, mund të formohen skemat algoritmike për:



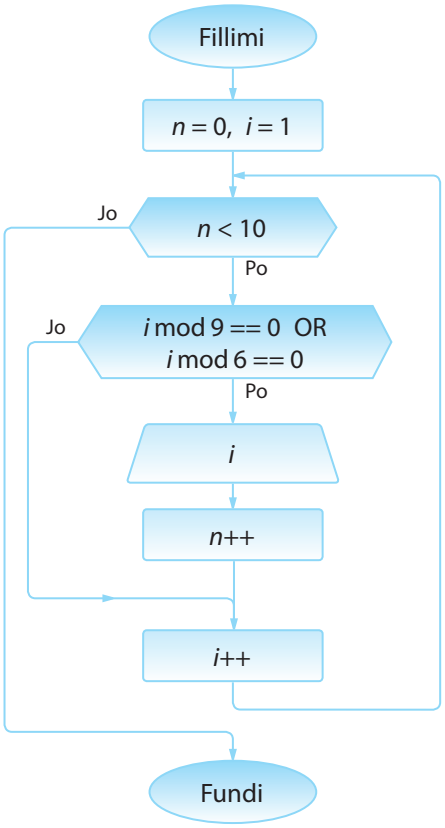
Algoritmi 25. Të shkruhet skema algoritmike me të cilën shënohen të gjithën numrat e Armstrongut. (Numri i takon Armstrongut në qoftë se është i barabartë me shumën e kubeve të shifrave të veta, p.sh. 407 = 4*4*4 + 0*0*0 + 7*7*7).



Nëpërmjet ciklit FOR për çdo numër treshifror $\overline{abc}$ verifikohet nëse është numri i Armstrongut, me ndihmën e këtyre hapave.

- Përcaktohen sipas radhës vlerat e shifrave *a*, *b*, *c*;
- Kontrollohet kushti se a është numri i dhënë (numëruesi i ciklit) i barabartë me shumën e kubeve të shifrave të tij;
- Në qoftë se pohimi paraprak është i saktë, numri i takon Armstrongut dhe shënohet.

Algoritmi 26. Të shkruhet skema algoritmike për përcaktimin e dhjetë numrave të parë të plotë më të mëdhenj se zeroja që plotpjesëtohen me 9 ose me 6.



Ndryshoret ndihmëse:

n – numri i numrave të gjetur që janë të plotpjesëtueshëm me 9 ose me 6 (vlera fillestare – 0, asnjë numër që plotëson kushtet e përkufizuara nuk është gjetur në fillim),

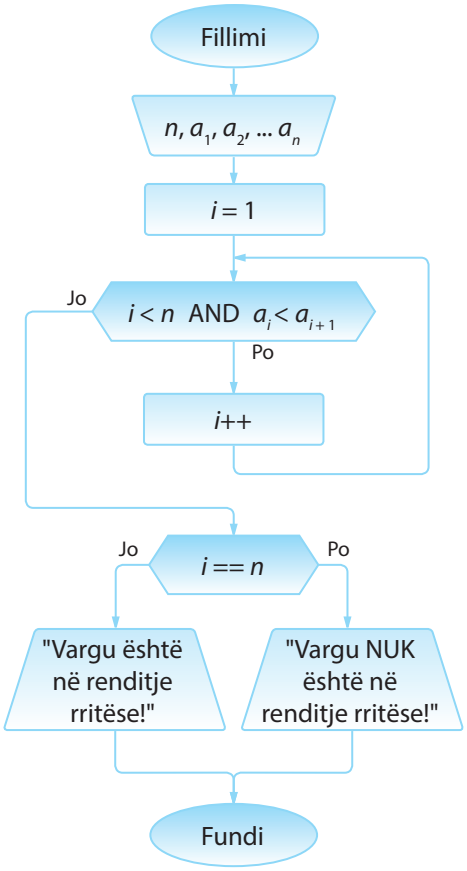
i – numri momental për të cilin bëhet kontrolli a i plotëson kushtet e përkufizuara (vlera fillestare - 1 numri më i vogël natyror).

Gjithnjë derisa ka më pak se 10 numra të gjetur, që plotësojnë kushtin e detyrës, përsëriten hapat:

- Për numrin e plotë momental i kontrollohet se a e plotëson kushtin. Në qoftë se e plotëson, paraqitet vlera e tij, kurse numëruesi ***n*** zmadhohet, sepse është gjetur edhe një numër që plotëson kushtin e përcaktuar.
- Kalohet në numrin tjetër për të cilin duhet të verifikohet, nëse e plotëson kushtin e përcaktuar.

Cikli përfundon kur gjinden 10 numra natyrorë që plotësojnë kushtin e përcaktuar.

Algoritmi 27. Të shkruhet skema algoritmike nëpërmjet të cilës për bashkësinë e dhënë të n numrave kontrollon, nëse janë të radhitur sipas renditjes rritëse.



Hyrja e n numrave $a_1, a_2, \ldots a_n$.

Ndryshorja ndihmëse ***i*** paraqet indeksin e numrit që kontrollohet (vlera fillestare është 1).

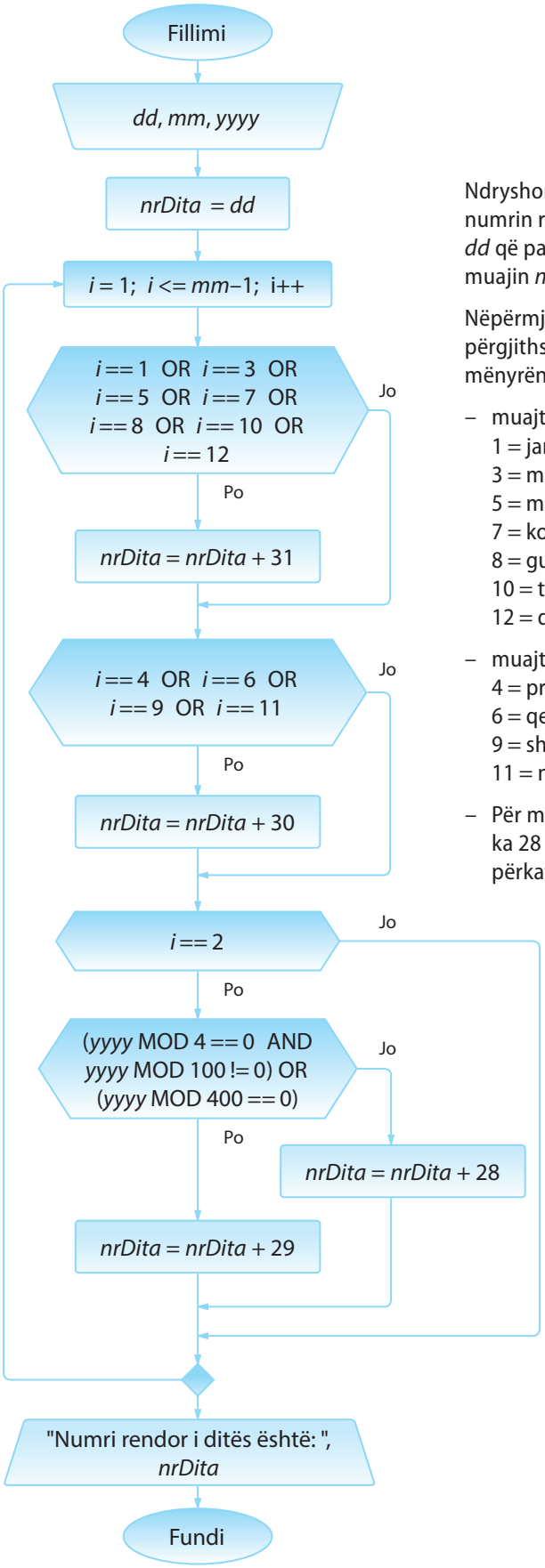
Për secilin numër kontrollohet a është më i vogël sesa pasardhësi ($a_i < a_i+1$) dhe a kemi mbërri deri në fund ($i = n$ tregon për numrin e fundit të futur).

Në qoftë se së paku njëri nga kushtet nuk është plotësuar, dilet nga cikli.

Domethënë, në dalje nga cikli vlen njëri prej këtyre dy kushteve:

- $i == n$, është arritur deri te numri i fundit i futur dhe njëkohësisht renditja rritëse nuk është prishur. Domethënë, numrat e futur JANË në renditjen rritëse!
- $i < n$, para se është kontrolluar vargu i tërë, është gjetur numri që nuk është më i vogël sesa pasuesi i tij (në daljen nga cikli paraprak), që nënkupton se vargu NUK është në renditjen rritëse!

Algoritmi 28. Të shkruhet skema algoritmike, argumenti hyrës i të cilës është data (në formën dd, mm, yyyy) dhe përcakton numrin rendor të asaj dite në vitin përkatës (në atë vit). Të kihet kujdes mbi vitet e brishta, kur shkurti ka 29 ditë. Vitet e brishta janë të gjitha ato vite që plotpjesëtohen me 4 por jo me 100, ose plotpjesëtohen me 400.



Ndryshorja ndihmëse ***nrDita*** – paraqet numrin rendor të ditëve (vlera fillestare është *dd* që paraqet numrin rendor të ditës në muajin *mm*).

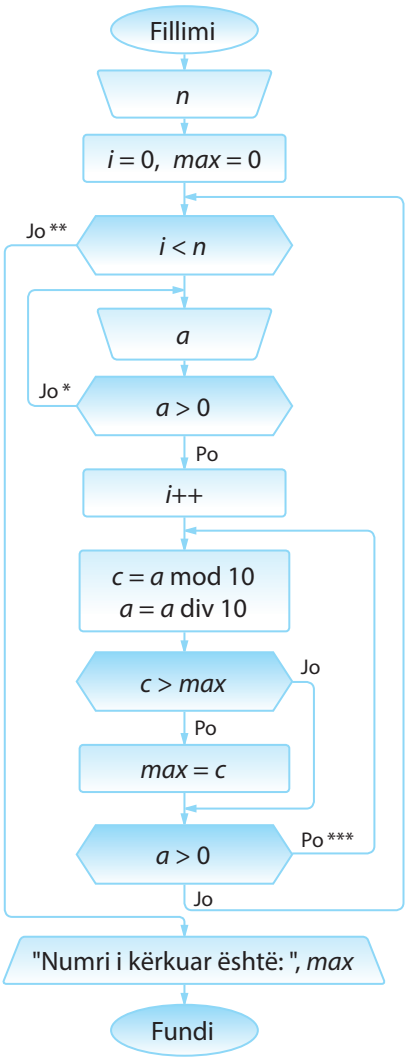
Nëpërmjet ciklit FOR njehsohet numri i përgjithshëm i ditëve deri te muaji *mm* në mënyrën e mëposhtme:

- muajt
  1 = janari,
  3 = marsi,
  5 = maji,
  7 = korriku,
  8 = gushti,
  10 = tetori,
  12 = dhjetori, kanë nga 31 ditë;
- muajt
  4 = prilli,
  6 = qershori,
  9 = shtatori,
  11 = nëntori, kanë nga 30 ditë;
- Për muajin 2 = shkurtin kontrollohet nëse ka 28 ose 29 ditë, varësisht se është viti përkatës vit i brishtë apo jo.

Shembulli 34.
Shkruaje skemën algoritmike me të cilën vargu i dhënë me gjatësi n kontrollohet se a është renditur sipas rregullit të mëposhtëm:

$$a_1 < a_2 > a_3 < a_4 > a_5.$$

Algoritmi 29. Të shkruhet skema algoritmike, nëpërmjet të cilës për numrin e dhënë n ($n > 0$) futen n numra natyrorë dhe përcaktohet shifra më e madhe të cilën ata numra e përbëjnë (p.sh. midis numrave 792, 156, 223 – shifra më e madhe është 9).



Ndryshoret ndihmëse:

i – që paraqet numrin aktual të numrave natyrorë të futur, vlera fillestare është 0;

max – shifra aktuale më e madhe ndërmjet i numrave të futur edhe të analizuar, vlera fillestare është 0;

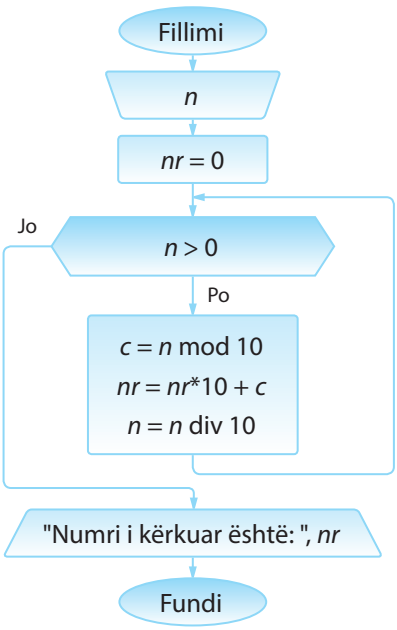
a – numri natyror hyrës.

Çdo numër hyrës zbërthehet në shifra (cikli ***) dhe shifra aktuale i shoqërohet ndryshores c.

Më tutje kontrollohet nëse ajo është më e madhe nga shifra më e madhe aktuale (vlera e ndryshores max). Në rast se po, vlera ndryshores c i shoqërohet ndryshores max.

Veprimi përsëritet derisa të hyjnë të gjithë n numrat natyrorë (cikli **). Para secilës hyrje, bëhet kontrolli i korrektësisë (cikli **). Në fund, ndryshores max i shoqërohet vlera e shifrës më të madhe ndër n numrat e futur.

Algoritmi 30. Të shkruhet skema algoritmike nëpërmjet të cilës lexohet numri n dhe përcaktohet numri i formuluar nga të njëjtat shifra, mirëpo në renditjen e anasjellë (p.sh. për $n = 12345$, numri i kërkuar është 54321).



Mbas hyrjes së numrit n bëhet zbërthimi shifër për shifër i numrit të dhënë, filluar nga shifra e njësheve. Ndryshores ndihmëse nr i shoqërohet numri aktual i krijuar, i shënuar nëpërmjet shifrave të përftuara.

Shifra e fundit (shifra e njësheve) e numrit n përftohet si mbetje e pjesëtimit me 10 dhe veprimi vazhdon derisa të përcaktohen të gjitha shifrat e numrit të dhënë, siç është paraqitur në shembullin e mëposhtëm:

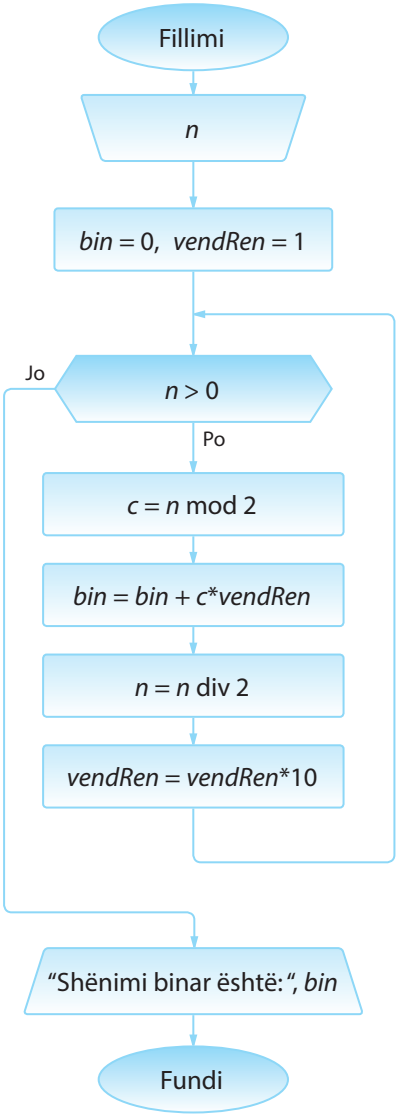
n	nr	c
12345	**0**	
1234	**5**	5
123	**54**	4
12	**543**	3
1	**5432**	2
0	**54321**	1

Në fund rezultati i kërkuar është vlera e ndryshores nr.

Duke përdorur analogjinë nga algoritmi i krijuar paraprakisht, mund të shkruajmë algoritmin për konvertimin e numrit nga sistemi dhjetor në sistemin binar. Për shkak të rëndësisë së sistemit binar numerik në shkenca kompjuteristike, jepet zgjidhja e plotë e kësaj detyre.

Algoritmi 31. Të shkruhet skema algoritmike nëpërmjet së cilës numri i dhënë në sistemin dhjetor zbërthehet në sistemin binar.

P.sh. $13 : 2 = 6$ (*mbetja* **1**)
$6 : 2 = 3$ (*mbetja* **0**)
$3 : 2 = 1$ (*mbetja* **1**)
$1 : 2 = 0$ (*mbetja* **1**) *Rezultati*: $(1101)_2$



Shifrat e shënimit binar përftohen si mbetje gjatë pjesëtimit të numrit të dhënë me 2. Numri binar përftohet duke shënuar shifrat e përftuara të shënimit binar në renditje të kundërt nga renditja e njehsimit.

Prandaj përdoret ndryshorja ndihmëse *vendRen* që paraqet vlerën rendore (shkallën e numrit 10) të shifrës së shënimit.

n	*bin*	*vendRen*	*c*
13	0	1	
6	1	10	1
3	1	100	0
1	101	1000	1
0	1101	10000	1

Në mënyrë analoge shënohet skema algoritmike për konvertimin e numrit nga sistemi binar në sistemin dhjetor (decimal),

P.sh. $(\mathbf{1101})_2 \rightarrow \mathbf{1}*2^3 + \mathbf{1}*2^2 + \mathbf{0}*2^1 + \mathbf{1}*2^0 = 8+4+1 \rightarrow (13)_{10}$,

çfarë mbetet të punohet si detyrë në shtëpi.

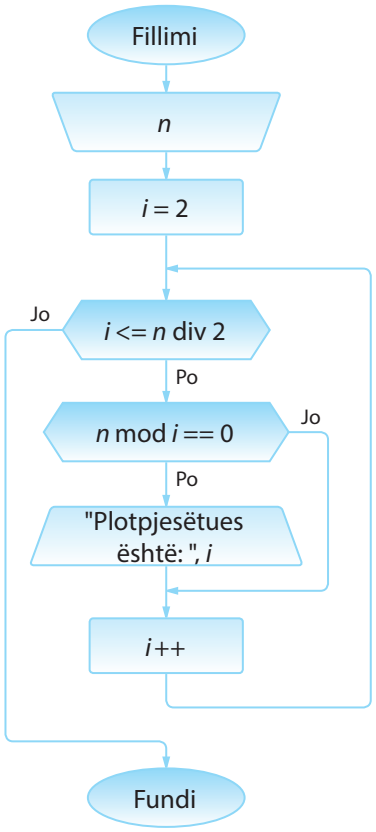
Shembulli 35.
Shkruaje skemën algoritmike me të cilën numri *n* i dhënë në sistemin decimal paraqitet në sistemin heksadecimal.

P.sh. 26 : 16 = 1 (mbetja 10 = A)
1 : 16 = 0 (mbetja 1)
Rezultati: $(1A)_{16}$

Shembulli 36.
Shkruaje skemën algoritmike me të cilën kontrollohet nëse numri i dhënë është i thjeshtë. Numri është i thjeshtë, në qoftë se është i plotpjesëtueshëm vetëm me vete dhe me numrin 1.

Shembulli 37.
Shkruaje skemën algoritmike me të cilën përcaktohet numri i thjeshtë i *n*të sipas madhësisë.
(P.sh. për $n = 3$ dalja është 5, si numri i tretë në vargun e numrave të thjeshtë:
2, 3, 5, 7, 11...)

Algoritmi 32. Të shkruhet skema algoritmike nëpërmjet të cilës përcaktohen të gjithë plotpjesëtuesit e numrit të dhënë *n*, përveç 1 dhe vetë atij numri.



Për numrin e dhënë ***n***, plotpjesëtuesit e mundshëm janë të gjithë numrat prej 1 deri në *n* div 2, dhe vetë numri *n*.

Përkufizohet ndryshorja ndihmëse ***i*** nëpërmjet të cilës për secilin numër nga intervali [2, *n* div 2] kontrollohet nëse është plotpjesëtues i numrit *n* (d.m.th. a është mbetja e pjesëtimit të numrit *n* me atë numër e barabartë me zero).

1.4. Tiparet e algoritmeve

Në seksionet paraprake kemi theksuar se algoritmi duhet të përkufizohet në mënyrë precize, pa situata që mund të keqkuptohen si dhe pa paqartësi. Vetitë e tjera të algoritmeve janë:

- **Përkufizimi korrekt.** Çdo hap në algoritëm duhet të përkufizohet në mënyrë të qartë. Bashkësia e rregullave duhet të jetë e tillë që veprimet të cilat përkufizohen mund të kryhen sipas radhës njëri mbas tjetrit.
- **Determinizmi (të përcaktuarit).** Vlera që përftohet mbas kryerjes së secilit hap, është përcaktuar në mënyrë unike nga vlerat e hapit paraprak.
- **Thjeshtësia.** Ligji (rregulla) e përftimit të madhësive dalëse duhet të jetë e qartë dhe e thjeshtë.
- **Përfundimi.** Algoritmi duhet të përbëhet prej një numri të fundmë hapash që kryhen në një interval kohor të fundmë. Numri i hapave mund të jetë shumë i madh.
- **Rezultati.** Për çdo bashkësi të madhësive hyrëse, algoritmi duhet të japë rezultat.
- **Masiviteti.** Algoritmi duhet të sigurojë zgjidhjen e një klase të tërë të problemave që dallohen vetëm sipas vlerave të madhësive hyrëse.

1.5. Kontrolli i saktësisë së algoritmit

Krijimi i algoritmeve ka natyrë kreative dhe nuk ekzistojnë rregullat universale sipas të cilave puna mund të përfundohet.

Vetëm te strukturat e thjeshta, siç janë strukturat lineare, saktësia mund të kontrollohet me verifikimin e kujdesshëm të të gjithë hapave.

Për kontrollin e saktësisë së algoritmit më shpesh përdoret testimi. Zgjidhet një numër i caktuar shembujsh dhe testohen hapat e algoritmit, mirëpo testimi i këtillë është më praktik për identifikimin e gabimit në algoritëm sesa për verifikimin e saktësisë së tij. Me fjalë të tjera, testimi mund të shërbejë vetëm për prezencën e gabimit, mirëpo me testim nuk mund të vërtetohet se nuk ka gabim në algoritëm. Testimi i skemave algoritmike merr kohë të gjatë dhe shpeshherë mund të sjellë te gabimet që bën njeriu. Prandaj sot, kontrolli i saktësisë së një algoritmi bëhet duke kontrolluar punën e programit të shkruar nëpërmjet të atij algoritmi.

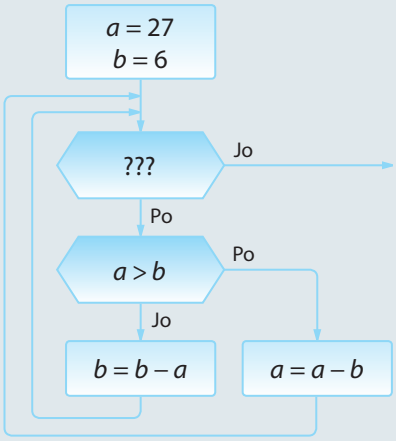


Figura 1.16.

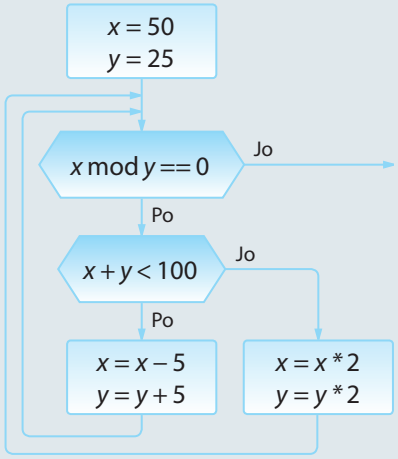


Figura 1.17.

Pyetje dhe detyra kontrolli

1. Cila nga komandat e mëposhtme duhet vendosur në vend të ??? në figurën 1.16, me qëllim që cikli të kryhet saktësisht 3 herë?

a) $b < 5$
b) $a \mathrel{!}= b$
c) $b < a$
d) $a > 5$
e) Asnjë komandë e përmendur më lart.

2. Cilat janë vlerat e ndryshoreve x dhe y në daljen nga skema algoritmike në figurën 1.17?

a) $x = 45,\ y = 30$
b) $x = 100,\ y = 50$
c) $x = 40,\ y = 35$
d) $x = 200,\ y = 100$
e) Asnjë komandë e përmendur më lart.

3. Cilat janë vlerat e ndryshoreve x dhe y në daljen nga skema algoritmike në figurën 1.18?

a) $x = 45,\ y = 50$
b) $x = 45,\ y = 45$
c) $x = 45,\ y = 55$
d) $x = 40,\ y = 55$
e) Asnjë komandë e përmendur më lart.

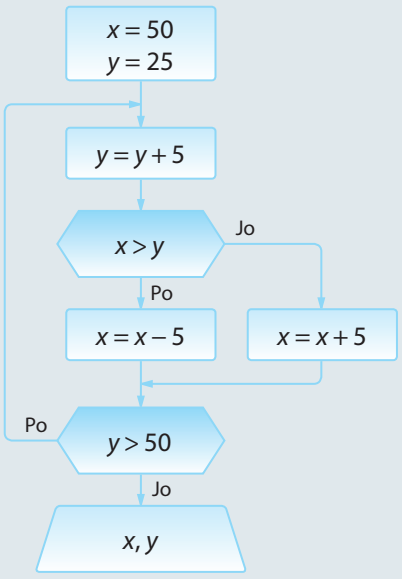


Figura 1.18.

Puno vetë

1. Të shkruhet skema algoritmike me të cilën lexohen n numra, pastaj njehsohet shuma e numrave pozitivë dhe shuma e numrave negativë.

2. Të shkruhet skema algoritmike nëpërmjet të cilës nga segmenti i dhënë $[a, b]$ përcaktohen të gjithë numrat që janë të plotpjesëtueshëm me 3, por jo me 2.

3. Të shkruhet skema algoritmike që për vargun e dhënë të n numrave, kontrollon nëse është i radhitur në renditje rënëse.

4. Të shkruhet skema algoritmike me të cilën përcaktohet sa herë shifra 2 është paraqitur në shënimin e numrit pesëshifror n.

5. Të shkruhet skema algoritmike e cila për numrin e dhënë a_0 krijon vargun me gjatësi n sipas rregullës së mëposhtme.

$$a_{n+1} = \begin{cases} \frac{a_n}{2}, & a_n - \text{çift} \\ 3a_n & a_n - \text{tek} \end{cases}$$

6. Të shkruhet skema algoritmike sipas të cilës shtypen elementet e vargut $i^3 - 3i + n$, $(i = 1, \ldots, n)$ të plotpjesëtueshëm me 7, në bazë të vlerës hyrëse për n.

7. Të shkruhet skema algoritmike me të cilën përcaktohen të gjithë numrat katërshifrorë simetrikë. (Numri është simetrik në qoftë se i ka të barabarta shifrën e parë dhe të katërt, si dhe shifrën e dytë dhe të tretë. P.sh. 5775).

8. Të shkruhet skema algoritmike që përcakton të gjithë numrat binjakë më të vegjël se 100. Dy numra janë binjakë në qoftë se janë të thjeshtë dhe ndryshojnë për 2 (p.sh. 3 dhe 5, 5 dhe 7, 11 dhe 13).

9. Të shkruhet skema algoritmike për njehsimin e thyesës "zinxhir":

$$S = \cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\ddots\,\cfrac{1}{109+\cfrac{1}{111}}}}}}$$

II.

PROGRAMIMI DHE GJUHA E PROGRAMIMIT JAVA

Në këtë kapitull u jepen përgjigje pyetjeve të mëposhtme:

- Çfarë është sistemi kompjuterik?
- Për çfarë shërben gjuha e programimit dhe si shkruhet programi?
- Për çfarë shërbejnë përkthyesit programorë?
- Cilat lloje të gjuhëve programuese ekzistojnë?
- Cilat paradigma programore ekzistojnë dhe cilat janë karakteristikat e secilës prej tyre?

Gjithashtu do të përshkruhen:

- gjuha e programimit Java dhe makinat virtuale të Javës,
- procesi i instalimit të rrethinës zhvilluese Eclipse,
- karakteristikat themelore të gjuhës programuese Java,
- llojet e prodhimeve softuerë që mund të krijohen me ndihmën e gjuhës programuese Java.

Njohja e algoritmeve është parakushti për programim të suksesshëm. Siç është theksuar më herët, nëpërmjet algoritmeve paraqitet zgjidhja logjike e problemit të paraqitur dhe është e pavarur nga gjuha e programimit. Me qëllim që në bazë të zgjidhjes algoritmike të përftohet zgjidhja programore, është e domosdoshme njohuria e sistemit kompjuterik dhe një gjuhë programimi e caktuar.

Sistemi kompjuterik

Në kapitullin paraprak është sqaruar se si zgjidhja e një problemi real paraqitet me skemën algoritmike. Implementimi i zgjidhjes algoritmike varet nga sistemi kompjuterik në të cilin programi do të kryhet (ekzekutohet). Shumë shpesh zgjedhja e sistemit kompjuterik përcakton bashkësinë e gjuhëve programuese nëpërmjet të cilëve zgjidhja algoritmike e propozuar mund të implementohet.

Sistemi kompjuterik, gjegjësisht kompjuteri, përbëhet nga bashkësia e pajisjeve elektronike të parapara për hyrjen, përpunimin dhe paraqitjen e të dhënave.

Sistemi kompjuterik mundëson hyrjen e të dhënave duke zbatuar pajisjet hyrëse (siç janë tastiera, mausi, skaneri, disqet magnetike dhe optike, etj). Të dhënat e futura (hyrëse) ruhen përkohësisht në memorien punuese, pastaj procesori, në bazë të instruksioneve i përpunon, dhe në fund rezultatet e përpunimit paraqiten në aparatin dalës (p.sh. monitor, printer) ose ruhen përherë në njërën prej memorieve të jashtme (hard disk, disketë, CD ROM, DVD etj). Shpeshherë ndodh që procesi i përpunimit të jetë i përcjellë nga ana e përdoruesit. Për një qëllim të tillë përdoren aparatet dalëse (monitori, printeri, ploteri etj.), të cilët në mënyrë adekuate mundësojnë paraqitjen e të dhënave dalëse.

Sistemi kompjuterik përbëhet nga dy komponentë: hardueri dhe softueri.

Hardueri kompjuterik

Softueri kompjuterik

Hardueri kompjuterik paraqet aparatet fizike (elektronike dhe mekanike) të sistemit kompjuterik, gjegjësisht të gjitha ato pjesë që shihen dhe mund të ndihen fizikisht.

Softueri kompjuterik është bashkësia e programeve në bazë të të cilave hardueri i përpunon të dhënat.

Softueri kompjuterik ndahet më shpesh në tri grupet e mëposhtme:

- Softueri i sistemit;
- Softueri aplikativ;
- Gjuhët programuese.

Kur furnizuesi i dorëzon kompjuterin blerësit, zakonisht së bashku me harduer dorëzohet edhe një pjesë e softuerit që mundëson funksionimin e rregullt të komponentëve të harduerit dhe sistemit kompjuterik në tërësi. Kjo bashkësi softuerësh quhet ***softueri*** i ***sistemit*** dhe ai përmban: sistemin operues, përkthyesit programorë dhe programet shërbyese. Sistemi i softuerit i lehtëson përdoruesit punën në kompjuter.

Platforma kompjuterike

Platforma kompjuterike paraqet kombinimin e harduerit kompjuterik dhe softuerit përkatës të sistemit.

Softueri aplikativ është emërtimi i përbashkët për programet dhe sistemet programuese të dedikuara përdoruesit për zgjidhjen e problemave konkrete.

Me qëllim që të zgjidhet një problem, kompjuterit duhet t'i përshkruhen të gjitha hapat (instruksionet dhe komandat), të cilat ai i ekzekuton dhe të dhënat që përpunohen me instruksione të dhëna.

Programi është një bashkësi e instruksioneve të shkruara për zgjidhjen e një problemi.

Programimi paraqet procesin e shënimit të instruksioneve.

Programi dhe programimi

Gjuha është sistem gjestikulimesh, simbolesh dhe fjalësh që zbatohen për paraqitjen dhe këmbimin e ideve, shenjave dhe mendimeve, ashtu që të paraqesë mjetin për përshkrimin dhe përcjelljen e informacioneve. Prandaj njerëzit përdorin gjuhën në komunikimin e përditshëm.

Që njeriu të mund t'i përcjellë kompjuterit një detyrë që duhet të zgjidhë, ai në njëfarë mënyre duhet të përdorë gjuhën që është e kuptueshme për të edhe për kompjuterin. Gjuha e programimit është mjeti me të cilin njeriu i kumton kompjuterit programin, d.m.th. përkufizon një varg instruksionesh me të cilat harduerit kompjuterik i urdhërohet ekzekutimi i një bashkësie veprimesh, të cilat çojnë deri te qëllimi i dëshiruar (zgjidhja e problemit).

Gjuha e programimit paraqet bashkësinë e simboleve, fjalëve kyç dhe rregullave për shënimin e programit, me të cilën kompjuterit i paraqiten instruksionet dhe i përshkruhen të dhënat.

Gjuha e programimit

Fjalët kyçe ose ***të rezervuara*** janë pjesë përbërëse e gjuhës programuese dhe përbëhen nga simbolet e grumbulluara në fjalë.

Bashkësia e rregullave të cilat përkufizojnë se si mbi simbolet e gjuhës ndërtohen konstruksionet elementare dhe konstruksionet e ndërlikuara quhet ***gramatikë e gjuhës***.

Gramatika e gjuhës

Disiplina që hulumton konstruksionet gramatike korrekte dhe jep rregullat e zbulimit formal të gabimeve në konstruksionet e gjuhës, quhet ***sintaksë e gjuhës***. Hulumtimi i një gjuhe programuese nënkupton njohurinë e sintaksës së saj. Në qoftë se gjatë shënimit të programit formohen gabime në sintaksë, përkthyesit programorë (mbi të cilët do të flitet në kapitullin 2.4) kanë mundësinë e zbulimit automatik të gabimit.

Sintaksa e gjuhës

Disiplina që hulumton domethënien e konstruksioneve të veçanta të gjuhës quhet ***semantikë***. Gjatë shënimit të programit të gjitha konstruksionet e gjuhës programuese krijohen në bazë të algoritmit të përpiluar për zgjidhjen e detyrës së caktuar. Në ndryshim nga gabimet në sintaksë, përkthyesi programor nuk mund të zbulojë gabimet në semantikë. Ato gabime mund t'i zbulojë vetëm njeriu.

Semantika e gjuhës

2.1. Klasifikimi i gjuhëve programuese

Nga paraqitja e parë e kompjuterëve, ata përsosen në mënyrë permanente, si në pikëpamjen fizike (komponentët, forma, teknologjia e përpunimit, dimensionet, harxhimi i energjisë etj.), gjithashtu edhe sipas karakteristikave funksionale (shpehtësisë së punës, numrit të veprimeve, sasisë së informatave etj.). Çdo gjeneratë kompjuterësh dhe sistemesh kompjuterike, krahas karakteristikave fizike dhe funksionale, karakterizohet edhe nëpërmjet gjuhës programuese për zhvillimin e softuerit. Kështu, krahas zhvillimit të sistemit kompjuterik, janë zhvilluar edhe gjuhët programuese që janë përdorur për komunikimin e njeriut dhe kompjuterit.

Klasifikimi i gjuhëve të programimit (programuese) kryhet në bazë të disa kritereve sipas qëllimit të caktuar, mënyrës së ekzekutimit (kryerjes), strukturimit dhe gjeneratës së kompjuterëve për të cilët është paraparë.

1. Sipas ***namjeni***, dallojmë këtë gjuhë programuese:
   - gjuhët programuese për probleme numerike,
   - gjuhët programuese për probleme biznesi,
   - gjuhët programuese të bazuara në lista dhe vargje,
   - gjuhët programuese me interesa të ndryshme.
2. Sipas ***mënyrës së ekzekutimit***, dallojnë këto gjuhë të programimit:
   - gjuhët programuese imperative,
   - gjuhët programuese funksionale,
   - gjuhët programuese të orientuara në një qëllim,
   - gjuhët programuese të orientuara në objekte,
   - gjuhët hibride.
3. Sipas ***strukturimit***, dallojmë këto gjuhë programimit:
   - gjuhët programuese procedurale,
   - gjuhët programuese jo-procedurale.

   Fokusi i gjuhëve programuese procedurale është zbërthimi i detyrave në ndryshore, në struktura të dhënash dhe në nën-programe. Programet e shkruara në gjuhët programuese procedurale përbëhen nga tërësitë programore (procedurat, funksionet, nënprogramet) që mund të kryhen në mënyrë të pavarur, si dhe komandat e kushtëzuara dhe ciklet për udhëheqje gjatë programit.

   Gjuhët jo-procedurale nuk kanë komandat për udhëheqje gjatë programit. Përdoruesi përshkruan detyrën që duhet zgjidhur, kurse gjuhës programuese i lejohet mënyra e zgjidhjes së detyrës.
4. Një nga kategorizimet më të përgjithshme të gjuhëve programuese është ***kategorizimi*** në bazë të gjeneratave. Ekzistojnë pesë gjenerata të gjuhëve programuese:
   - Gjenerata e *parë* karakterizohet me përdorimin e gjuhës së makinës. Programet janë të shkruara me anë të shënimit binar, që është drejtpërdrejt i kuptueshëm për procesorin në të cilin ekzekutohet.
   - Gjeneratën e *dytë* e karakterizon zbatimi i gjuhëve simbolike të ashtuquajturve asemblerëve. Nga gjuhët e gjeneratës së parë i dallon përdorimi i simboleve në vend të shënimit binar.
   - Gjenerata e *tretë* karakterizohet me përdorimin e paradigmës procedurale, e cila nënkupton që programet ekzekutohen duke thirrur procedurën (Pascal, C, C++, Java, Visual Basic, C#).

Disa gjuhë programuese kanë numër të kufizuar instruksionesh dhe janë të parapara për shënimin më të lehtë të programeve të specializuara ngushtë ashtu që karakteristikat e tyre (shpejtësia e ekzekutimit, zënia me punë e memories, përdorimi i lehtë...) të jenë sa më të mira.

Në gjuhët procedurale kompjuterit i japim bashkësinë komplete të instruksioneve me të cilat zgjidhet problemi, d.m.th. i japim algoritmin për zgjidhjen e detyrës. Shembuj të gjuhës procedurale janë Fortran-i, Cobol, Basic, Pascal, C.

Gjuhët e orientuara në objekte janë Java, C++, C#, SmallTalk.

Me gjuhë deklarative përshkruhet se çfarë është e njohur mbi problemin dhe cili është qëllimi i zgjidhjes së tij, kurse sistemi (interpreteri) vet vjen deri te procesi për zgjidhjen e problemit. Shembuj gjuhësh deklarative janë Prolog dhe SQL.

– Gjenerata e *katërt* karakterizohet me zbatimin e gjuhëve programuese joprocedurale (gjuhët programuese deklarative të nivelit të lartë: Prolog, SQL, gjeneratorët e kodit dhe mjedisit të përdoruesit). Këto gjuhë kombinojnë mjete të ndryshme me qëllim thjeshtëzimi të procesit të programimit.

– Gjenerata e *pestë* përfshin zbatimin e gjuhëve ngushtë të specializuara: inteligjencës artificiale dhe gjuhët e programimit të bazuara në gjuhën natyrore.

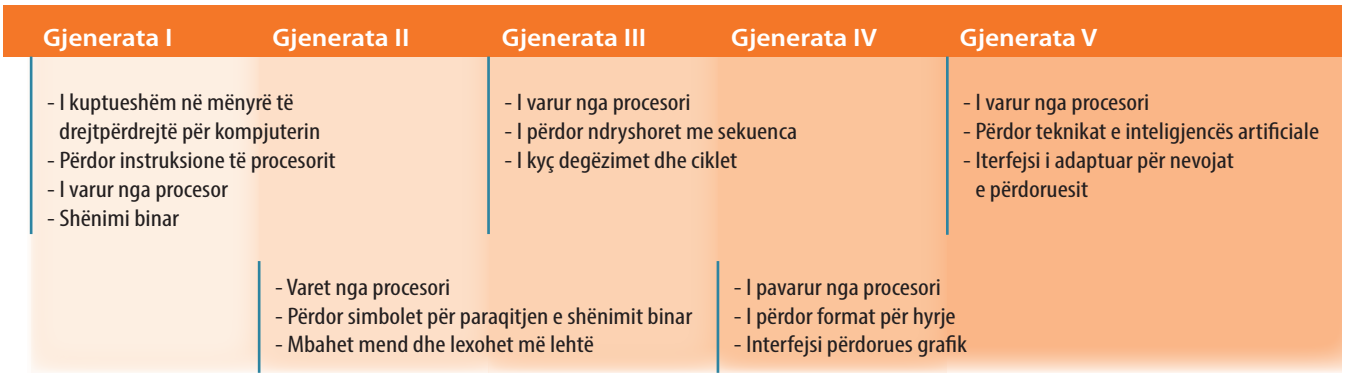


Figura 2.1. Gjeneratat e gjuhëve programuese dhe karakteristikat e tyre

2.2. Gjuhët e programimit të nivelit të ulët dhe të lartë

Në mënyrë identike si te komunikimi i njerëzve, ku përdoret një numër i madh gjuhësh (anglisht, gjermanisht, frëngjisht...), për komunikimin me kompjuter sot përdoret edhe një numër i madh gjuhësh, madje edhe për një tip të njëjtë kompjuterësh. Sa më e ngjashme që të jetë gjuha programuese me gjuhën natyrore, niveli i saj do të jetë më i lartë. Me ndihmën e gjuhëve programuese të nivelit të ulët është më vështirë për të programuar, mirëpo, zakonisht, ato programe ekzekutohen më lehtë.

Në nivelin më të ulët është ***gjuha e makinës*** që përbëhet nga vargu i kodeve të shënuara me numra binarë (me zero dhe njëshe). Një kod i shënuar kështu paraqet komanda të brendshme të procesorit të sistemit kompjuterik. Programimi në gjuhën e makinës është shumë i komplikuar dhe kërkon njohuri të mirë të arkitekturës së sistemit kompjuterik në të cilin programi do të ekzekutohet.

Instruksionet e programit dhe të dhënat futen në formën binare dhe është vështirë t'i dallojmë, çfarë paraqet shkakun e shumë gabimeve në programim. Kodi burimor lexohet vështirë dhe kuptohet, gjë që procesin e mirëmbajtjes dhe superstrukturën e programeve e bën dukshëm më të komplikuar.

Duke u zhvilluar gjuhët simbolike janë tejkaluar disa nga vështirësitë e shënimit të programit në gjuhën e makinës.

Gjuha simbolike në vend të instruksioneve të shënuara me një varg bitesh, i përdor shkurtesat për operacione (veprime) dhe shenja simbolike të të dhënave. Në këtë mënyrë procesi i programimit në masë të konsiderueshme është lehtësuar, por edhe më tutje varet nga procesori konkret, d.m.th. edhe më tutje nevojitet të njihen karakteristikat teknike të një kompjuteri konkret për të cilin programi shkruhet.

Shembulli 1.
Të shkruhet komanda që shënon vlerën 1234 në regjistër (lokacionin e memories) CX në procesor. Rendi i parë në tabelën e mëposhtme paraqet komandën e shkruar në gjuhën e makinës. Në rendin e dytë të tabelës është paraqitur shënimi më i shkurtër në sistemin heksadecimal. Rendi i tretë i tabelës paraqet komandën e shënuar duke përdorur gjuhën simbolike (asemblerin).

1000 1011	0000 1110	0011 0100	0001 0010
8B	0E	34	12
MOV	CX,	1234H	

Biti është njësia më e vogël e informatës në programim. Një bit paraqet sasinë e informacionit të nevojshëm për dallimin e dy gjendjeve të kundërta ndërmjet tyre.

Gjuha simbolike (asembleri)

Shembulli 2.
Të shkruhet programi në gjuhën simbolike (asembler) nëpërmjet të cilit vlera nga regjistri AX zvogëlohet për 32, në qoftë se ajo i takon intervalit (97, 122). Në të kundërtën, vlera e regjistrit AX mbetet e pandryshuar.

```
SUB32  PROC          ; procedure begins here
       CMP   AX,97   ; compare AX to 97
       JL    DONE    ; if less, jump to DONE
       CMP   AX,122  ; compare AX to 122
       JG    DONE    ; if greater, jump to DONE
       SUB   AX,32   ; subtract 32 from AX

DONE:  RET           ; return to main program
SUB32  ENDP          ; procedure ends here
```

Që kompjuteri të mund të kryejë programin e shënuar në gjuhën simbolike, programi duhet të përkthehet në gjuhën e makinës. Një komandë e gjuhës simbolike i përgjigjet një komande të gjuhës së makinës. Procesi i përkthimit të komandave të gjuhës simbolike në instruksione makine është automatizuar plotësisht, sepse krijohet një program i veçantë i cili si hyrje përfton programin e shënuar në gjuhën simbolike, dhe si dalje jep programin përkatës në gjuhën e makinës. Programi që bën përkthimin nga gjuha simbolike në atë të makinës quhet ***asembler***. Prandaj gjuha simbolike quhet shpeshherë gjuha asembler ose shkurtimisht asembler.

Instruksionet e shënuara në gjuhën simbolike kryhen shumë shpejt, me çfarë arrihet përdorimi optimal i të gjitha resurseve në diskonim.

Mangësitë e gjuhës simbolike, para së gjithash, manifestohen në:

- varësia nga tipi i kompjuterit – programi i shkruar në gjuhën simbolike të një kompjuteri nuk mund të ekzekutohet pa ndryshime shtesë në një tip tjetër kompjuteri,
- domosdoshmëritë e përshkrimit të hollësishëm të procesit të përpunimit,
- domosdoshmëritë e njohurisë në nivel specialisti të problemit që zgjidhet dhe të sistemin kompjuterik në të cilin programi do të kryhet.

Gjuhët e programimit të nivelit më të lartë

Në gjuhët e programimit të nivelit të lartë përshkrimi i komandave dhe të dhënave bëhet në mënyrë të përafërt me gjuhën natyrore (gjuhën angleze). Në këto gjuhë, një komande i përgjigjen më shumë komanda të gjuhës simbolike (gjegjësisht gjuhës së makinës). Gjuhët e nivelit të lartë kanë shkallë të lartë të pavarësisë në raport me arkitekturën e kompjuterit dhe sistemit operues në të cilin ekzekutohen.

Meqenëse kompjuteri kupton vetëm programin e shënuar në gjuhën e makinës, çdo program i shënuar në gjuhë të nivelit më të lartë duhet të përkthehet në gjuhën e makinës.

Përparësitë e përdorimit të gjuhëve programuese të nivelit të lartë karakterizohen nga:

- thjeshtësia e programimit,
- adaptimi në detyra specifike,
- modulariteti dhe përdorimi i shumëfishtë,
- koha e kufizuar e trajnimit për programim,
- nuk është e domosdoshme njohuria e komponentëve të harduerit të sistemit kompjuterik në të cilin ekzekutohet programi,
- transferueshmëria e programit – programi mund të ekzekutohet edhe pa problem në sisteme të tjera kompjuterike.

Përdorimi i gjuhëve programuese të nivelit të lartë ka edhe mangësitë e veta:

- duhet të bëhet përkthimi në gjuhën e makinës (që sjell me vete humbjen e kohës dhe zënien në punë të memories dhe procesorit),
- programi burimor optimal nuk jep gjithmonë programin optimal ekzekutiv.

Gjuha e programimit e nivelit më të lartë

2.3. Paradigmat e programit

Paradigma e programit përcakton stilin e programimit, gjegjësisht këndvështrimin që programuesi ka në program dhe ekzekutimin e tij.

Paradigma e programit

Është e dobishme që të ilustrohen gjuhët programuese të ndryshme në shkallën lineare në të cilën pozicioni i gjuhës është i përcaktuar nëpërmjet shkallës në të cilën përdoruesi është i varur nga platforma e kompjuterit (figura 2.2.).

Sipas ndarjes së tillë, në pjesën e majtë të fundit të shkallës janë gjuhët programuese me ndihmën e të cilave njerëzit u adaptohen karakteristikave të kompjuterit gjatë zgjidhjes së problemave. Sa më shumë që i afrohemi pjesës së djathtë të shkallës, takohemi me gjuhët e programimit të cilat mundësojnë që kompjuteri t'i adaptohet nevojave të njerëzve. Në realitet, zhvillimi i gjuhëve programuese nuk ka rrjedhë në këtë mënyrë, por përgjatë shkallëve të ndryshme, që karakterizojnë qasjen e ndryshme të procesit të programimit (d.m.th. paradigmën).

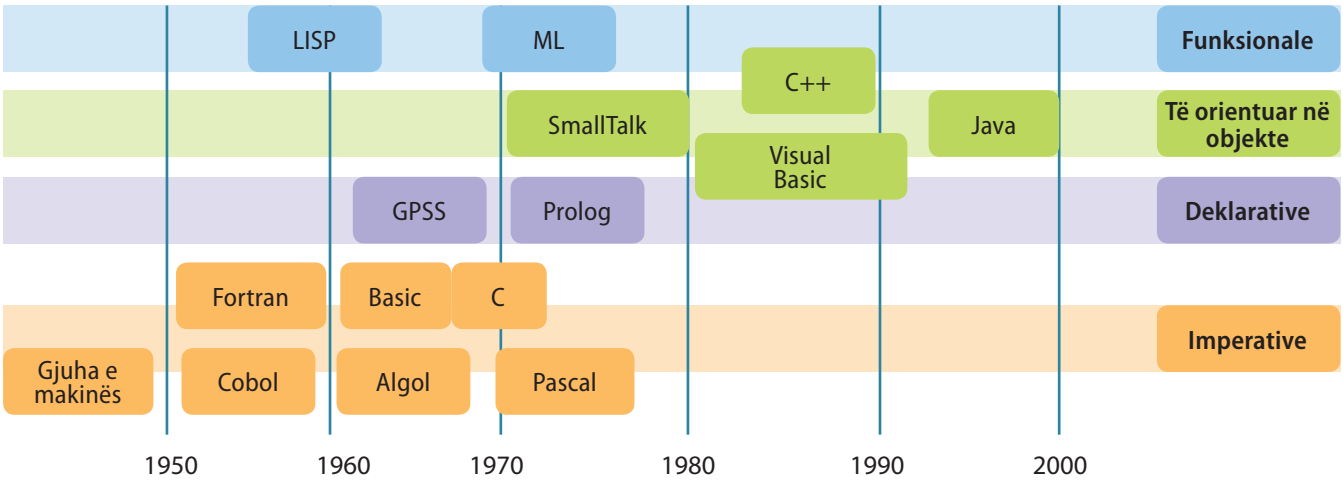
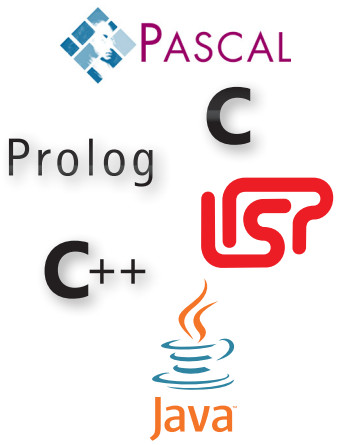


Figura 2.2. Paraqitja në gjenerata e zhvillimit të gjuhëve programuese në kuadrin e zhvillimit të paradigmave të programit

Paradigma imperative ose procedurale paraqet qasjen tradicionale të procesit të programimit. Në paradigmën imperative themelohen programet e shkruara në gjuhët simbolike dhe gjuhët e makinës. Me paradigmën imperative, procesi i programimit përkufizon një varg sekuencash të komandave të cilat në mënyrë përkatëse udhëheqin me të dhënat, që të përftohet rezultati i dëshiruar. Shembuj të gjuhëve programuese nga ky grup janë: *C*, *Pascal*, *Basic*.



Paradigma deklarative nga programuesi kërkon që problemin ta përshkruaj në mënyrë përkatëse. Mënyra e përshkrimit të problemit është e diktuar nga algoritmi paraprakisht i përkufizuar. Në atë mjedis, programuesi është ai që jep deklarata precize të lidhura me problemin, por që nuk krijon algoritmin për problemin e caktuar. Pikërisht, kuptimi i algoritmit ekzistues është vështirësia kryesore për zhvillimin e programit që themelohet në paradigmën programuese. Prandaj gjuhët deklarative të para kanë pasur qëllim shumë të caktuar për krijimin e një lloji të caktuar të aplikacioneve. Shembull gjuhe programuese deklarative është *Prologu*.

Paradigma funksionale përkufizon procesin e zhvillimit të programit si mënyrë lidhjeje të moduleve programuese të pavarura në mënyrë funksionale. Secili modul funksional ka parametrat hyrës të përkufizuar në mënyrë të qartë, funksionin që kryen mbi parametrat hyrës dhe parametrat dalës të pritur. Gjuhët e programimit kanë modulet programore themelore funksionale. Nga programuesi pritet që nëpërmjet tyre të zhvillojë modulet programore të reja (komplekse) me qëllim të caktuar. Në momentin kur zhvillohet funksioni i ri, ai mund të bëhet funksion primitiv i një funksioni më kompleks (p.sh.

brenda funksionit kompleks mund të përdoret funksioni themelor). Shembuj gjuhësh funksionale janë *LISP* dhe *ML*. Versioni origjinal i *LISP*-it ka pasur vetëm disa module funksionale themelore, kurse *LISP*-i i sotshëm ka qindra module.

Paradigma e orientuar në objekte (*POO*) është një prej paradigmave programore që përdor objekte si bazë për projektimin e programeve dhe softuerëve me qëllime të ndryshme. Bazohet në konceptet e orientuara në objekte, siç është trashëgimia, polimorfizmi dhe enkapsulacioni, që do të sqarohen më vonë gjatë njohjes me gjuhën e programimit JAVA, si përfaqësuesin udhëheqës të kësaj paradigme programore.

Nga vitet 80-të deri sot kjo paradigmë është bërë paradigma me influencë më të madhe në zhvillimin komercial të softuerit. Gjuhët e programimit siç është *C++* dhe JAVA, me popullaritetin e vetë kanë përforcuar statusin udhëheqës të paradigmës së orientuar në objekte gjatë përpunimit të softuerit.

2.4. Përkthyesit e programit

Programi i shkruar në një gjuhë programimi të nivelit të lartë quhet ***kodi burimor (origjinal)***. Që kodi burimor i programit të mund të kryhet në kompjuter, është e domosdoshme që ai të përkthehet në gjuhën e makinës, sepse ajo është gjuha e vetme të cilën e kupton procesori.

Kodi burimor

Kodi i përkthyer burimor i programit në gjuhën e makinës quhet ***kodi ekzekutiv***. Procesi i përkthimit është automatizuar, falë programeve speciale (të ashtuquajturit përkthyesit e programit) që bëjnë përkthimin e kodit burimor në kodin ekzekutiv.

Kodi ekzekutiv

Dallojmë llojet e mëposhtme të përkthyesve të programit: kompajlerët, intrerpreterët dhe përkthyesit hibridë.

Përkthyesit e programit

> ***Kompajleri*** është program që përkthen kodin burimor (origjinal) komplet të programit të shkruar në një gjuhë të programimit të nivelit të lartë në gjuhën e makinës, duke krijuar kodin unik ekzekutiv.

Kompajleri

Matematikania Grejs Huper në vitin 1952 ka krijuar A-0 kompajlerin.

Procesi i kompajlimit mund të ndahet në dy faza:

- faza e analizës së programit burimor,
- faza e sintezës së kodit ekzekutiv.

Faza e analizës së programit burimor

Faza e analizës së programit burimor përbëhet nga tre hapa: analiza leksikore, sintaksore dhe semantike.

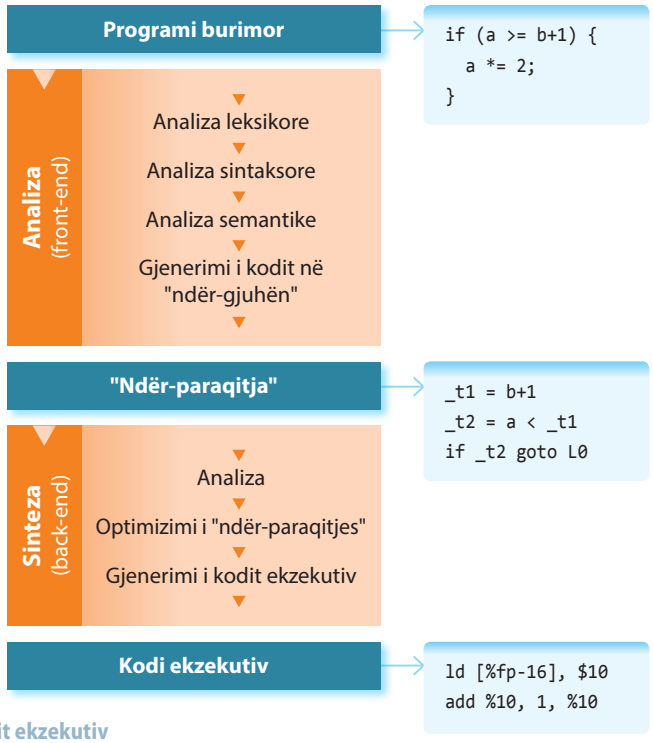
- *Analiza leksikore.* Gjatë analizës leksikore simbolet e kodit burimor grumbullohen në elementet themelore të gjuhës që quhen tokene, pastaj secili token zëvendësohet me një simbol unik. Rregullat leksikore përcaktojnë bashkësinë e tokeneve të gjuhës së programimit të shkruara si duhet.
- *Analiza sintaksore.* Gjatë analizës sintaksore, identifikimi i strukturës sintaksore të programit kryhet nëpërmjet procesit të parsimit (analizës gramatike) të sekuencave të tokeneve. Detyra e analizës sintaksore është që të verifikojë nëse programi është i shkruar në pajtim me rregullat gramatikore të gjuhës së programimit.
- *Analiza semantike.* Gjatë kësaj faze në bazë të rregullave semantike kryhen kontrolle semantike, lidhja e objekteve, shoqërimi i vlerave, publikimi i paralajmërimeve ose refuzimi i programit. Rregullat semantike janë interpretimet e rregullave që bëjnë lidhjen e ekzekutimit të programit me sjelljen e kompjuterit.

Tokenet janë ndryshoret, operatorët, fjalët kyçe, konstantat...

Gjatë ***fazës së sintezës së kodit ekzekutiv*** kryhen tri procese të përkthimit:

- *Analiza.* Gjatë kësaj faze kryhet grumbullimi dhe analiza e të dhënave hyrëse të programit. Analiza e drejtë është bazë për procesin pasues, procesin e optimizmit. Analiza tipike është e ashtuquajtura *analiza e rrjedhjes së të dhënave* (anglisht *dataflow*).
- *Optimizimi.* Gjatë kësaj faze kryhet transformimi i kodit në formë më të shpejtë ose më të vogël (ndërkodi) me mbajtje të domosdoshme të funksionalitetit. Metodat/teknikat popullore të optimizimit janë: eliminimi i kodit të vdekur, shumëzimi i konstantave, transformimi i cikleve, shpërndarja (alokacioni) i regjistrave dhe paralelizimi automatik.
- *Gjenerimi i kodit ekzekutiv.* Gjatë kësaj faze ndërkodi i përftuar gjatë procesit të optimizimit përkthehet në kodin ekzekutiv (kodin e makinës). Aktivitetet më të shpeshta në këtë fazë janë: vendimi mbi mënyrën e ruajtjes së ndryshoreve në regjistra dhe në memorie, zgjedhja dhe planifikimi kohor i instruksioneve përkatëse të makinës, etj.

Faza e sintezës së kodit ekzekutiv



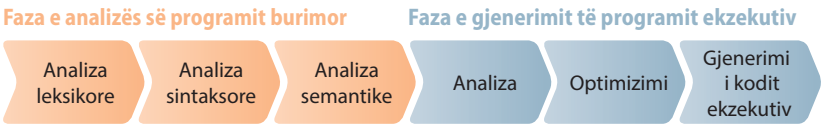


Figura 2.3. Procesi i kompilimit të programit

Interpreteri

Interpreter është përkthyesi që përkthen dhe kryen instruksione në bazë të instruksioneve të programit burimor në kodin ekzekutiv.

Interpreteri është program që mundëson ekzekutimin e kodit burimor, në atë mënyrë që secila komandë përkthehet në një ose më shumë komanda makine, varësisht nga kompleksiteti, pas të cilës ekzekutohen. Interpreteri si rezultat të përkthimit nuk gjeneron një kod unik ekzekutiv (siç vepron kompajleri), por përkthimi kryhet gjatë çdo inicimi të programit burimor. Prandaj ekzekutimi i programit nëpërmjet interpreterit është më i ngadalshëm sesa me kompajler. Një ndër përparësitë e interpreterit manifestohet në faktin se çdo instruksion analizohet menjëherë, gë që lehtëson mënjanimin e gabimeve.

Përkthyesi hibrid

Përkthyesit hibridë duke kombinuar teknikat e kompajlerave dhe interpreterëve përkthejnë programet e shënuara në kodin burimor në ndërkodin, i cili mundëson interpretim më të lehtë.

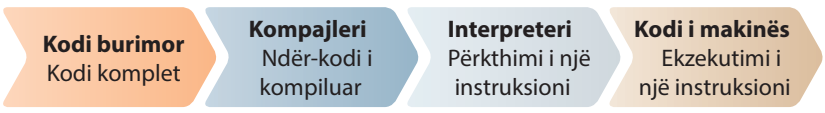


Figura 2.4. Përkthyesi hibrid

Gjuha programuese Java është gjuha që sot përdoret më shpesh, që zbaton mënyrën hibride për përkthimin e kodit burimor në kodin ekzekutiv.

2.5. Gjuha programuese Java

Zhvillimi i gjuhës programuese Java është i ndërlidhur ngushtë me nevojën që në internet të kenë qasje një numër i gjerë përdoruesish. Prandaj në vazhdim jepet një pamje e përgjithshme e shkurtër e krijimit dhe zhvillimit të Internetit, me qëllim që të kuptoni më mirë nevojën që ka kushtëzuar zhvillimin e gjuhës programuese Java.

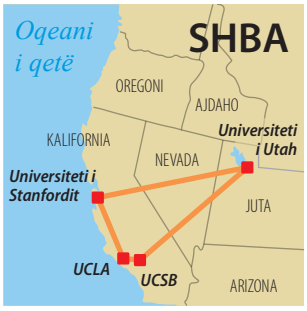


Figura 2.5. ARPAnet në vitin 1969.

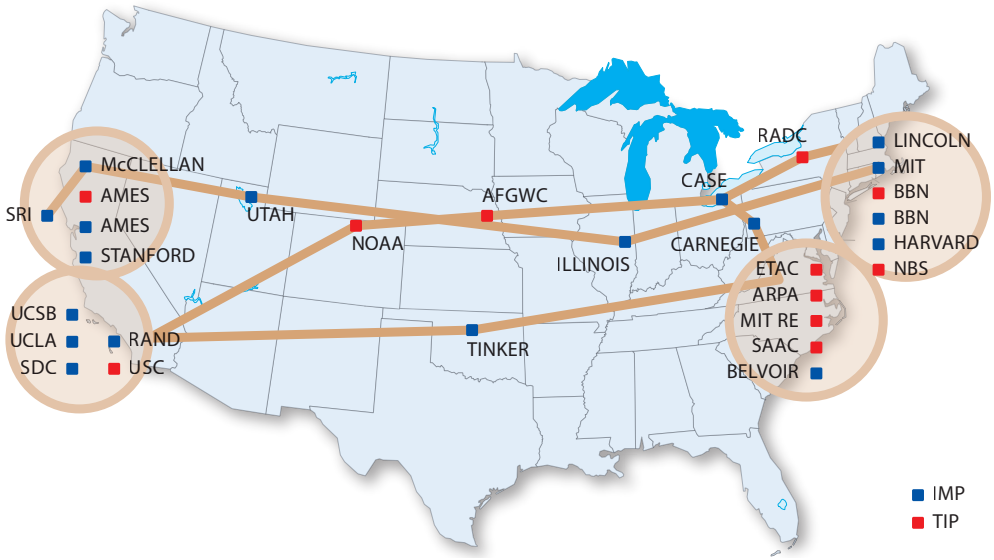


Figura 2.6. ARPAnet në vitin 1974.

Ministria e mbrojtjes e Shteteve të Bashkuara të Amerikës (SHBA) në vitin 1968 ka filluar projektin e krijimit të rrjetit kompjuterik që do të ndërlidhte kompjuterët në lokacione të ndryshme nëpër tërë SHBA me qëllim shkëmbimin e të dhënave. Projekti i është besuar organizatës *Advanced Research Projects Agency* (*ARPA*), dhe është krijuar rrjeti kompjuterik ***ARPAnet***. Në fillim, ARPAnet-i ka bërë lidhjen e katër universiteteve, kurse pak më vonë në vitin 1972, rrjeti ka ndërlidhur njëzet institucione akademike.

Komercializimi i rrjetit ARPAnet ka filluar në vitin 1986, kur për mbrojtjen e konfidencialitetit të të dhënave, është vendosur që të gjithë kompjuterët ushtarakë të zhvendosen në një nënrrjet të veçantë, kurse rrjeti ARPAnet të modernizohet dhe të hapet për mundësi qasjeje të gjitha kompanive dhe individëve.

Interneti është bërë rrjeti publik global në dispozicion të të gjithëve, në të ndodhen me miliona rrjete shtëpiake, akademike, biznesi dhe qeveritare, të cilat midis tyre shkëmbejnë të dhënat dhe shërbimet. Ekspertët nga fusha të ndryshme (ekonomistët, juristët, doktorët...) që nuk janë profesionistë nga fusha e ICT, e përdorin shumë shpesh internetin, mirëpo puna e tyre është e ngadalësuar për shkak të njohurisë modeste të shënimit të komandave dhe përdorimit të programeve për klientët.

Me zhvillimin e *World Wide Web*-it, shërbimit më popullor të internetit, në mënyrë të dukshme janë përmirësuar karakteristikat e Internetit, me çfarë është mundësuar rritja e tij dhe përdorimi i tij i pabesueshëm.

Ideja nga e cila është zhvilluar *World Wide Web* ka lindur në vitin 1989, kur interneti nuk është përdorur në përmasa të gjera. Tim Berners Li nga laboratori Evropian për fizikën nukleare në Zvicër CERN (frëngjisht *Conseil European pour la Recherche Nucleaire*) ka meritat kryesore për

Interneti është bashkësi programesh që komunikojnë me protokollin TC/IP.

Rrjeta botërore (anglisht World Wide Web, W3, WWW) ose thjeshtë ***ueb*** është bashkësia e dokumenteve në formë hiperstrukture të ndërlidhura midis vete, që ndodhen në internet.

Tim Berners Li
krijuesi i World Wide Web-it, i lindur me 8 qershor të citit 1955 në Londër, Britania e Madhe.

vendosjen e koncepteve themelore të *World Wide Web*-a (WWW). Ai ka propozuar formimin e sistemit të hiperstrukturave që do të mundësonte shkëmbimin më të thjeshtë të informacioneve midis përdoruesve të internetit.

Programi i parë për qasjen e *World Wide Web*-it është zhvilluar në vitin 1991. Programi ka punuar me mjedisin tekstual. Tashmë në vitin 1992 kanë ekzistuar pesëdhjetë programe për WWW, dhe disa nga ato kanë pasur mjedisin grafik të zhvilluar, të ngjashëm me mjedisin e sotëm të Windowsit. Qendra CERN, në mesin e vitit 1994 i ka besuar zhvillimin e projektit të WWW organizatës W3, të cilën e kanë themeluar bashkërisht me MIT (anglisht: *Massachusetts Institute of Technology*). Kjo organizatë kujdeset sot mbi të gjitha aspektet e zhvillimit të uebit: nga protokolli deri te gjuha për formimin e Ueb dokumenteve, por edhe për qëllimet e zhvillimit të Uebit në të ardhmen.

2.5.1. Shfaqja e Javës

Në fillim të viteve nëntëdhjetë të shekullit të kaluar, programimi në objekte në gjuhën C++ ka qenë në majën e popullaritetit. Është dukur sikurse programuesit kanë gjetur "gjuhë të përsosur" që ka ofruar një efikasitet të posaçëm dhe ka mundur të përdoret për krijimin e programeve me qëllime të ndryshme. Mirëpo, si edhe shumë herë më parë në të kaluarën, është paraqitur nevoja për zhvillimin e një gjuhe tjetër të programimit (kompleksiteti i programimit në gjuhën programuese C++, nevoja për interoperabilititet (ndërveprim), etj.). Zhvillim i internetit, dhe në veçanti i *World Wide Web*-it ka kushtëzuar nevojën për gjuhën e programimit me qëllim të veçantë, e cila internetin do ta bëjë më dinamik dhe më të përdorshëm. Kërkesat për rolin më dinamik të përdoruesit, që janë manifestuar në nevojën që përmbajtja e faqes së internetit të krijohet dhe vendoset nga vetë përdoruesi, kanë kushtëzuar zhvillimin e gjuhës së re të programimit të specializuar për zhvillimin e aplikacioneve, që do të kryhet brenda vetë programit kërkimor (anglisht *Web Browser*). Në këtë mënyrë mund të shkruhen programet që do të avancojnë dukshëm mundësinë e aplikacionit Web.

Shfaqja e parë e Javës lidhet me vitin 1991, kur një grup programuesish nga kompania *Sun Microsystem* ka filluar projektin për zhvillimin e gjuhës programuese për programimin e mikroprocesorit, që ndodhen në lloje të ndryshme të aparateve elektronike. Zbatimin e parë praktik "paraardhësi i Javës" e ka pasur në aparatet elektronike personale PDA (anglisht *Personal Digital Assistant*), d.m.th. aparateve multifunksionale të cilat në vete kanë pasur funksionet e planerit elektronik, librit të adresave, kalkulatorit, këmbyesit të valutave dhe shumë të tjera

2.5.2. Karakteristikat e Javës

Karakteristikat kryesore të **gjuhës së programimit Java** janë: thjeshtësia, orientimi në objekte, distribucioni, transferueshmëria, siguria, besnikëria, programimi paralel, dinamizmi.

Thjeshtësia e gjuhës së programit Java reflektohet në faktin se programuesit mund ta përvetësojnë lehtë, sepse përmban një bashkësi të komandave të reduktuara dukshëm (në raport me, p.sh. C++). Java sipas aspektit sintaksor është shumë e ngjashme me gjuhët më të hershme të programimit (Pascal, C, C++), çfarë për programuesit me eksperiencë paraqet një lehtësi të dukshme për praninë e saj.

Numri i programuesve në Javë në vitin 2003 ka qenë në intervalin 1500000 dhe 3000000. Në vitin 2007, kur Java është bërë open souce (softuer i lirë), ai numër është rritur në 6000000. Evans Data Corporation në raportin e vetë për vitin 2009 nxjerr informacionin se Javën e përdorin 9007346 programues.

Xhejms Artur Gozling, krijuesi i gjuhës programuese Java, ka lindur me 19 maj të vitit 1955, në Kalgari të Kanadës.

Gjuha e programit Java

Java është gjuhë e ***orientuar në objekte*** që nënkupton që programuesi mund të fokusohet në të dhënat dhe metodat nëpërmjet të cilave do të kryej një punë, dhe mos të ketë kujdes mbi faktin se si do t'i shkruajë disa pjesë të programit.

Java i ka të ngulitura të gjitha funksionet themelore për drejtimin e protokollit të rrjetit. Përmban edhe biblioteka të klasave për komunikimin me protokolle të ndryshme, dhe kështu sigurohet ***distribucioni*** në zhvillimin e programit. Falë përkrahjes së rrjetit, programet e shkruara në Java mund të përdorin të dhënat që ndodhen në tërë Internetin, në lloje të ndryshme kompjuterësh dhe në sisteme operative të ndryshme.

Programi i tipit Java mund të ekzekutohet në çdo lloj kompjuteri, pavarësisht nga sistemi operativ, dhe kështu **transferueshmëria** e tij është përmirësuar dukshëm në raport me gjuhët e tjera të programimit.

Siç është përmendur më herët, Java është paraparë të jetë shumë e ***sigurt*** dhe ***besnike***. Kjo, natyrisht nuk domethënë se nuk mund të shkruhet programi pa gabime, mirëpo janë bërë përpjekje të dukshme të mënjanohen situatat e paqarta, me çfarë shënimi i programeve të sigurta është lehtësuar dukshëm. Siguria e programeve të shënuara në Javë është shumë e madhe. *Sun* ka krijuar (dizajnuar) konceptin "nënshkrimet e aplikacioneve në Java", me të cilin është e mundshme të përcillen ndryshimet në kodin në raport me versionin origjinal.

Java është projektuar me qëllim që të ballafaqohet me kërkesat e krijimeve të programeve interaktive të rrjetës. Për një qëllim të tillë, Java përkrah ***programimin paralel*** – programimin i cili mundëson që programi të ekzekutohet në mënyrë të sinkronizuar paralele në më shume procese. Sistemi i Javës ka zgjidhje elegante dhe të plotë për sinkronizimin e më shumë proceseve (ekzekutimin e në të njëjtën kohë të pjesëve të pavarura të të njëjtit program), me çfarë dukshëm përshpejtohet puna e programit.

Java është gjuhë ***dinamike***, në të cilën pa problem mund të shtohen klasat e reja, interfejset dhe paketat, me çfarë përdorimi i saj zgjerohet në mënyrë permanente.

2.5.3. Platforma e Javës

Platforma kompjuterike paraqet kombinimin e mjedisit të harduerit dhe softuerit (i ashtuquajturi softueri i sistemit), në të cilin programi ekzekutohet. Sistemi operativ është komponent kyç i softuerit të sistemit, qëllimi themelor i të cilit është që të drejtojë resurset e harduerit të sistemit kompjuterik dhe që përdoruesit t'i lehtësojë punën në kompjuter. Sistemet operuese më të njohura janë: Windows, Linux, Solaris dhe MacOs.

Platforma Java

Platforma Java dallohet nga shumica e platformave të tjera si platforma e vetme për softuerë e cila niset pavarësisht nga platformat e tjera të harduerit.

Platforma Java përbëhet nga:

- Makina virtuale Java (anglisht: *Java Virtual Machine – Java VM*),
- Java e mjedisit aplikativ të programit (anglisht: *Java Application Programming Interface – Java API*).

Gjuha e programimit Java kërkon mënyrën hibride të përkthimit që nënkupton që kodi burimor fillimisht përkthehet, pastaj interpretohet në makinën virtuale Java.

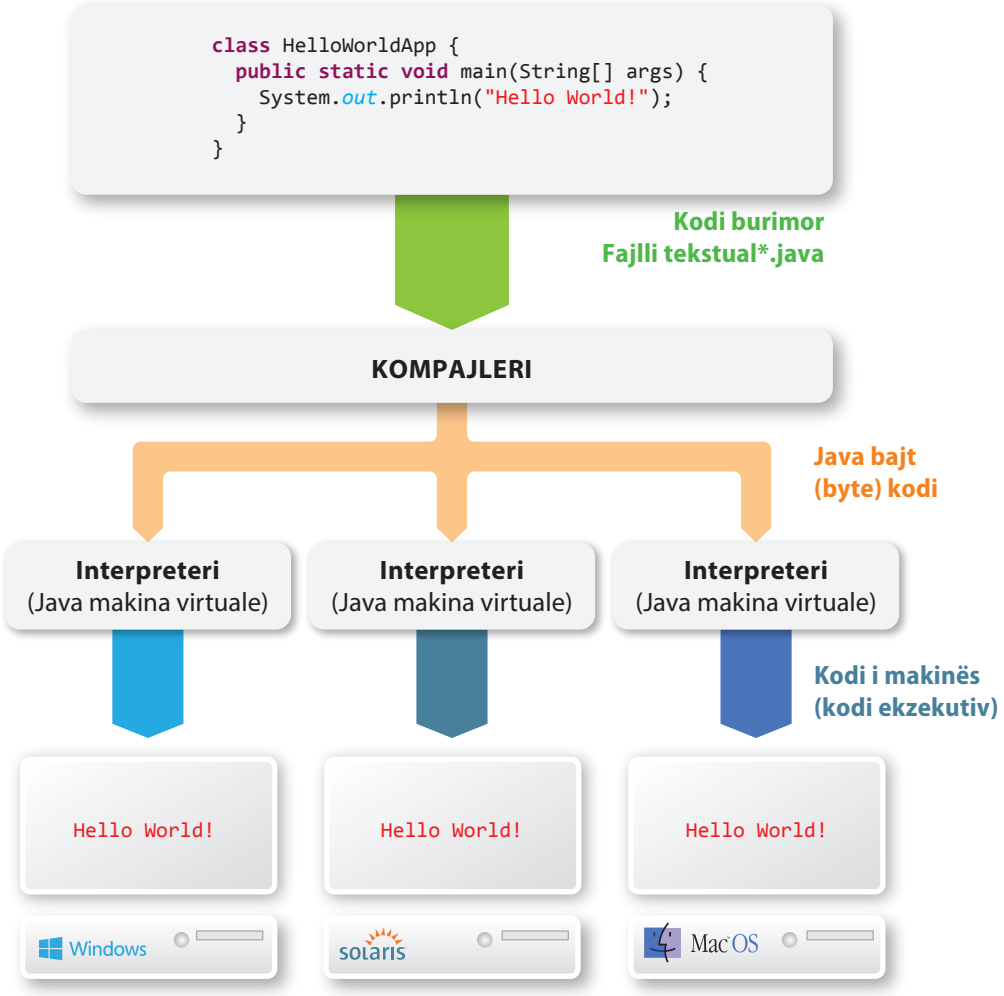


Figura 2.7. Java platforma për sisteme të ndryshme kompjuterike

Kompajleri Java përkthen programin në të ashtuquajturin Java bajt kod (anglisht: *Java byte-code*), i dedikuar Java makinës virtuale (anglisht: *Java virtual machine*). Emërtimi "*makinë virtuale*" rrjedh nga fakti se Java bajt kodi i përftuar nuk ekzekutohet (kryhet) në një kompjuter të vërtetë (një kompjuter konkret), por në një kompjuter të imagjinuar të një platforme të përgjithshme.

Java makina virtuale

Që ***Java bajt kodi*** të ekzekutohet, në kompjuter "të vërtetë" duhet të jetë instaluar interpreteri për Java bajt kodin. Çdo tip i kompjuterit ka interpreterin e vet për Java bajt kodin, i cili mundëson që programi Java i kompiluar mund të ekzekutohet.

Java bajt kodi

Komponentin tjetër të Java platformës e përbën ***Java interfejsi aplikativ i programit (API)*** (anglisht: *Java Application Programming Interface*). Java API paraqet një bashkësi komponentësh softueri që mundëson krijim e programit. Java API është grumbulluar në bibliotekat e klasave të lidhura, të ashtuquajturat paketat (anglisht: *packages*).

Java interfejsi aplikativ i programit

Në vazhdim do t'u njoftojmë me instalimin e Javës dhe mjedisit zhvillimor të Eclipsës, në të cilin do të shkruani programin e parë.

2.5.4. Instalimi i Javës

Instalimi i Javës vjen në dy pakete *Java Runtime Environment* (JRE) dhe *Java Development Kit* (JDK). JRE përmban vetëm funksionalitetet e domosdoshme për nisjen (inicimin) e programit Java, kurse JDK përmban dhe bibliotekat zhvillimore. JDK është i nevojshëm që të mund të zhvillojmë aplikacionet tona dhe t'i kompilojmë.

Instalimi i JDK bëhet në disa hapa. Fillimisht duhet të shkarkohen skedarët instalues, të niset dhe të përcillet rrjedha e instalimit (figura 2.8. dhe 2.9).

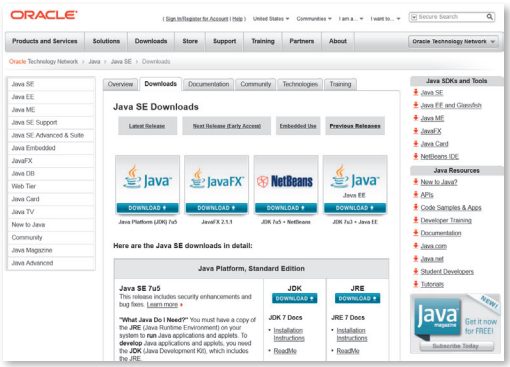


Figura 2.8. Shkarkimi i JDK

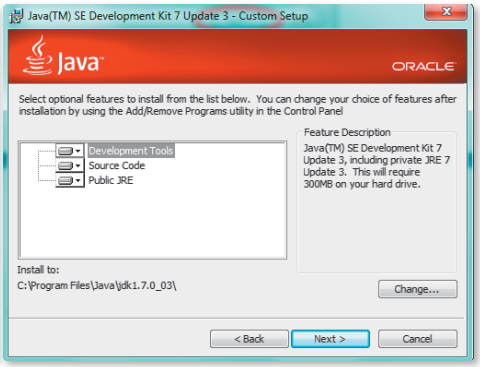


Figura 2.9. Zgjedhja e opsioneve

Versionin e fundit të Java JDK mund të merrni nga ueb sajti zyrtar në adresën: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Hapi tjetër i instalimit është zgjedhja e rrugës (vendit) të direktoriumit instalues (figura 2.10). Rekomandohet vendosja e rrugës së paraqitur.

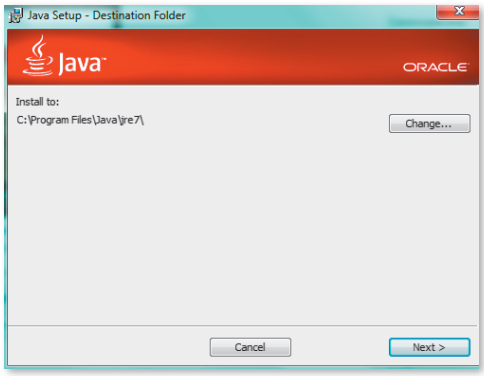


Figura 2.10. Zgjedhja e rrugës për instalim

Figura 2.11. Konfirmimi i instalimit të suksesshëm

Figura 2.11. paraqet fundin e instalimit të suksesshëm, i cili pranohet duke shtypur në butonin *Continue*.

Kodi për verifikimin e versionit aktiv të Javës ndodhet në CD.

Mbas instalimit të Javës, është e mundshme të verifikohet versioni aktiv. Kodi për verifikimin e versionit aktiv të Javës mund ta gjeni në CD-në që vjen si mjet shtesë i këtij libri.

2.5.5. Instalimi i Eclipse

Mjedisi zhvillimor i Eclipse është krijuar nga komuniteti *Open Source* dhe përdoret në disa fusha të ndryshme. Për shembull, përdoret si mjedis zhvillimor për Java aplikacione dhe aplikacionet android. Eclipse mund të zgjerohet me funksione shtesë, d.m.th. me module. Për përdorim të suksesshëm, Eclipse kërkon *Java Runtime* të instaluar, kurse rekomandohen versionet *Java* 7 dhe *Java* 6.

Për nisjen e programit Java duhet të keni Java kompajlerin, mirëpo nëse përdorni Eclipse, nuk u nevojitet kompajleri sepse Eclipse e përmban në vete si komponentë të integruar.

Kur bëhet fjalë mbi instalimin e mjedisit Eclipse, procedura është shumë më e thjeshtë sesa e instalimit të vetë Javës. Nevojitet, si në rastin paraprak, nga sajti zyrtar (http://www.eclipse.org/downloads/), të shkarkohet versioni i fundit i mjedisit Eclipse për sistemin operativ në kompjuterin tuaj (si në figurën 2.12).

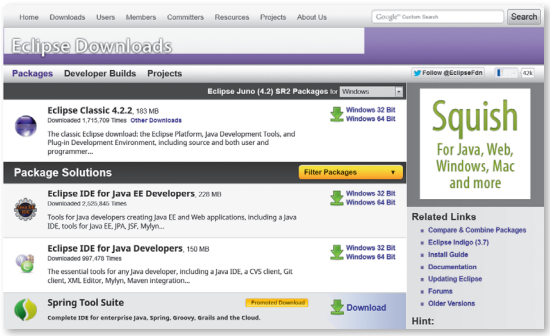


Figura 2.12. Shkarkimi i Eclipses

Mbas shkarkimit të arkivit nga instalimi Eclipse, nevojitet që i njëjti arkiv të shpaketohet në adresën e dëshiruar në kompjuterin tuaj. Në qoftë se përdoret sistemi operativ Windows, rekomandohet C:\. Arkivi i shpaketuar duhet të duket së në figurën 2.13. Me këtë ka përfunduar instalimi i Javës dhe Eclipse.

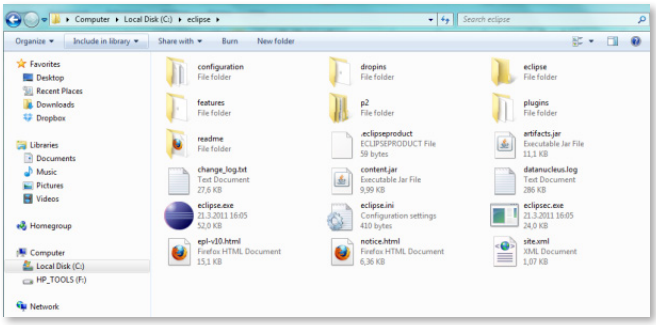


Figura 2.13. Arkivi i shpaketuar me instalimin Eclipse

2.5.6. Programi i parë

Në qoftë se keni instaluar në mënyrë të suksesshme Javën dhe mjedisin Eclipse, mund të krijoni programin tuaj të parë. Që të filloni të përdorni Eclipse, qasuni direktoriumit instalues dhe nisni ikonën ***eclipse.exe***.

Mbas nisjes, sistemi kërkon që të zgjedhni direktoriumin punues (anglisht *workspace*) në të cilin dëshironi të ruani projektet tuaja; zgjidhni rrugën (trajektoren) e dëshiruar në diskun tuaj (p.sh. C:/workspace) dhe shtypni butonin OK.

Nocioni "workspace" shënon dosjen (folderin) në kompjuter në të cilin ndodhen të gjitha Eclipse projektet dhe Eclipse skedarët e tua. Projektet sipas defaultit (parazgjedhjes) krijohen në dosjen workspace, mirëpo përdoruesi mund të zgjedhë në mënyrë të çfarëdoshme vendin në të cilin do të ndodhet projekti i tij. Gjithashtu mund të ekzistojë edhe një numër më i madh i workspace-ve në të cilët ndodhen projektet e përdoruesit, të grumbulluar sipas dëshirës nga ana e përdoruesit.

Mbas nisjes (inicializimit) të parë të Eclipse do t'u paraqitet faqja *welcome* (figura 2.14).

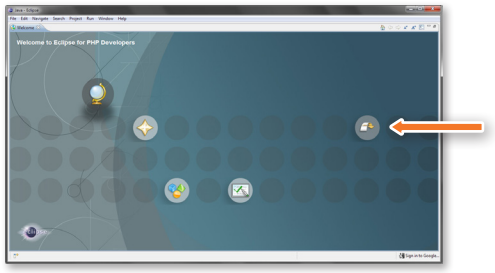


Figura 2.14. Eclipse – faqja fillestare

Duke shtypur në butonin ***Switch to workspace*** (shigjeta në anën e djathtë) transferoheni në dritaren punuese (figura 2.15.) dhe këtu përfundon nisja juaj e parë e mjedisit Eclipse.

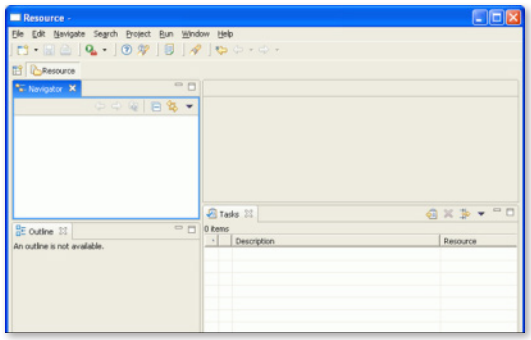


Figura 2.15. Eclipse – workspace

Tradicionalisht, programi i parë me të cilin hyjmë në botën e programimit, e shënon tekstin "*Hello World*".

Nga menyja Eclipse duke zgjedhur opsionin ***File > New > Java project*** hapet programi për krijimin e projektit, si në figurën 2.16. Në fushën e shënuar jepet emri i projektit (emri merret sipas dëshirës), dhe fill mbas duke shtypur butonin ***Finish*** krijohet projekti.

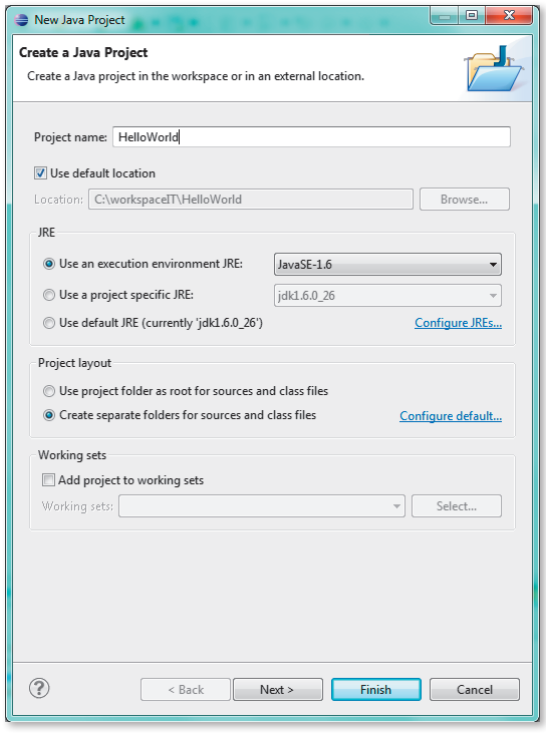


Figura 2.16. Krijimi i Java projektit

Mbas krijimit të projektit, në anën e majtë të mjedisit Eclipse do të paraqitet struktura e projekteve të porsakrijuara. Duke shtypur në "+", pranë emrit të projektit, hapet mundësia për krijimin e klasave të reja, interfejsave, skedarëve etj. brenda projektit (figura 2.17).

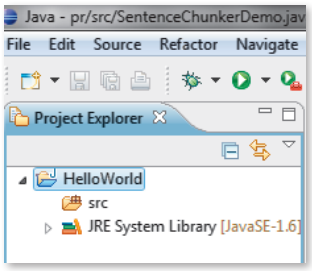


Figura 2.17. Menyja e navigacionit

Për programin e parë nevojitet të krijohet klasa ***Main*** me metodën ***main*** (do të sqarohet më vonë, tani e paramendoni si lloj procedure).

Në strukturën e projektit ndodhet paketi me emrin ***src***. Duke shtypur butonin e djathtë të mausit në paqetin *src* përftohet menyja, nga e cila duhet të zgjidhet ***New > Class*** (figura 2.18), me çfarë hapet dritarja për krijimin e klasës.

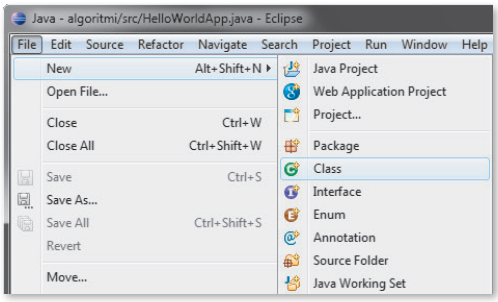


Figura 2.18.

Në fushën e shënuar nevojitet të përkufizohet emri i klasës (në rastin tonë krijohet klasa *Main*), të shënohet artikulli ***public static void main (String [] args)*** sepse secili Java aplikacion duhet të përmbajë metodën main. Duke shtypur butonin ***Finish*** përfundohet krijimi i klasës (figura 2.19).

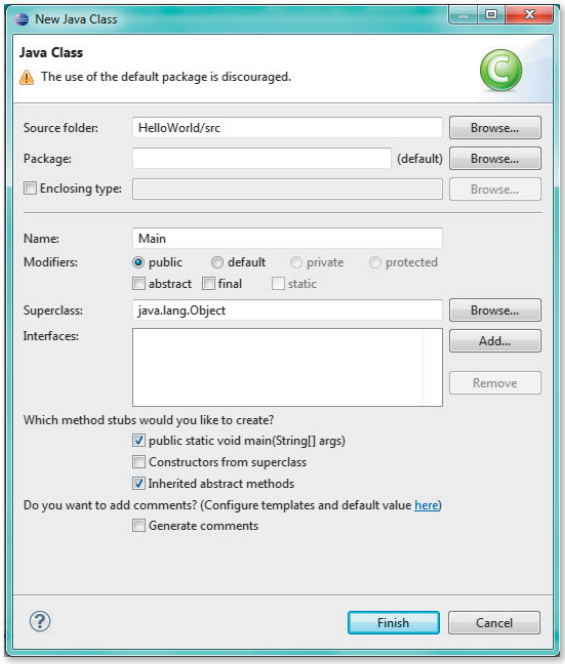


Figura 2.19. Krijimi i klasës

Hapi vijues është që në Editorin për klasën Main shënohet kodi i mëposhtëm për metodën main. Editori hapet me klik të dyfishtë të butonit të majtë të mausit në klasën Main në strukturën e projektit.

```
class Main {
  public static void main(String[] args) {
   System.out.println("Hello World!");
  }
}
```

Projekti komplet ndodhet në dosjen (folderin) ProframiParë në CD.

Mbas hyrjes së kodit, mund të nisni programin tuaj të parë në këtë mënyrë: Nga menyja e navigacionit zgjidhni ikonën ose nëpërmjet të klikut të djathtë të mausit në klasën *Main* dhe zgjedhjen ***Run As > Java Application*** (figura 2.20).

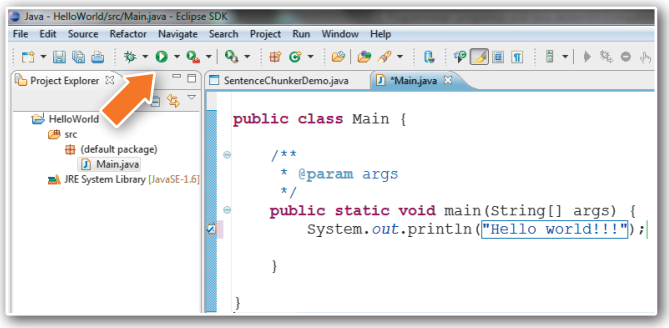


Figura 2.20. Nisja e programit

Në fund, në konsolën e mjedisit Eklipse paraqitet rezultati i ekzekutimit të programit të nisur, si në figurën 2.21.

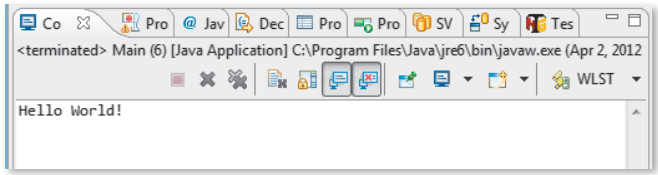


Figura 2.21. Ecklipse konsola dhe paraqitja e rezultatit

Ju e keni shkruar dhe nisur me sukses programin tuaj të parë

2.6. Java aplikacionet dhe apletët

Gjuha programuese Java ofron përkrahje për zhvillimin e softuerit për qëllime të përgjithshme dhe të veçanta.

Me ndihmën e kësaj gjuhe programuese shkruhen softuerët për kompjuterët e mëdhenj (anglisht: *main frame*), serverë, kompjuterë personalë, aparate speciale për telekomunikacion, televizionin interaktiv, marrësit satelitorë, aparatet PDA, për telefonat celularë, aparate shtëpiake (makina larëse, pajisje me mikrovalë, frigoriferë dhe shumë të tjera).

Dy format themelore të Java programit janë ***aplikacionet*** (anglisht: *application*) dhe ***apletet*** (anglisht: *applet*). Sot Java përdoret më shumë për zhvillimin e ueb aplikacioneve dhe aplikacione për telefona celularë.

Figura 2.22. Java platforma për kategoritë e ndryshme të tregut

2.6.1. Java aplikacionet

Java Aplikacioni është program i pavarur që ekzekutohet drejtpërdrejt në platformën Java. Java aplikacionet mund të ekzekutohen në platforma të ndryshme kompjuterike (PC, Mac, kompjuterë të mëdhenj, etj.) edhe pajisje elektronike (telefona, pajisje PDA, makina larjeje, furra, frigoriferë, kamera, etj.).

Java aplikacionet

Shënimi i Java aplikacioneve përbëhet nga hapat e mëposhtëm:

- shënimi i kodit burimor,
- kompilimi i kodit burimor në bajt kod me ndihmën e kompajlerit,
- nisja e aplikacioneve me ndihmën e interpreterëve të bajt kodit.

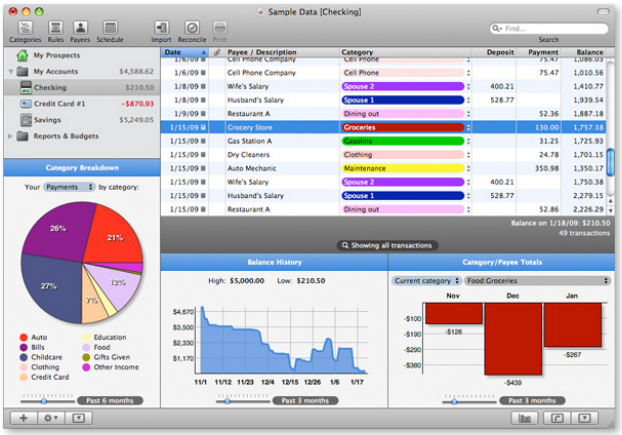


Figura 2.23. Java aplikacioni

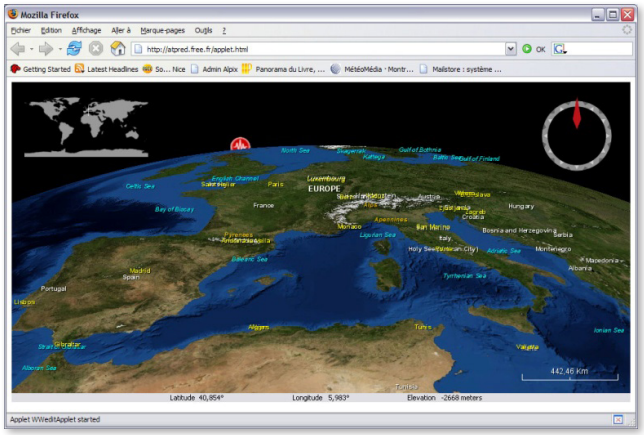


Figura 2.24. Java apleti

2.6.2. Java apletët

Java apletët janë programe të shkruara me qëllim të zmadhimit të komponentit intelektual të Ueb aplikacioneve. Java apletët ekzekutohen brenda ueb faqes. Prandaj për punën e tyre është i domosdoshëm programi ueb kërkues. Duke pasur parasysh se apletët kryhen brenda ueb aplikacioneve, shënimi i tyre është lehtësuar dukshëm në raport me shënimin e aplikacioneve, sepse mund të mbështeten në shërbimet e kërkuesve (printimit të grafikës, hapjes së dritareve të reja).

Java apletët

Jeta e apletit përbëhet nga pesë fazat themelore:

- inicializimi,
- nisja,
- shënimi (printimi) në ekran,
- ndërprerja,
- shkatërrimi.

Faza e *inicializimit* është faza e parë e jetës së një apleti. Në atë fazë apleti hyn në programin kërkues. Inicializimi i apletit kryhet vetëm një herë gjatë ekzekutimit të programit.

Faza e *nisjes* del në skenë atëherë kur programi kërkues nis me ekzekutimin e apletit. Nisja ndodh atëherë kur përdoruesi herën e parë i qaset ueb faqes në të cilën ndodhet apleti dhe secilën herë tjetër kur përdoruesi kthehet në faqe.

Faza e *printimit në ekran* fillon mbas nisjes së apletit, kur apleti dëshiron të shkruajë ose vizatojë diçka në dritaren e programit kërkues.

Fazom *ndërprerjes* është e kundërt me fazën e nisjes dhe ndodh atëherë kur përdoruesi kalon në një faqe tjetër. Në këtë fazë ndërpritet të duket në ekran.

Me fazën e *shkatërrimit* ndalohet ekzekutimi i apletit. Kjo fazë ndodh në momentin kur programi kërkues përfundon me punën. Faza e shkatërrimit, ngjashëm si dhe faza e inicializimit ndodh vetëm njëherë gjatë jetës së apletit.

Në internet ndodhet një numër i madh i Java apletëve, kurse më vonë do të takohemi me mënyrën e shënimit të tyre. Të fillojmë sipas radhës, nga elementet themelore të gjuhës së programit.

Apletët interesantë nga fusha e fizikës mund t'i gjeni në ueb faqen: http://www.walter-fendt.de/ph14yu/

Pyetje dhe detyra kontrolli

1. Sqaroji nocionet: sistemi kompjuterik, hardueri kompjuterik dhe softueri kompjuterik.
2. Çfarë është programimi?
3. Çfarë janë gjuhët e programimit?
4. Si ndahen gjuhët e programit në bazë të qëllimit të tyre?
5. Sa gjenerata gjuhësh programuese ekzistojnë?
6. Cilat janë karakteristikat e gjeneratës së dytë të gjuhëve programuese?
7. Jepi karakteristikat e programimit në gjuhën e makinës.
8. Cili është dallimi midis gjuhës së makinës dhe asemblerit?
9. Sqaroji karakteristikat e gjuhëve programuese të nivelit të lartë.
10. Sqaro nocionin e paradigmës programore.
11. Sqaro dallimin midis paradigmës procedurale dhe paradigmës së orientuar në objekte.
12. Numëro fazat e procesit të kompilimit.
13. Sqaro dallimin midis kompajlerëve dhe interpreterëve.
14. Çfarë është ARPAnet?
15. Sqaro veçoritë themelore të gjuhës programuese Java.
16. Sqaro se prej çfarë përbëhet Java platforma.
17. Sqaro mënyrën se si programi i shënuar në Java përkthehet dhe ekzekutohet.
18. A duhet që Java aplikacionet të ekzekutohen në ueb apo jo?
19. A duhet të ekzekutohen në kompjuterët personalë programet e shkruara në Java?
20. Thuaji hapat në procesin e shënimit të Java programit.
21. Çfarë janë java apletët?
22. Thuaji fazat e ciklit të jetës së Java apletit.
23. Sa herë bëhet inicializimi i apletit?

III.

ELEMENTET THEMELORE TË GJUHËS PROGRAMUESE JAVA

```
public void updateGraphNodeDistance(GraphNode n) {
    this.pQueue.remove(n);
    this.pQueue.add(n);
}

public void PrintContents() {
    ArrayList<GraphNode> _temp = new ArrayList<GraphNode>();
    System.out.println("Size of Q=" + pQueue.size());
    while (!pQueue.isEmpty()){
        GraphNode n =pQueue.remove();
        _temp.add(n);
        System.out.println(n.getVal()+" distance=" + n.getDistance());
    }
    pQueue.addAll(_temp);
}
```

Në këtë kapitull janë përshkruar elementet themelore të gjuhës programuese Java. Ky kapitull dhe kapitujt e ardhshëm duhet të konsiderohen së bashku, sepse janë lidhur ngushtë. Në to flitet mbi mënyrën e shënimit të programit, shumë gjëra të lejueshme dhe të palejueshme, praktikën e mirë dhe të keqe, vështirësitë që mund të paraqiten gjatë fazës fillestare të programimit në Java.

Me përvetësimin e suksesshëm të materialit të përfshirë në këtë kapitull do të aftësoheni që:

- të përdorni komentet në program me qëllim sqarimi shtesë të njësive të veçanta të kodit,
- të përdorni literalet, separatorët (ndarësit) dhe fjalët kyçe,
- të njihni funksionin dhe domethënien e llojeve komplekse themelore të të dhënave,
- të përdorni veprimet aritmetike, veprimet logjike, operatorët e krahasimit dhe operatorët mbi bite,
- të deklaroni dhe inicializoni ndryshoret,
- të shkruani një program të thjeshtë.

Java është gjuhë e Internetit, kurse Interneti i përdor personat që flasin dhe shkruajnë në gjuhë të ndryshme. Prandaj është vendosur që Java duhet të përkrahë bashkësinë e simboleve që përdoren në të gjitha gjuhët botërore. Në pajtim me këtë, Java përdor bashkësinë e simboleve Unicode që janë superstrukturë e bashkësisë tradicionale të simboleve ASCII: Për dallim nga bashkësia ASCII, e cila për kodimin e simboleve përdor 8 bite dhe mund të paraqesë vetëm 256 simbole, Unicode përdor 16 bite për paraqitjen e simboleve, prandaj mund të paraqesë 65536 simbole, me çfarë janë përfshirë të gjitha simbolet e gjuhëve të rëndësishme botërore. Kjo në praktikë do të thotë që mund të përdorni fjalët nga gjuha shqipe gjatë programimit.

Elementet themelore të gjuhës programuese Java janë:

- komentet,
- literalët dhe separatorët,
- fjalët e rezervuara,
- tipat e ndryshoreve,
- operatorët,
- ndryshore.

Komapjleri Java merr të gjitha rendet e kodit burimor dhe nga ai i heq të gjitha hapësirat dhe kalimet në rreshtin e ri. Meqenëse këto elemente hiqen nga kodi burimor, në program mund të largoni komandat me numër të çfarëdoshëm të hapësirash (space), tabesh dhe/ose me rreshta të rinj, me qëllim rregullimi të kodit burimor që të jetë sa më i lexueshëm dhe i qartë.

Tabela 3.1. ASCII kod

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SCH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

3.1. Komentet

Komentet

Komentet janë pjesë të kodit burimor që nuk përkthehen në kodin ekzekutiv, por programuesit i shërbejnë që të përcjellë më mirë (dokumentojë) mënyrën e punës së disa pjesëve të programit.

Është praktikë e mirë përdorimi i komenteve brenda kodit programues, sepse në atë mënyrë në hollësi përshkruhet domethënia e një klase, metode ose pjese kodi. Duke shkruar komentet, lehtësohet dukshëm çdo superstrukturë e mëtejshme, shtim ose ndryshim kodi.

```
/* Kjo klasë paraqet teoremën e Pitagorës.
   Treshi i numrave të Pitagorës plotëson ekuacionin
   e teoremës së Pitagorës a^2 + b^2 = c^2 */

class Pitagora {
…
```

Komentet në Javë mund të shënohen në tri mënyra. Mënyra standarde përdor simbolet /* dhe */ dhe teksti që ndodhet brenda këtyre simboleve konsiderohet si koment, pa marrë parasysh se në sa rreshta është shënuar komenti.

Nuk guxojmë të grumbullojmë komentet, d.m.th. brenda komentit të shënuar me simbolet /* dhe */ nuk guxojmë të hapim komentin e ri. Prandaj komenti i mëposhtëm është jokorrekt.

Kompajleri i kapërcen plotësisht komentet, prandaj është mendim i gabuar dhe i pa themeluar se komentet e ngadalësojnë ekzekutimin e programit. Megjithatë, komentet te gjuhët programuese interpretuese dhe të vjetra mund të shkaktojnë ngadalësimin e programit gjatë ekzekutimit.

```
/* Ky koment që brenda vete ka
/* edhe një koment
   është shënuar në mënyrë jokorrekte */ */
```

Mënyra tjetër i referohet shënimit të komenteve në një rresht duke përdorur simbolin //. Teksti që vazhdon mbas simbolit // në atë rresht deri në fund konsiderohet si koment.

```
if (ime == null) {    // në qoftë se përdoruesi nuk ka shënuar emrin
  System.out.println ("Shëno emrin"); // shënoji mesazhet e paralajmërimit
…
```

Mënyra e tretë e shënimit të komenteve lidhet me mjetin e dedikuar për krijimin e dokumentacionit që është pjesë e Java Development Kit-it. Teksti që ndodhet midis simboleve /** dhe */ konsiderohet si koment dhe nuk përfillet gjatë kompilimit, mirëpo gjeneratori i dokumentacionit i kyç ato komente gjatë krijimit automatik të dokumentacionit. Simboli /** paraqet fillim e komentit, kurse simboli */ përfundimin e komentit.

```
/************************************************
 Ky është komenti që kyçet gjatë krijimi automatik
 të dokumentacionit. */
```

3.2. Literalët

Literalët

Literalët janë vlera konstante që paraqiten në programe dhe nënkuptojnë ruajtjen e një vlere të cekur. Në qoftë se jepet numri 5, në Javë ai është numri i plotë 5, kurse nëse jepet simboli '*e*', për Javën ai është simboli që përmban vlerën e shkronjës *e*.

Literalët mund të jenë numra të plotë, numra decimalë, vlera logjike, simbole ose stringje (vargje simbolesh). Për secilin nga ato është e përkufizuar mënyra e shënimit. Shembuj literalesh janë:

```
100                 // Numër i plotë
12.6                // Numër decimal
true                // Vlerë logjike
'E'                 // Shenja
"Podgorica"         // Varg simbolesh (String)
```

3.2.1. Numrat

Numrat

Java përkrah dy lloje numrash: numrat e plotë dhe numrat decimalë (numrat me presje lëvizëse). Shënimi i këtyre llojeve të numrave është shumë i ngjashëm me shënimin në gjuhët e tjera programuese, mirëpo megjithatë ekzistojnë disa dallime në mënyrën se si ata numra ruhen në memorien e kompjuterit gjatë ekzekutimit të programit.

Numrat e plotë mund të shënohen në sistemin numerik dhjetor, heksadecimal ose oktal. Numrat e shënuar në sistemin dhjetor shënohen pa zeron udhëheqëse. Për paraqitjen e numrave të plotë përdoren 8, 16, 32 dhe 64 bite, çfarë do të sqarohet më vonë.

Numrat decimalë paraqiten nëpërmjet pesë elementeve: pjesës së plotë, pikës decimale, pjesës decimale, simbolit e (ose E) dhe eksponentit të numrit 10.

3.2.2. Vlerat logjike

Vlerat logjike shpeshherë quhen vlerat e Bulit. Ekzistojnë vetëm dy vlera logjike në Javë: **`true`** (e saktë, e vërtetë) dhe **`false`** (e pasaktë, rrenë). Ato më shpesh përdoren për prurjen e vendimeve mbi kryerjen e një pjese të caktuar të kodit të programit.

3.2.3. Shenjat dhe stringjet

Shenja është cilido simbol që ndodhet brenda thonjëzave të njëfishta. Mund të flitet mbi cilëndo shenjë nga bashkësia Unicode. Vini re se shenja i konsiderojmë edhe shifrat dhe të gjitha shenjat speciale që janë cekur brenda thonjëzave të njëfishta.

Stringje konsiderohen të gjitha vargjet e shenjave të shënuara brenda thonjëzave të dyfishta. Stringjet mund të përbëjnë edhe hapësira (space).

Shembull numrash të plotë të paraqitur në sistemin numerik dhjetor:

-10
15
25
1278
123456789

Shembull numrash decimalë të paraqitur në sistemin numerik dhjetor:

126,32
12.3456e4 znači 123456.0
12.3445E6 znači 12344500.0
.316
2.

Shembuj vlerash logjike:

```
true
false
```

Shembuj simbolesh:

'a'
'B'
'9'
'!'

Shembuj stringjesh:

"Shenja"
"Programimi në Java"

3.3. Separatorët

Separatorët janë shenja që përdoren për grumbullimin dhe rregullimin e kodit të programit. Separatorët dhe domethëniet e tyre janë cekur në tabelën 3.2, kurse zbatimi i tyre do të sqarohet me shembuj në vazhdim të këtij Teksti.

Tabela 3.2. Paraqitja e separatorëve në gjuhën programuese Java

Separatori	Domethënia
{ }	Kllapat e mëdha përdoren për shënimin e bllokut të programit dhe grumbullimin e komandave që i takojnë po të njëjtit bllok.
[ ]	Kllapat e mesme përdoren për përkufizimin e vargut dhe qasjen e kufizave të tij.
( )	Kllapat e vogla përdoren për përkufizimin e parametrave të metodës, cekjes së parametrave hyrës së metodës dhe përkufizimin e kushteve në komanda drejtuese.
;	Pikëpresja përdoret në fund të çdo shprehjeje.
.	Pika përdoret për citimin e emrit të pakos, klasës dhe metodës në to.
,	Presjet përdoren për ndarjen e literaleve, ndryshoreve dhe argumenteve të metodës.

Separatorët

3.4. Fjalët kyçe

Fjalët kyçe janë fjalë të përkufizuara më herët të cilat i përdorë gjuha programuese. Më shpesh bëhet fjalë mbi komandat që në kodin e programit përdoren për respektimin e rregullave sintaksore dhe semantike. Nuk guxohet të përdoren fjalët kyçe për emrat e ndryshoreve. Në të kundërtën, kompajleri nuk do të dallojë ndryshoret nga fjalët kyçe (komandat), çfarë do të gjenerojë gabim në procesin e kompilimit.

Fjalët kyçe

Meqenëse numri i fjalëve kyçe është i madh, mund të ndodhë që si rezultat ngatërrimi, disa nga ato të përdoren si emër i një objekti. Prandaj duhet që para se të shoqërohen emrat objekteve, të kryhet verifikimi dhe të konstatohet se njëri prej emrave të dhënë nuk është njëra prej fjalëve kuçe. Fjalët kyçe shënohen me shkronja të vogla, në gjuhën angleze dhe pikërisht të tilla duhet të përdoren. Në tabelën 3.3. janë paraqitur fjalët kyçe që përdoren më shpesht.

Tabela 3.3. Fjalët kyçe që përdoren më shpesh në Javë

abstract	const	float	int	protected	threadsafe
boolean	continue	for	interface	public	throw
break	default	future	long	rest	throws
byte	do	generic	native	return	transient
byvalue	double	goto	new	short	true
case	else	if	null	static	try
cast	extends	inner	operator	super	var
cacth	false	implements	outer	switch	void
char	final	import	package	synhronized	volatile
class	finally	instanceof	private	this	while

3.5. Shenjat e lejueshme në emra

Përkufizimi korrekt i emrave

Çdo objekt (ndryshore, metodë, klasë etj.) brenda programit emërtohet dhe njëkohësisht duhet të kihet kujdes për përkufizimin e rregullt të emrave. Emri i objektit, nga aspekti teorik, mund të përmbajë të gjitha shenjat nga bashkësia Unicode, mirëpo ekzistojnë disa përjashtime.

- Emri duhet të fillojë me një shkronjë dhe mund të ketë numër të çfarëdoshëm shenjash.
- Mbas shenjës së parë (shkronjës) mund të pasojnë të gjitha simbolet e tjera: shkronjat, numrat dhe simboli "..." kudo në emër.
- Në shënimin e emrave mund të përdoren edhe shkronjat e mëdha dhe të vogla. Me qëllim që programi të jetë sa më i lexueshëm, rekomandohet që të gjithë emrat të shënohen me shkronja të vogla, kurse nëse emri përbehet nga më shumë fjalë, atëherë me shkronjë të madhe të fillohet shënimi i çdo fjale të re (p.sh. *emriMbiemri*). Ekzistojnë edhe marrëveshje (konventa) të tjera për shënim (p.sh. ndarja e fjalëve me vizën poshtë "_", etj.), mirëpo konventa paraprake është treguar shumë e suksesshme dhe e lexueshme.
- Pasi që objekti (klasa, metoda, ndryshorja etj.) është deklaruar, që t'i qaset në mënyrë të rregullt, duhet që në të gjitha vendet të shënohet emri i tij në mënyrë të saktë.
- Gjuha programuese Java i dallon shkronjat e vogla nga ato të mëdhatë. D.m.th. programimi, Programimi dhe PROGRAMIMI paraqesin tri emërtime të ndryshme.
- Praktika e shoqërimit të emrave të njëjtë objekteve të ndryshme, mirëpo me renditje të ndryshme të shkronjave të vogla dhe të mëdha, duhet të shmanget, sepse

Të rikujtohemi se në algoritma kemi cekur se për ndryshoret nuk ekzistojnë rregullat rigoroze për emërtim, por nevojitet që emrat të kenë domethënie të qartë dhe të mund të shënohen në kuadrin e algoritmit.

në atë mënyrë kodi burimor bëhet i paqartë dhe zmadhohet probabiliteti i paraqitjes së gabimit në program.

Shembuj emrash të përkufizuar mirë dhe keq:

```
a                    // emër i përkufizuar mirë
blerësi              // emër i përkufizuar mirë
emri_nxenesit        // emër i përkufizuar mirë, ndërmjet dy
                     // vargjeve shenjave mund të përdoret shenja _
vjet                 // emër i përkufizuar mirë
numritelefonit       // emër i përkufizuar mirë

import               // keq, emri import është fjalë kyçe
2014_viti            // keq, emri duhet të fillojë me një shkronjë
emri-nxenesit        // keq, emri nuk guxon të përmbajë simbolin -
emri nxenesit        // keq, emri nuk guxon të përmbajë hapësirën (space)
telefoni#            // keq, emri nuk guxon të përmbajë shenjën #
```

3.6. Tipat e të dhënave

Tipat e të dhënave

Tipat e të dhënave përdoren për përcaktimin e llojit të të dhënave, të cilat ndryshorja i ruan. Në ndryshore nuk mund të ruhen të dhënat e tipave të ndryshme në krahasim me atë për të cilën ndryshorja është përkufizuar, as nuk mund të përdoren operatorët që nuk janë përkufizuar për atë tip ndryshoreje.

Java është shumë "rigoroze" në përkufizimin e tipit të ndryshoreve. Çdo ndryshore duhet të ketë llojin (tipin) e vet të të dhënave, çdo shprehje e cekur në program ka gjithashtu tipin e vet të të dhënave, dhe çdo tip i të dhënave është përkufizuar në mënyrë precize. Ekzistojnë rregullat precize për përdorimin e tipave të caktuara të të dhënave. Është mundësuar edhe zbërthimi eksplicit i një tipi të dhënash në tipin tjetër. Çdo shoqërim i vlerës në program nënkupton verifikimin me qëllim që të konstatohet, nëse vlera e shoqëruar i përgjigjet tipit të ndryshores në të cilën do të ruhet apo jo. Tipat e të dhënave që Java i përkrah ndahen në tipa të thjeshtë (primitivë) dhe ata kompleksë.

3.6.1. Tipat e thjeshtë (primitivë) të të dhënave

Tipat e thjeshtë të të dhënave

Tipat e thjeshtë të të dhënave janë ata që nuk mund të ndahen në të dhëna më të vogla, gjegjësisht ata që nuk janë të përbërë nga tipat e tjera të të dhënave. Në Javë janë përkufizuar tetë tipa të thjeshta të dhënave: `byte`, `short`, `int`, `long`, `char`, `float`, `double` dhe `boolean`, që quhen ***tipat primitive të të dhënave***. Mund t'i ndajmë në katër grupe sipas tipit të dhënave që e ruajnë:

- *Tipat për numrat e plotë* (`byte`, `short`, `int`, `long`) përdoren për ruajtjen e numrave të plotë.
- *Tipat për numrat decimalë* (`float`, `double`) përdoren për ruajtjen e numrave realë, të cilët për paraqitjen e tyre përdorin thyesën lëvizëse.
- *Tipi për shenja* (`char`) përdoret për ruajtjen e të dhënave që paraqesin shenjat, siç janë: shkronjat, shifrat, simbolet edhe shenjat e tjera individuale.
- *Tipi logjik* (`boolean`) është tip i veçantë i të dhënave me ndihmën e të cilit mund të paraqiten vlerat `true` (i saktë, i vërtetë) dhe `false` (i pasaktë, i rremë) dhe përdoren në kushte logjike në program.

3.6.1.1. Tipat për numrat e plotë

Katër tipa për numrat e plotë (byte, short, int, long) përdoren për paraqitjen e numrave të plotë pozitivë dhe negativë. Në tabelën 3.4. janë paraqitur tipat e të dhënave të numrave të plotë. Shtylla e dytë e kolonës tregon gjatësinë në bite që përdoret për paraqitjen e tyre, kurse dy shtyllat e tjera tregojnë vlerën më të vogël, gjegjësisht, vlerën më të madhe të lejueshme.

Tabela 3.4. Vetitë e tipave të dhënave me numra të plotë

Tipi	Gjatësia	Vlera më e vogël	Vlera më e madhe
`byte`	8	−128	127
`short`	16	−32 768	32 767
`int`	32	−2 147 483 648	2 147 483 647
`long`	64	−9 223 372 036 854 775 808	9 223 372 036 854 775 807

Tipi byte është tipi më i vogël i të dhënave për numra të plotë. Gjatësia e deklaruar e tij është 8 bite, që mjafton të paraqesim numrat e plotë prej −128 deri te 127. Ky tip të dhënash, shpeshherë përdoret gjatë leximit nga skedari (fajlli) ose dërgimi i të dhënave nëpërmjet rrjetit.

Tipi short paraqitet me 16 bite, çfarë mjafton për ruajtjen e numrave të plotë midis −32768 dhe 32767.

Tipi `int` është tipi që përdoret më shpesh. Fjala është mbi tipin me 32 bite që mund të ruajë vlerat prej −2.147.483.648 deri te 2.147.483.647. Ndryshoret e këtij tipi përdoren më shpesh në programe si numëruesa ciklesh, si indekse etj.

Tipi `long` përdoret për paraqitjen e numrave të gjatë të plotë. Falë gjatësisë së tij prej 64 biteve, mund të paraqesë numrat e mëdhenj deri me 19 shifra. Natyrisht, ky tip është praktik, nëse rezultati që pritet është numër i madh. Për shembull, nëse dëshironi të njehsoni gjatësinë që drita do të kalojë për një vit, kjo e dhënë nuk mund të ruhet në ndryshoret e tipit `int`, por duhet të përdoren ndryshoret e tipit `long`.

3.6.1.2. Numrat decimalë (numrat me presje lëvizëse)

Numrat decimalë përdoren për njehsimin e rrënjës së një numri, pjesëtimin e dy numrave, njehsimin e numri π(Pi), njehsimin e funksioneve logaritmike dhe trigonometrike, etj. Java dallon dy tipa për paraqitjen e numrave decimalë: `float` dhe `double`. Vetitë themelore të këtyre dy tipave janë paraqitur në tabelën 3.5.

Tabela 3.5. Vetitë e tipave të të dhënave për punën me numra decimalë

Tipi	Gjatësia	Vlera më e vogël pozitive	Vlera më e madhe pozitive
`float`	32	2^{-149}	$(2-2^{-23})\cdot 2^{127}$
`double`	64	2^{-1074}	$(2-2^{-52})\cdot 2^{1023}$

Tipi `float` ka gjatësinë 32 bite, dhe për të më shpesh thuhet se ka "precizitetin" e njëfishtë. Rezultatet e disa veprimeve mund të jenë joprecize, kur bëhet fjalë për numra shumë të mëdhenj ose numra shumë të vegjël, që duhet të paraqiten.

Prandaj tipi `float` përdoret kur nevojitet të paraqitet një numër real, por nuk është i rëndësishëm preciziteti i madh i pjesës decimale. Çdo veprim pasues mbi ndryshoret e tipit `float` rezulton në gabimin më të madh të rezultatit.

Tipi `double` ka gjatësinë prej 64 biteve dhe siguron të ashtuquajturin "precizitetin e dyfishtë". Meqenëse përdor dyfish më shumë bite, zmadhohet edhe preciziteti i paraqitjes së numrave. Tipi `double` është zgjedhje e domosdoshme kur kërkojmë vlerën precize të një madhësie.

3.6.1.3. Tipat për shenja

Ndryshoret që duhet të ruajnë një shenjë, deklarohen si shenjat e tipit char. Java përdor bashkësinë ***Unicode*** të shenjave me gjatësi 16 bitesh, prandaj me këtë tip mund të ruhen 65 536 shenja të ndryshme. Vlerat, ndryshoreve të tipit char, mund t'u jepen në thonjëza të njëfishta ose me numra të plotë që paraqesin kodin e shenjës në bashkësinë *Unicode*.

Vlera 65 e Unicode paraqet shkronjën A.
Vlera 97 e Unicode paraqet shkronjën a.

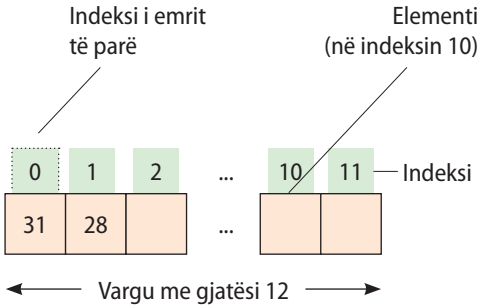
3.6.1.4. Tipi logjik

Tipi logjik boolean përdoret për ruajtjen e vlerave logjike. Vlera logjike mund të jetë `true` (e saktë, e vërtetë) ose `false` (e pasaktë, rrenë). Tipi logjik më shpesh përdoret për kontrollin e rrjedhës.

3.6.2. Tipat komplekse të të dhënave

Tipat komplekse të të dhënave ndërtohen me ndihmën e tipave të thjeshtë ose atyre kompleksë. Shembuj tipash kompleksë janë *vargjet* dhe *stringjet*.

Tipat komplekse të të dhënave



Vargjet përmbajnë një sasi më të madhe të dhënash të një tipi themelor ose kompleks. Çdo kufizë e vargut përmban vlerën e vet të tipit të përkufizuar. Elementit të vargut i qaset duke cekur indeksin që përcakton pozicionin e tij në varg.

Për shembull, me qëllim që të përkufizoni vargun në të cilin për çdo muaj të vitit do të ruani numrin e ditëve të tij, shkruani kodin:

```
int[] ditën_në_muaj = new int[12];
ditën_në_muaj[0] = 31;
ditën_në_muaj[1] = 28;
...
```

Në Javë, *stringu* paraqitet si klasë e veçantë (që quhet *String*), për dallim nga gjuhët e tjera programuese, në të cilat paraqitet si varg shenjash.

Deklarimi i ndryshores të tipit string bëhet në mënyrën e mëposhtme:

```
String emri_nxenesit = "Marku";
```

Mbi vargjet dhe stringjet do flitet më shumë në kapitujt e mëposhtëm.

3.7. Ndryshoret

Ndryshoret

Ndryshoret shërbejnë për ruajtjen e vlerave të parametrave, ndër-rezultateve dhe rezultateve. ***Çdo ndryshore duhet të jetë e deklaruar nëpërmjet emrit dhe tipit të dhënave që ruan.*** Rregullat që zbatohen te objektet përshijnë edhe ndryshoret. Gjatë shoqërimit të emrit ndryshores, duhet të kihet kujdes mbi çështjet e mëposhtme:

- emri i ndryshores duhet të fillojë me shkronjë,
- emri nuk guxon të jetë fjalë kyçe ose e rezervuar,
- emri i ndryshores duhet të jetë unik në kuadër të mjedisit ku përdoret.

Sipas marrëveshjes, emrat e ndryshoreve shënohen me shkronja të vogla. Në qoftë se emri përbëhet nga më shumë fjalë, emrat ndahen me vizën e poshtme (_) ose shkronja fillestare e çdo fjale shkruhet me shkronjë të madhe. Për shembull, ndryshorja në të cilën jepet emri dhe mbiemri i nxenesit mund të emërtohet *emri_mbiemri_nxenesit* ose *emriMbiemriNxenesit*.

3.7.1. Deklarimi i ndryshoreve

Deklarimi i ndryshoreve

Deklarimi i ndryshoreve nënkupton përkufizimin e tipit të të dhënave që do të ruhen në ndryshore. Në një rresht të kodit mund të deklarohen më shumë ndryshore po të njëjtit tip, ashtu që emrat e ndryshoreve ndahen nëpërmjet presjeve. Ndryshorja mund të deklarohet kudo në program, mirëpo duhet të deklarohet para se të fillojë përdorimi i saj.

Duke marrë parasysh rregullat e emërtimit, janë paraqitur shembujt e deklarimit të rregullt të ndryshoreve.

```
int i;
int j, k, l;
byte numri_vogel;
short numri_shkurter;
long numri_madh;
float konstanta;
double numri_pi;
char c;
Boolean i_kycur;
```

3.7.2. Shoqërimi i vlerave ndryshoreve

Inicializimi

Mbas deklarimit të ndryshores, ndryshorja mund të përdoret për ruajtjen e të dhënave. ***Inicializimi i ndryshores*** nënkupton momentin kur ndryshores, herën e parë i shoqërohet një vlerë. Ndryshores mund t'i shoqërohet vetëm vlera që ka tipin e barabartë me tipin e deklaruar të ndryshores. Komandat për shoqërimin e vlerave ndryshoreve, mund të ndodhën kudo në program.

```
i = 1200;
j = -13256;
numri_vogel = 52;
numri_shkurter = 123456789;
```

```
numri_madh = 1234567890123456;
konstanta = 11.9;
numri_pi = 3.1415926;
c = 'S';
i_kycur = true;
```

3.8. Operatorët

Qëllimi themelor i operatorëve është ***ekzekutimi i funksioneve*** mbi një, dy ose tre operanda. Me disa lloje operatorësh u kemi njoftuar në pjesën e algoritmeve, kurse tani do t'u njoftojmë me llojet e operatorëve që mund të përdoren në gjuhën programuese Java.

Operatorët unarë

Operatori që kërkon vetëm një operand quhet ***unarni operator***. Operatorët e zmadhimit (e inkrementit) ++ dhe të zvogëlimit (të dekrementit) −− janë shembuj operatorësh unarë, që zmadhojnë, gjegjësisht, zvogëlojnë operandin e vet për 1. Operatorët unarë mund të jenë të tipit prefiks, kur operatori ceket para operandit (p.sh. ++*i* dhe të tipit postfix, kur operatori jepet mbas operandit të tij (p.sh. *i*++).

Në versionin prefiks shprehja ++op/-- op njehsohet në bazë të vlerës së operandit mbas zmadhimit/zvogëlimit. Në versionin postfiks, vlera e shprehjes njehsohet në bazë të vlerës së operandit para inkrementit/dekrementit./

Operatori që kërkon dy operandë quhet ***operator binar***. Për shembull, = është operator binar i cili i shoqëron vlerën e operandit të djathtë operandit të majtë. Operatorët binarë përdorin shënimin infiks, çfarë nënkupton që operatori ndodhet ndërmjet dy operandëve.

Operatorët binarë

Operatori ternar kërkon tre operanda. Gjuha programuese Java ka vetëm një operator ternar – ?: që është versioni i shkurtuar i shprehjes ***if - else*** izraza. Operatori ternar është infiks dhe çdo komponent i operatorit ndodhet ndërmjet operandëve.

Operatorët ternarë

Krahas funksionit të ekzekutimit të veprimeve, operatorët kanë aftësi që të ***kthejnë vlerat***. Për shembull, operatorët aritmetik, që kryejnë veprimet themelore aritmetike, siç është mbledhja dhe zbritja, kthejnë numrat si rezultat i veprimit aritmetik. Tipi i të dhënës që e kthen operatori aritmetik varet nga tipi i operandave të tij, p.sh. në qoftë se mblidhen dy numra të plotë, do të përftohet numër i plotë; kurse nëse mblidhet numri real me numrin e plotë, do të përftohet numri real.

Operatorët mund të ndahen në kategoritë e mëposhtme:

- operatorët aritmetikë,
- operatorët relacional dhe kondicionalë,
- operatorët mbi bite dhe operatorët logjikë,
- operatorët e shoqërimit,
- operatorët e tjerë.

3.8.1. Operatorët aritmetikë

Operatorët aritmetikë përdoren në shprehjet matematike për kryerjen e veprimeve themelore matematike, në të njëjtën mënyrë si në matematikë. Në gjuhën programuese Java përkrahen operatorët aritmetikë për numra të plotë dhe ata realë. Operatorët aritmetikë janë: + (mbledhja), − (zbritja), * (shumëzimi), / (pjesëtimi) dhe % (pjesëtimi sipas modulit).

Operatorët aritmetikë

```
int a,b;
int shuma, ndryshimi;
int prodhimi;
int heresi;
shuma = a + b;
ndryshimi = a + b;
prodhimi = a * b;
heresi = a / b;
```

Tabela 3.6. Operatorët aritmetikë

Operatori	Shembull	Përshkrimi
+	a + b	Mbledhja e **a** dhe **b**
–	a - b	Zbritja e **b** nga **a**
*	a * b	Shumëzimi i **a** me **b**
/	a / b	Pjesëtimi i **a** me **b**
%	a % b	Njehsimi i mbetjes gjatë pjesëtimit të **a** me **b** (e përkufizuar vetëm për numra të plotë)

Operatorët relacionalë

3.8.2. Operatorët relacionalë dhe kondicionalë

Operatorët relacionalë krahasojnë dy vlera dhe përcaktojnë raportin midis tyre. Operatorët kondicionalë krahasojnë vlerat logjike të operandëve.

```
int a,b;
boolean me_madhe, me_vogël, baraz;
boolean me_madhe_ose_baraz;
boolean me_vogël_ose_baraz;

a = 5;
b = 10;
me_vogel = a < b;
/* vlera e ndryshores më_vogël
është true */

me_madhe = a > b;
/* vlera e ndryshores më_madhe
është false */

baraz = a == b;
/* vlera e ndryshores baraz është
false */

me_madhe_ose_baraz = a >= b;
/* vlera e ndryshores
me_madhe_ose_baraz është false */

me_vogel_ose_baraz = a <= b;
/* vlera e ndryshores me_vogel_
ose_baraz është true */
```

Tabela 3.7. Operatorët relacionalë dhe kondicionalë

	Operatori	Shembull	Kthen "*true*" nëse
Operatorët relacionalë	<	**a** < **b**	**a** më i vogël se **b**
	<=	**a** <= **b**	**a** më i vogël ose i barabartë se **b**
	>	**a** > **b**	**a** më i madh se **b**
	>=	**a** >= **b**	**a** më i madh ose i barabartë se **b**
	==	**a** == **b**	**a** i barabartë me **b**
	!=	**a** != **b**	**a** i ndryshëm nga **b**
Operatorët kondicionalë	&&	**a** && **b**	**a** = ***true*** ose **b** = ***true*** **(AND logjike)**
	||	**a** || **b**	**a** = ***true*** ose **b** = ***true*** **(OR logjike)**
	!	! **a**	**a** = ***false*** **(NOT logjike)**

Nga matematika dihet se shprehjet e mëposhtme janë ekuivalente:

```
kushti = (a>b) && (c<d)
kushti = !((a<=b) || (c>=d))
kushti = !(a<=b) && !(c>=d)
```

Operatorët mbi bite

3.8.3. Operatorët mbi bite

Operatorët mbi bite kryejnë veprime të lëvizjes së biteve të operandëve majtas ose djathtas, si dhe veprime logjike në bite.

Lëvizja ndodh në drejtim (orientim) të shënuar nëpërmjet operatorit. P.sh. shprehja vijuese 13 >> 1 i lëviz bitet e numrit 13 djathtas për një pozicion. Prezantimi binar i numrit **13** është **1101**. Rezultati i lëvizjes për një vend djathtas mbi **1101** është **0110**, d.m.th. në formën decimale numri 6. Në anën e majtë i shtohet zeroja, kurse shifra e fundit e djathtë zhduket.

Tabela 3.8. Shift operatorët

Operatorët	Shembujt	Veprimet
>>	**a** >> **b**	Lëvizja e biteve të operandit **a** djathtas për **b** vende
<<	**a** << **b**	Lëvizja e biteve të operandit **a** majtas për **b** vende
>>>	**a** >>> **b**	Lëvizja e biteve të operandit **a** djathtas për **b** vende (pa parashenjë)

```
int a, b, c, d;
a = 10;
b = a >> 2;
c = a << 2;
d = a >>> 2;
```

Shembull shift operatori

a	0	1	1	0	0	0	0	48
a>>2	0	0	0	1	1	0	0	12

Tabela në vazhdim tregon katër operatorët në Javë, të cilët mundësojnë operacionet logjike mbi bajtet e operandëve të vet.

Tabela 3.9. Operatorët logjikë mbi bitet

Operatorët	Shembujt	Veprimet
&	**a** & **b**	AND logjike mbi bitet
|	**a** | **b**	OR logjike mbi bitet
^	**a** ^ **b**	OR ekskluzive mbi bitet
~	! **a**	NOT ekskluzive mbi bitet

Operatorët logjikë

Shembuj operatorësh logjikë mbi bite.

a	0	1	0	0	1	9
b	1	0	0	1	1	19
a & b	0	0	0	0	1	1

a	0	1	0	0	1	9
b	1	0	0	1	1	19
a ^ b	1	1	0	1	0	26

a	0	1	0	0	1	9
b	1	0	0	1	1	19
a | b	1	1	0	1	1	27

a	0	1	0	0	1	9
!a	1	0	1	1	0	22

3.8.4. Operatorët e shoqërimit

Operatorët e shoqërimit

Operatori themelor për shoqërimin e vlerës është = (p.sh. $c = a + b$). Ky operator vlerën e shprehjes në anën e djathtë i shoqëron ndryshores në anën e majtë. Kështu për shembull, shprehja $a + b = c$ nuk është korrekte.

Në Javë ekzistojnë edhe disa operatorë të tjerë të shoqërimit që mundësojnë shënim më të shkurtër.

Në qoftë se dëshironi të zmadhoni vlerën e një ndryshoreje, por që rezultati të ruhet në të njëjtën ndryshore, mund të veproni si në shembullin vijues: `a = a + 5;` ose shkurtimisht `a += 5;` Dy shprehjet paraprake janë ekuivalente.

```
int a, b, c, d, e;
a = 10;
b = 5;
d = 15;
e = 25;
c = a + b;
d += b;
e *= b;
```

Tabela 3.10. Operatorët e shoqërimit

Operatorët	Shembujt	Veprimet
=	a = b	a = b
+=	a += b	a = a + b
-=	a += b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b

3.8.5. Operatorët e tjerë

Operatorët e tjerë

Në tabelën 3.11. janë paraqitur edhe disa operatorë që përdoren për programim në gjuhën programuese Java. Domethënia e tyre do të sqarohet në kapitujt e mëposhtëm.

Tabela 3.11. Operatorët e tjerë në Java

Operatorët	Shembujt	Përshkrimi
`? :`	`op1 ? op2 : op3`	Versioni i shkurtuar i shprehjes *if-else*.
`[ ]`	`vargnumrash[6]`	Përdoret për deklarimin dhe krijimin e vargut dhe qasjes së elementeve të vargut.
`.`	`class.methode()`	Përdoret për qasjen e metodave të një klase të caktuar.
`new`	`new Integer(10);`	Krijon objektin apo vargun e ri.
`instanceof`	`op1 instanceof op2`	Përcakton a është operandi i parë instancë e operandit të dytë.

3.9. Konvertimi implicit dhe eksplicit i tipave

Shpeshherë nevojitet që vlera e një tipi t'i shoqërohet ndryshores së tipit të dytë. Kjo domethënë se duhet të kryhet ndryshimi i mënyrës së paraqitjes, i ashtuquajturi ***konvertimi i tipave***. Java mundëson dy mënyra të konvertimit të tipave: konvertimin implicit dhe eksplicit.

Konvertimi implicit i tipave

Konvertimi implicit kryhet në mënyrë automatike, në qoftë se janë të plotësuara kushtet e mëposhtme:

- tipi fillestar dhe tipi përfundimtar (destinacioni) janë kompatibil ndërmjet tyre (i takojnë po të njëjtit grup),
- tipi përfundimtar është më i madh sesa ai fillestar (përdoren më shumë bite).

Për shembull, gjithmonë mundet që vlera e tipit `int` t'i shoqërohet ndryshores së tipit `long`, ose vlera `float` t'i shoqërohet ndryshores së tipit `double`.

Shtrohet pyetja, si t'i shoqërojmë vlerën e tipit `int` ndryshores së tipit `byte`? Ky konvertim nuk do të bëhet në mënyrë implicite (automatike), sepse tipi `byte` është më i vogël se tipi `int`. Konvertimi implicit i tipave numerikë në `char` ose në `boolean` nuk është i mundshëm. Mirëpo, literali i plotë mund t'i shoqërohet tipit `char` (`char` ka numrin rendor në UNICODE tabelën, p.sh. numri rendor i A është 65).

```
int x = 2;
float y;
y = x;  /* konvertimi implicit i
           int në float */

short a = 120;
long b;
b = a;  /* konvertimi implicit i
           short në long */
```

Konvertimi eksplicit i tipave

Edhe pse konvertimi implicit i tipave të të dhënave është i dobishëm, ai nuk praktikohet e tipat që nuk janë kompatibile (të pajtueshme). Për këto raste zbatohet ***kastimi*** (anglisht cast). Kastimi është instruksion i kompajlerit që të konvertojë një tip në tipin tjetër. Ajo kërkon ***konvertimin eksplicit të tipave***. Forma e përgjithshme është:

(*tipi në destinacion*) *shprehja*

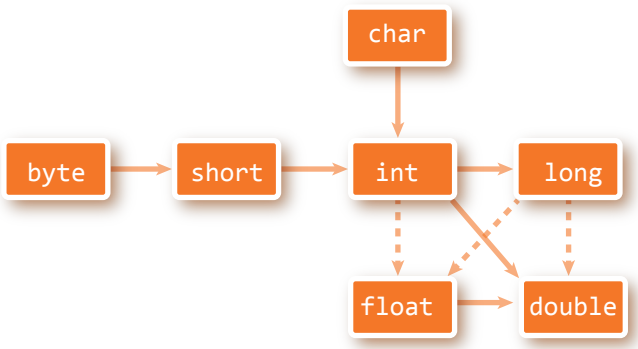


Figura 3.1. Konvertimet implicite dhe eksplicite të tipave primitivë

Tipi në destinacion paraqet tipin e dëshiruar në të cilin shprehja konvertohet.

Për shembull, në qoftë se dëshirojmë të konvertojnë vlerën e shprehjes *x*/*y* që i takon tipit `double` në tipin `int`, mund të shkruajmë:

```
double x, y;
int z;
z = (int) (x / y);
```

Kastimi është i domosdoshëm në këtë rast, sepse nuk ekziston konvertimi automatik i tipit `double` në tipin `int`. Kur vlera `double` kastohet në tipin e numrave të plotë, pjesa decimale zhduket dhe do të këputet.

Për shembull, në qoftë se vlerës 1,23 i shoqërohet ndryshorja nga numrat e plotë, rezultati do të jetë 1. Pjesa decimale 0,23 zhduket.

Gjithashtu, kur kastohet `long` në `short`, informata humbet, në qoftë se vlera për tipin `long` e tejkalon vëllimin e tipit `short`.

Duhet të jemi shumë të kujdesshëm gjatë konvertimit!

3.10. Programi i dytë

Më në fund, erdhi koha të shkruani programin e dytë në Javë.

Programi i dytë

Detyra: Të shkruhet programi që për numrat e dhënë: 3, 33 dhe 333, të njehsojë shumën, prodhimin dhe vlerën mesatare.

Që rezultati i programit të mund të paraqitet në ekran, do të përdorim mënyrën standarde për paraqitjen e vlerës së ndryshores në ekran me ndihmën e komandës (thirrjes) **System.out.println(*emri_ndryshores*)**. Mbi rrjedhat standarde hyrëse dhe dalëse të të dhënave do të takoheni në kapitujt e mëposhtëm të librit. Për zgjidhje të suksesshme të detyrës së dhënë, mjafton që të njihni sintaksën e mëposhtme për paraqitjen e përmbajtjes së dëshiruar në ekran:

System.out.println("Vlera mesatare është:" + *emri_ndryshores*)

Për paraqitjen e rezultatit në ekran në dy rreshta mund të përdoret simboli special *\n*, si në shembullin.

System.out.println("Shuma e numrave është:" + *emri_ndryshores* +
"\n kurse prodhimi i numrave është: " + *emri_ndryshores*)

Projekti i tërë ndodhet në dosjen *Teksti/src/ProgramiDytë*

Zgjidhje:

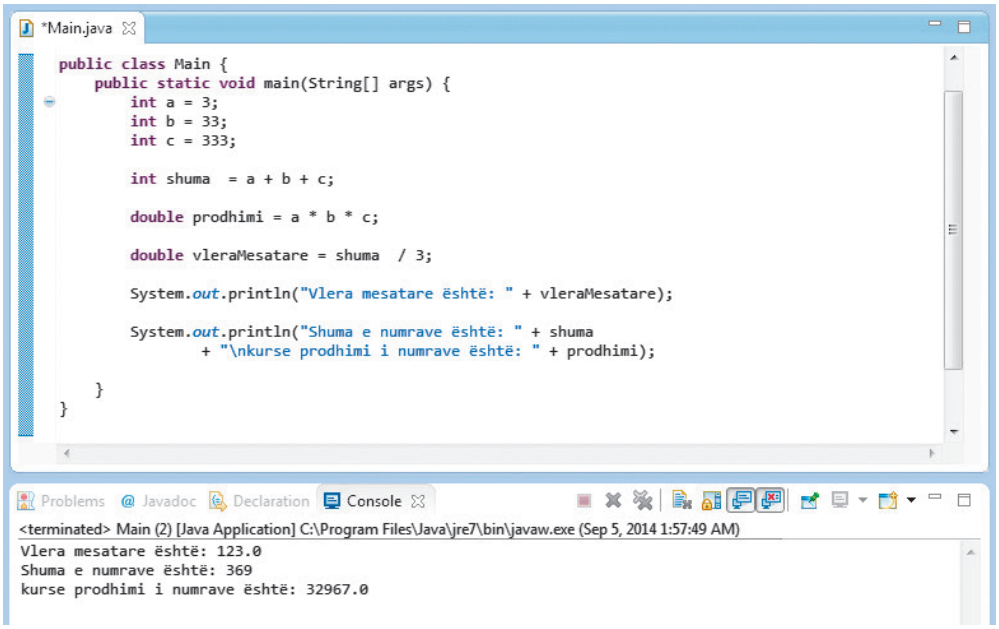


Figura 3.2. Kodi i programit të dytë dhe paraqitja e rezultatit

Pyetje dhe detyra kontrolli

1. Thuaji elementet themelore të gjuhës programuese Java.
2. Sqaro qëllimin (rolin) e komenteve në shënimin e programit.
3. A mund të shënohen komentet brenda komenteve?
4. Çfarë janë literalët?
5. Për çfarë shërbejnë separatorët?
6. Sqaro rolin e fjalëve të rezervuara.
7. A e dallon Java përdorimin e shkronjave të vogla dhe të mëdha gjatë shënimit të objekteve.
8. Thuaji tipat e thjeshtë të të dhënave në Java.
9. Cilit tip të të dhënave duhet t'i takojë ndryshorja që të mund të ruajë vlerën 12345?
10. Cilat vlera mund të ruhen në ndryshoret e tipit `char`?
11. Cili është dallimi midis tipave të dhënë të thjeshtë dhe kompleksë?
12. Sqaro qëllimin (rolin) e ndryshoreve.
13. Si deklarohen ndryshoret?
14. Sqaro qëllimin (rolin) kryesor të operatorëve dhe cilat kategori operatorësh ekzistojnë.
15. Cili është konvertimi implicit dhe cili është konvertimi eksplicit i të dhënave?
16. Jepni shembuj tipash të të dhënave që nuk janë kompatibile.

Puno vetë

1. Plotëso vlerat që mungojnë në tabelë.

Veprimi	Rezultati
`a = 10` `a += 7`	`a=?`

`a = 124`
`b = 20`
`c = a%b`
	`c=?`

`x = 35` `x = x>>2`	`x=?`

2. Në tabelën e mëposhtme janë cekur deklarimet e ndryshoreve dhe komandat përkatëse mbi to. Cilat nga komandat janë të palejueshme dhe pse?

Deklarimi i ndryshoreve	Komanda	
`int a, b, c;` `short x, y, z;`	`x=1;`	`a=x;`
	`x=a=b);`	`x=c=y;`
	`x=y-a;`	`c=y-5;`
	`a=(b=c);`	`c=a-b;`
`int a, b, c;` `boolean x, y, z;`	`x=true;`	`x=y+a;`
	`a=x;`	`c=y/2;`
	`x=a*b;`	`a=b==c;`
	`x=c==y;`	`c=a-b;`

IV.

BAZAT E PROGRAMIMIT TË ORIENTUAR NË OBJEKTE: KLASAT DHE OBJEKTET

Zhvillimi i softuerit të orientuar në objekte (OO) është aktual që nga viti 1960. Sot gati çdo program i rëndësishëm përdorë objektet. Që të bëheni programues, të cilët dëshirojnë të përpilojnë softuerë të mirë, dhe ky duhet të jetë qëllimi kryesor i të gjithë neve, duhet të përvetësoni konceptet e rafinuara që quhen objekte dhe klasa.

Prandaj në këtë kapitull do të mësoni:

- çfarë është klasa dhe çfarë është objekti,
- çfarë janë mesazhet ndërmjet objekteve,
- si modelohen problemet nga jeta e përditshme nëpërmjet klasave dhe objekteve,
- çfarë paraqet koncepti i trashëgimit,
- cilët janë principet themelore të Javës si gjuhë OO që mundëson programim me klasa dhe objekte.

Përvetësimi i suksesshëm i këtyre koncepteve në këtë kapitull është hapi i parë dhe më i rëndësishëm në botën e programimit OO!

Kërkesat e tregut të sotëm për softuerë imponojnë nevojën për zhvillimin e shpejtë të softuerit të dedikuar një numri të madh përdoruesish në platforma të ndryshme (i ashtuquajturi interoperabilitet ose ndërveprim). Me qëllim plotësimi të këtyre kërkesave, industria e sotme kompjuterike, ajo për softuerë dhe ajo për harduerë, zhvillohet shpejt duke mundësuar përkrahjen programore me theks të veçantë në mundësinë e përdorimit të pjesëve të veçanta të kodit, ndërveprimin dhe nivelin e lartë të sigurisë. Kështu ka lindur ***programimi i orientuar në objekte***.

Shpeshherë thuhet se ***programimi procedural*** është gur themeli që nga "epoka e bronzit" e kompjuterëve.

Emërtimi i tij mund t'u sugjerojë se bëhet fjalë mbi një mënyrë të ndërlikuar të programimit që nuk është lehtë të përvetësohet. Megjithatë, gjuha programuese Java është konceptuar ashtu që nga fillimet e saj të jetë gjuhë e orientuar në objekte, që lehtëson dhe thjeshtëson dukshëm shënimin e kodit të OO.

4.1. Idetë themelore të orientimit në objekte

Para se të thellohemi në terminologjinë OO, të shqyrtojnë pyetjen: **Çfarë është, në të vërtetë, objekti?**

Shumica e njerëzve në jetën e përditshme mendojnë nëpërmjet objekteve, botën e përjetojnë si një varg objektesh, të cilat gjithashtu mund të përbëhen nga objekte ose mund të komunikojnë me të tjerët. Një shembull shumë i thjeshtë: kur e shikoni një person e shqyrtoni si një objekt. Personi ka vetitë, siç janë ngjyra e syve, sjellja dhe mënyra e ecjes.

Aftësia e njohjes së objekteve fizike është veti që njerëzit e zhvillojnë në pjesën e hershme të jetës. Dekarti ka konstatuar në kohën e tij se orientimi në objekte pasqyron më së miri parafytyrimin e njerëzve të botës. Mendimet tona dhe memoria janë të organizuara në formën e objekteve dhe lidhjeve midis tyre.

OO, siç tregon edhe vetë emri, është i orientuar në drejtim të objekteve, prandaj për fillim është më së miri të ceket se objekti është njëri prej njësive mësimore themelore të programimit OO. Në qoftë se programin e konsideroni si organizëm, do të ishte i përbërë nga modulet me funksione të ndryshme, siç janë organet. Këto module funksionale përbëhen nga pjesët e veta, që janë në të vërtetë, objektet që komunikojnë ndërmjet vete.

Objekti përkufizohet si njësi e posaçme që përmban të dhënat dhe sjelljen. Të dhënat tregojnë ***gjendjen*** e objektit. Pra, objekti ka ***gjendjen*** dhe ***sjelljen***.

Objekti

Marrim si shembull objektin e një shkolle. Që të jetë objekt, duhet ta paraqesim me një gjendje dhe sjellje të caktuar. Fillimisht të përgjigjemi në pyetje se çfarë është gjendja e një shkolle? Në pyetje janë karakteristikat e saj siç janë emri, viti i themelimit, adresa, numri i nxënësve etj. Në figurën 4.1. mund të vini re se janë paraqitur nxënësit që hyjnë në shkollë. Edhe ata janë objekte, por të tipit të ndryshëm në krahasim me shkollën dhe kanë gjendjen dhe sjelljen.

SHKOLLA

emërtimi: "Gjimnazi i parë malazez"
viti i themelimit: 2000
adresa: Podgorica p.n.
numri i nxënësve: 1000
sipërfaqja e godinës: 2.000,00 (m^2)
numri i kateve: 2

NXENESI

emri: Mark
mbiemri: Markaj
paralelja: VII 2
nota nga matematika: 5

Figura 4.1. Objektet "shkolla" dhe "nxënësi"

Disa shembuj objektesh: *lapsi*, *tastiera*, *tavolina*, *libri*.

Karakteristika e objekteve:

- Objekti sjellët si tërësi, e përbërë nga pjesët,
- Objekti ka gjendjet (ka veti që mund të ndryshojnë)
- Objekti mund të kryej disa punë (p.sh. lapsi shkruan) dhe në të mund të kryhen disa punë (p.sh. librit t'i ndryshohet kopertina).

Në figurë janë paraqitur karakteristikat e këtyre dy objekteve. Vëmë re se çdo karakteristikë ka një vlerë të veten. Këto janë vlerat që një objekt e dallojnë nga objekti tjetër (p.sh. nuk do të ketë çdo nxënës emrin e njëjtë, mbiemrin apo notën në matematikë). Të gjitha këto vlera kanë tip të caktuar. Mund të vini re se sipërfaqja e godinës është paraqitur në metra katrorë, kurs numri i kateve është numër i plotë. Në program, këto të dhëna do të jenë të shënuara nëpërmjet tipave përkatës të Javës, mbi çka do të flasim në hollësi në kapitullin pasues.

Tani duhet të përgjigjemi në pyetjet, çfarë paraqet sjellja e një shkolle? Edhe pse mendja e shëndoshë na tregon se një shkollë nuk ka "sjellje", përveç nëse ka ndihmesën e një teknologjie moderne (p.sh. godinat inteligjente), ne si programues mund të shqyrtojmë këtë pyetje nga një kënd tjetër. Sjellja nuk nënkupton doemos një aktivitet që shkolla e ndërmerr, por edhe aktivitetet që bëhen në shkollë i përkasin kategorisë së sjelljes. Kështu, për shembull sjellje shkolle mund të jenë: ndryshimi i emrit të shkollës, ndërtimi i pjesës së re të ndërtesës me çfarë zmadhohet sipërfaqja e përgjithshme e shkollës ose ndërtimi i paraleles së re për një numër më të madh të nxënësve të klasës së tetë, etj.

Ne vetëm përmendëm një shembull objekti. Tani duhet të sqarohen edhe disa koncepte të OO, prej të cilave më i rëndësishmi është koncepti i klasës.

4.2. Klasat dhe objektet

Në natyrë mund të takohet një numër i madh i entiteteve të ngjashme. Disa janë identikë disa kanë shumë pak dallime, kurse disa të tjera janë plotësisht të ndryshëm. Mirëpo, njeriu, falë inteligjencës së vet, ka kryer disa klasifikime themelore të sendeve, qenieve dhe dukurive që i ka takuar në jetë. Kështu, për shembull, është themeluar klasifikimi i qenieve të gjalla në njerëz, kafshë dhe bimë.

Pra, klasa i përkufizon disa veti të përbashkëta dhe sjellje të entiteteve që numërohen nën klasifikimin e saj, por që dallohen në detajet e lidhura me vetitë e caktuara.

Klasa

Në programim, ***klasa*** mund të imagjinohet si një shabllon nga i cili formohen objektet. Klasa përkufizon se cilat ***karakteristika*** do t'i ketë secili objekt i asaj klase, si dhe ***sjelljet*** e saj. Pra çdo objekt është një instancë e klasës së vet. Të fillojmë me shembullin që është paraqitur në figurën 4.2. dhe që mund të përdoret gjatë tërë Tekstit. Në figurë, në anën e majtë ndodhet klasa, që përgjithëson nxënësin. Ajo përshkruan karakteristikat e nxënësit: nxënësi ka emrin, mbiemrin, ka numrin e evidencës në ditar, ka emrin e shkollës në të cilën mëson, paralelen, notën nga matematika. Ajo klasë në mënyrë abstrakte paraqet nxënësin dhe që të paraqitni të dhënat mbi nxënësit konkretë (p.sh. Marku, Ana, Liria) nevojitet që nga klasa ekzistuese të formoni mostra (d.m.th. instanca) që i përgjigjen secilit nga ata nxënës.

Instanca (objekti)

Në fjalorin e programimit të orientuar në objekte, ***instanca*** është thjesht shprehja për objektin, në dallim nga ***klasa***, që paraqet objektin në mënyrë abstrakte. Prandaj termat ***"objekti nxënës"***, ***"instanca nxënës"*** i referohet të njëjtës gjë: objektit që është formuar nga klasa ***Nxenesi***.

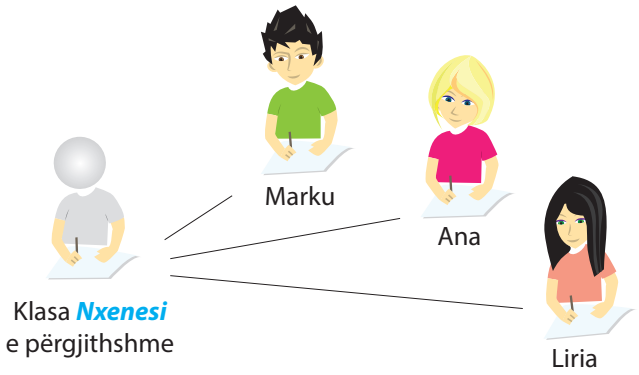


Figura 4.2. Shembuj objektesh të klasës *Nxenesi*

Dhe si duket kjo në Javë?

Shumë thjesht. Mund të formoni klasën tuaj me emrin *Nxenesi*, që paraqet nxënësit. Klasa do të përkufizojë karakteristikat e tij të përgjithshme që përfshijnë emrin, mbiemrin, numrin evidentues të nxënësit në ditar, shkollën në të cilën shkon, paralelen, notën nga programimi... Mbas përkufizimit të klasës, ajo mund të përdoret më tutje gjatë programimit. Nevojitet të formohen objektet përkatëse të asaj klase, d.m.th. të gjitha karakteristikave të klasës *Nxenesi* t'i shoqërohen vlera konkrete.

Kur krijohet objekti, themi se është *krijuar një mostër e klasës,* gjegjësisht është *formuar objekti*. Prandaj, nëse krijojmë tre nxënës, ne në të vërtetë krijojmë tri mostra (anglisht *instance*) të klasës *Nxenesi*. Duke pasur parasysh emërtimin e cekur në anglisht, në gjuhën shqipe themi se ***kemi tri instanca të klasës Nxenesi***.

Kur shkruani programin në Javë, krijoni në të vërtetë një bashkësi klasash. Mbasi që programi juaj të niset, kompjuteri formon instanca të klasave. Detyra juaj është të përkufizoni të gjitha ato klasa që u nevojiten për kryerjen e detyrës konkrete të programit tuaj. Krahas klasave që i krijoni ju, Java ka një varg klasash që mund t'i përdorni në programet tuaja dhe të cilat do të përkujdesen për të gjitha funksionet e programit: hyrja dhe dalja e të dhënave, vizatimi i grafikës dhe lëshimit të zërit, komunikimi nëpërmjet internetit dhe shumë të tjera.

4.3. Karakteristikat dhe sjellja e objekteve

Në mënyrë të njëjtë si në botën reale, në të cilën çdo objekt ka karakteristikat dhe sjelljen që e dallojnë nga objektet e tjera, ashtu edhe në gjuhën programuese Java, çdo klasë përmban karakteristikat (atributet) dhe posedon sjelljen e përkufizuar, me të cilën dallohet nga klasat e tjera.

Karakteristika (atribute) konsiderohen të gjitha vetitë që klasën dhe objektin e dallojnë nga të tjerët dhe e përkufizojnë identitetin e vet, p.sh. forma, ngjyra, madhësia etj.

Karakteristikat e objekteve

Në shembullin e paraqitur në figurën 4.3., atributet me të cilat janë përshkruar të gjithë nxënësit janë: emri, mbiemri, numri i evidencës, emri i shkollës, paralelja, nota nga matematika. Atributet në kapitujt e mëposhtëm do të sqarohen në detaje, kur do të sqarojmë krijimin e klasave në Javë.

Nxenesi
Numri i evidencës
Emri
Mbiemri
Emri i shkollës
Paralelja
Nota nga matematika

Figura 4.3. Klasa *Nxenesi*

Disa mostra (d.m.th. instanca) klase që paraqesin objektet e klasës *Nxenesi* janë paraqitur në figurën 4.4.

Figura 4.4. Shembuj dy objektesh të klasës *Nxenesi*

Krahas karakteristikave, objektet i karakterizon edhe sjellja:

Sjellja e objektit

Të gjitha ***funksionet*** që ndikojnë në karakteristikat e objektit konsiderohen si sjellje të tij.

P.sh. funksioni i përmirësimit të notës nga matematika, ka një ndikim konkret të përkufizuar në notën e nxënësit nga matematika. Të gjitha funksionet me të cilat ndikohet në gjendjen e një objekti (ose nëpërmjet të cilave objektet ndikojnë në mënyrë individuale në gjendjen e vet), në Javë quhen ***metoda***. Në lëmin e programimit, që të përcaktohet se si sillet një objekt, përdoren metodat që kryejnë detyra të caktuara. Ato, punën e vetë e kryejnë mbi instancat e asaj klase, mirëpo ndikimi i tyre nuk është i kufizuar ekskluzivisht në gjendjen e një objekti të vetëm.

Metodat, të cilave nuk mund t'u qasen objektet e tjera nga jashtë quhen ***sjellje e brendshme***. Në anën tjetër, metodat mund të "ftojnë" edhe metodat në objektet e tjera dhe në atë mënyrë të "lutet" objekti tjetër që të ndryshojë gjendjen e tij. Një mënyra e tillë e komunikimit shpeshherë quhet ***dërgimi i mesazheve***.

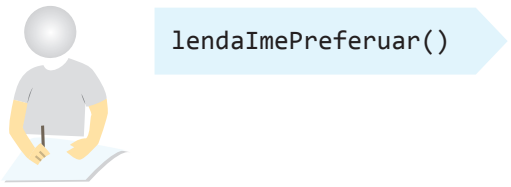


Figura 4.5. Shembull i sjelljes së brendshme të klasës *Nxenesi*

Në figurën 4.5. është paraqitur shembulli i sjelljes së brendshme me të cilën nxënësi i përgjigjet pyetjes se cila është lënda e tij e preferuar. Në shembullin në figurën 4.6. supozojmë se paraprakisht është krijuar objekti i klasës *Profesori p* dhe ai dërgon mesazhin objektit në klasën klase *Nxenesi*, me të cilën ia rrit notën nga matematika mbas provimit të shkëlqyeshëm me shkrim.

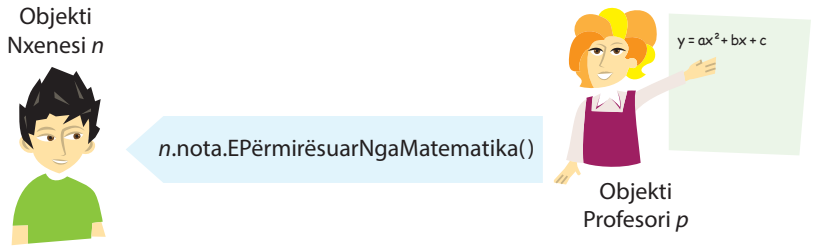


Figura 4.6. Shembull i dërgimit të mesazhit të objektit të klasës *Profesori* objektit të klasës *Nxenesi*

Në fund, klasa *Nxenesi* që e kemi përkufizuar nëpërmjet atributeve dhe metodave të dhëna është paraqitur në figurën 4.7.

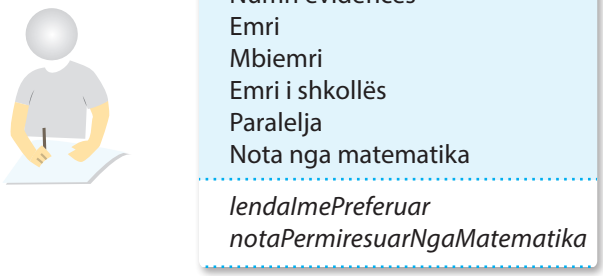


Figura 4.7. Klasa *Nxenesi*

Shembull.

Të përkufizojmë klasën *Shkolla*. Në fillim kemi cekur karakteristikat e saj (emri, adresa, viti i themelimit, numri i nxënësve, sipërfaqja e godinës dhe numri i kateve), si dhe sjelljet (ndryshimi i numrit të nxënësve dhe krijimi i paraleleve të reja), ashtu që përftojmë klasën e paraqitur në figurën 4.8.

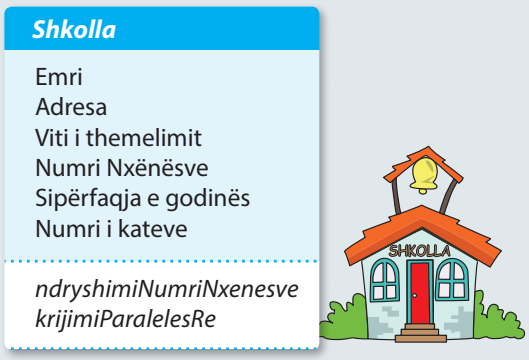


Figura 4.8. Klasa *Shkolla*

4.4. Trashëgimi

Siç kemi cekur më herët, një ndër vetitë më të fuqishme të programit OO është përdorimi i shumëfishtë i kodit programues. Edhe në gjuhët procedurale, në një masë të caktuar është e mundshme të bëhet përdorimi i shumëfishtë i pjesëve të kodit – duke përkufizuar funksione/ procedura që më vonë mund të thirren (aktivohen) një numër të çfarëdoshëm herësh. Mirëpo programimi OO shkon një hap më tutje dhe mundëson përkufizimin e konceptit të trashëgimit:

Trashëgimi mundëson krijimin e klasës që është e ngjashme me atë paraprake, por që megjithatë ka edhe disa veti të veta të posaçme.

Trashëgimi mundëson krijimin e hierarkisë në atë mënyrë që klasën e përgjithshme mund ta trashëgojnë klasat e tjera. Klasat specifike trashëgojnë atribute dhe sjellje të klasës së përgjithshme dhe/ose shtojnë atë që për të është e vetme. Në atë rast, ekziston raporti prindër – pasardhës. Në Javë klasa e përgjithshme (d.m.th. klasa që trashëgohet) quhet **mbiklasa** (anglisht ***superclass***), kurse klasa që trashëgon quhet **nënklasa** (anglisht *subclass*). Çdo klasë ka mbiklasën e vet, dhe mund të ketë më shumë nënklasa. Nënklasa trashëgon të gjitha ndryshoret dhe metodat që janë përkufizuar në mbiklasën dhe i shton ndryshore ose/dhe metodat e veta të posaçme. Një shembull është paraqitur në figurën 4.9., kurse mbi raportin e klasave A, B, C dhe D në hierarki mund të thuhet:

- Klasa A është *mbiklasë* e klasës B, kurse klasa B është *nënklasë* e klasës A (shpeshherë thuhet se klasa B e *trashëgon* klasën A).
- Klasa B është *mbiklasë* e klasës C dhe D, kurse klasat C dhe D janë *nënklasa* (*e trashëgojnë*) klasën B.

Një nga problemet kryesore që paraqitet në një organizatë hierarkike të tillë është klasifikimi i karakteristikave dhe sjelljeve të përbashkëta të klasave të ndryshme. P.sh. shqyrtojmë klasat ***NxenesiShkollaProfesionale*** dhe ***NxenesiIT*** që paraqesin nxënësit e

Trashëgimi

Në Javë në maje të hierarkisë ndodhet klasa e quajtur *Object*. Ajo është klasa më e përgjithshme që përcakton sjelljet e përgjithshme të objekteve në Javë. Të gjitha klasat që krijohen në Javë doemos trashëgojnë karakteristikat e kësaj klase.

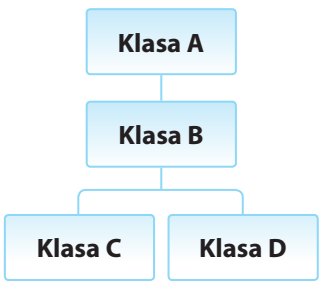


Figura 4.9.
Shembull hierarkie klasash

shkollave të mesme: klasa e parë përfshin vetëm nxënësit e një shkolle profesionale, kurse klasa e dytë gjimnazistët që për lëndë zgjedhëse kanë marrë grupin e lëndëve nga informatika. Të dy klasat i karakterizojnë: emri, mbiemri, numri evidentues, emri i shkollës, paralelja, nota nga matematika (si edhe për objektet e klasës *Nxenesi*). Mirëpo, për klasën *NxenesiShkollaProfesionale* është e njohur edhe nota nga praktika profesionale të cilën, nxënësit, janë të detyruar ta bëjnë, kurse për *NxenesiIT* nota nga programimi. Domethënë, të dy klasat e trashëgojnë klasën *Nxenesi*, siç është paraqitur në figurën 4.10.

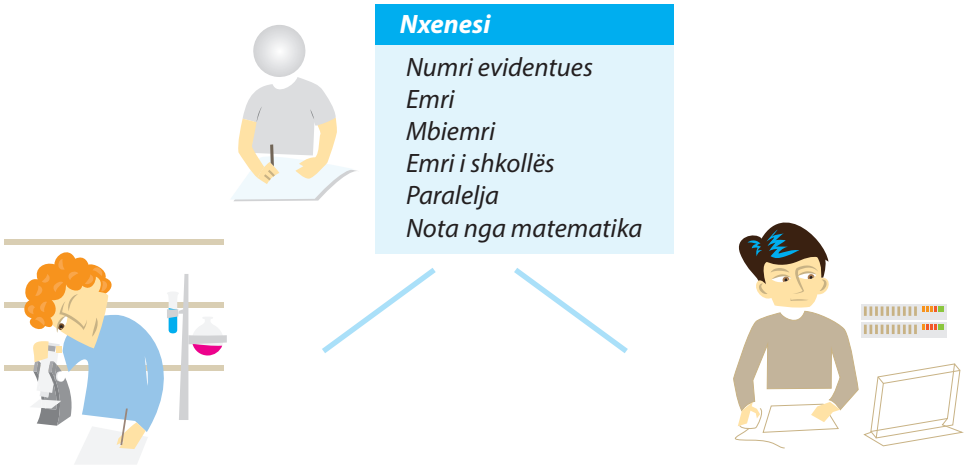


Figura 4.10. Shembull klasash *NxenesiShkollaProfesionale* dhe *NxenesiIT* që trashëgojnë klasën *Nxenesi*

Siç kemi cekur më herët, nënklasat trashëgojnë të gjitha atributet dhe metodat nga mbiklasa. Shikoni shembullin 4.11. në të cilin janë përmendur dy nivele të hierarkive të klasave, kështu që *KlasaC* i trashëgon të gjitha atributet e *KlasaB*, kurse *KlasaB* i trashëgon të gjitha atributet e *KlasaA*.

Në shembullin tonë, klasa *NxenesiIT* që e trashëgon klasën *Nxenesi* ka atributet e mëposhtme.

```
numriEvidences
emri
mbiemri
emriShkolles          // trashëguar nga klasa Nxenesi
paralelja
notaNgaMatematika
notaNgaProgramimi     // përkufizuat vetëm për gjimnazistët me informatikë
```

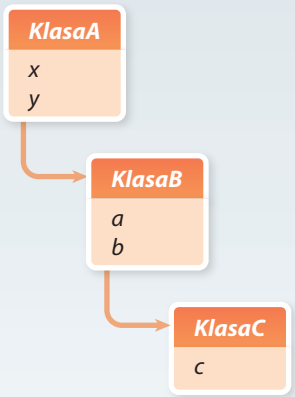
Kjo mënyrë e trashëgimit quhet ***trashëgim i njëfishtë***, që domethënë se çdo klasë mund të ketë vetëm një mbiklasë.

Si funksionon trashëgimi dhe çfarë në të vërtetë ndodh në momentin e trashëgimit me ndryshore dhe metoda që janë përkufizuar në klasa të trashëguara?

Për çdo instancë, hapësira në memorie krijohet për secilin nga atributet që klasa e re e përkufizon, duke i shtuar të gjithë atributet që klasa e re i trashëgon nga mbiklasa dhe mbiklasat e saj. Në mënyrë të ngjashme mund të shqyrtojmë edhe metodat. Objektet e reja kanë qasje të të gjitha metodave që ndodhen në to dhe në mbiklasat e tyre, mirëpo Java gjithmonë së pari verifikon nëse metoda e thirrur ndodhet në objektin momental, dhe fill mbas, nëse nuk ndodhet, i fton metodat përkatëse nga mbiklasa.

Prandaj mund të ndodhë që klasa e re të përmbajë metodën me emër të njëjtë dhe me parametra të njëjtë si edhe njëra nga mbiklasat. Në atë rast do të ekzekutohet metoda që ndodhet më poshtë në hierarki. Edhe pse kjo duket si problematike, në të vërtetë

Në disa gjuhë të tjera të OO, siç është C++, klasat mund të kenë më shumë se një mbiklasë, si dhe mund të trashëgojnë njëkohësisht karakteristikat dhe sjelljet e më shumë klasave (çfarë quhet ***trashëgimi e shumëfishtë***). Mundësia e trashëgimit të shumëfishtë mund të krijojë, krahas dobive të ndryshme, edhe probleme të padëshiruara. Prandaj autorët e Javës kanë vendosur që të disfavorizojnë trashëgiminë e shumëfishtë, dhe kështu programet në Javë bëhen të qarta, kurse probabiliteti i gabimit është i vogël.



KlasaB ka atributet e mëposhtme:
x
y
a
b

KlasaC ka atributet e mëposhtme:
x
y
a
b
c

Figura 4.11.
Shembull trashëgimi

bëhet fjalë për një veti shumë to dobishme të Javës. Mund të trashëgohet klasa e tërë dhe pastaj të "mposhten" disa metoda dhe atribute të saj, ashtu që të kryejnë një detyrë të re, ose të adaptohen me probleme konkrete të klasës. Kjo është një prej përparësive më të mëdha të teknologjisë OO dhe shpeshherë kjo theksohet veçanërisht me emrin ***polimorfizëm*** (grumbull formash).

Polimorfizëm

Tani zgjerojmë shembullin tonë të klasës *Nxenesi* me metodën *perfaqesimiProfesionit* që do të ilustrojë termin polimorfizëm. Duke thirrur këtë metodë, nxënësi duhet të specifikojë që me përfundimin e shkollës së mesme ka nivelin e përgatitjes së mesme profesionale. Mirëpo, *NxenesiIT* do të specifikojë përshkrimin shtesë që njeh llojet e ndryshme dhe fushat e aplikimit të teknologjive të informacionit dhe komunikimit duke theksuar notën e vet nga programimi, si një ndër lëndët më të rëndësishme nga ajo fushë. Në anën tjetër, *NxenesiShkollaProfesionale* dhe *NxenesiIT* janë klasa që do të trashëgojnë metodën *perfaqesimiProfesionit* nga mbiklasa *Nxenesi*, mirëpo të dy klasat sigurojnë realizimin e ndryshëm, d.m.th. ato nuk do të marrin realizimin që është përkufizuar në klasën *Nxenesi*.

Paraqitja grafike e mposhtjes së metodave është dhënë në figurën 4.12. (ku klasa B merr përkufizimin e metodës nga klasa A, kurse klasa C i mposht sipas përkufizimit këto metoda). Do të flitet më shumë mbi trashëgimin në kapitullin 5.5.

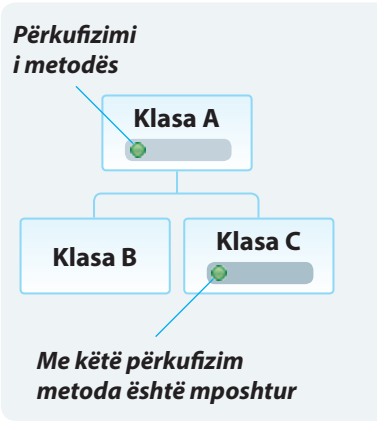


Figura 4.12. Shembull mposhtjes së metodës

4.5. Shembull

Idetë dhe parimet kryesore të orientimit në objekte kuptohen shumë më lehtë nëse ndërlidhen me shembuj nga praktika. Prandaj duhet të njoftohemi në detaje me një shembull të thjeshtë të paraleles së nxënësve të shkollës së mesme, në të cilin do të sqarojmë në hollësi se çfarë janë në të vërtetë objektet dhe çfarë konsiderohen si karakteristika objektesh.

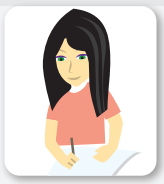
Përbërja e objekteve të paraleles IV 3 është paraqitur në figurën 4.13. Figura tregon të dhënat mbi sukseset nga matematika të tre nxënësve të kësaj paralele. Çdo nxënës është shembull objekti të klasës *Nxenesi* që e kemi përkufizuar më herët.

BOTA E JASHTME

Paralelja IV-3

NumriEvidence: 05
emri: Mark
mbiemri: Markaj
paralelja: IV 3
emriShkolles: Gjimnazi, Podgorica
notaNgaMatematika: 3

NumriEvidence: 10
emri: Ana
mbiemri: Malaj
paralelja: IV 3
emriShkolles: Gjimnazi, Podgorica
notaNgaMatematika: 4



NumriEvidence: 18
emri: Jehona
mbiemri: Gjokaj
paralelja: IV 3
emriShkolles: Gjimnazi, Podgorica
notaNgaMatematika: 5

Figura 4.13. Shembuj objektesh të klasës *Nxenesi*

Çdo objekt karakterizohet me numër unit të evidencës (në ditar të paraleles), emër, mbiemër, paralele dhe emër të shkollës. Si shqyrtohen notat nga matematika? Çdo objekt ka edhe atributin *notaNgaMatematika*. ***Bashkësia e të gjitha të dhënave që përfaqësojnë objektin e dhënë quhet gjendja e tij.***

Gjendja e objektit

Një gjendje e plotë e një objekti mund të duket kështu:

numriEvidences: 18
emri: Lirie
mbiemri: Gjokaj
paralelja: IV 3
emriShkolles: Gjimnazi, Podgorica
notaNgaMatematika: 5

Figura 4.13. është ndarë në dy pjesë. Pjesa e brendshme paraqet zonën e modelit tonë për paraqitjen e të dhënave mbi nxënësit, kurse pjesa e jashtme paraqet botën e jashtme: botën reale të nxënësve dhe paraleleve. Në botën e jashtme ndodhin ngjarje që ndikojnë drejtpërdrejt në botën e objekteve. Me fjalë të tjera, bota e objekteve paraqet një pamje e përshtatshme të ngjarjeve të vërteta (reale).

Ngjarjet e mundshme që ndikojnë në ndryshimin e botës së objekteve janë shënimet e përmendura të notave të reja nga matematika nga ana e arsimtarit mbas kontrollit të provimeve me shkrim ose regjistrimi i nxënësit të ri. Rezultatet e ngjarjeve janë më shpesh ndryshimi i gjendjes së objektit (nota e re nga matematika), krijimi i objektit të ri (nxënësi i ri në paralele), zhdukja e objektit ekzistues (p.sh. nxënësi është shpërngulur me prindër në Kanada, prandaj është çregjistruar nga shkolla) ose dërgimi i informacioneve mbi objektin në botën e jashtme (informacioni mbi notën momentale nga matematika që kumtohet në mbledhjen e prindërve).

Siç është thënë më herët, mesazhet paraqesin mënyrën me të cilën nga bota e jashtme iu qaset objekteve të caktuara ose ndryshohet gjendja e tyre. Kur duhet t'i qaset një objekti, i çohet mesazhi përkatës, kurse objekti duhet të ketë metodën e përkufizuar për ofrimin e përgjigjes në mesazh.

P.sh., në qoftë se duhet t'i shkruhet Markut nota 4 nga Matematika, do të dërgohet mesazhi:

nxenesi05.shenoNotenMatematika(4);

Ky kod programues, i shënuar në Javë, përbëhet nga dy pjesë:

- pjesa e parë është emri i objektit i cili duhet të fitojë mesazhin. Në rastin tonë, bëhet fjalë për Markun, nxënësin me numrin e evidencës 05, prandaj objekti përkatës është quajtur *Nxenesi05*;
- pjesa e dytë e asaj komande është vetë mesazhi i objektit. Në këtë rast thirret metoda *shenoNotenMatematika*, duke specifikuar argumentet e saj, d.m.th. nota nga matematika.

Detajet rreth shënimit të kodit programues për krijimin e klasave (përkufizimin e atributeve dhe metodave), si dhe punën me objekte në Javë do ta mësoni në kapitullin e ardhshëm.

Mesazhet e ndryshojnë gjendjen e objektit dhe mbas kësaj e kthejnë vlerën e përftuar ose thjesht e kthejnë vlerën pa ndryshimin e gjendjes së objektit.

Disa nga mesazhet e tjera nuk duhet të kenë argumente:

nxenesi10.tregoNotenMatematika();

Ky mesazh i dërgohet nxënësit me numrin e evidencës 10 (d.m.th. Anës) dhe rezultati është nota e saj nga matematika.

nxenesi05.zvogeloNotenMatematika();

Me këtë mesazh nxënësit me numër të evidencës 05 (d.m.th. Markut) i shënohet nota e re nga matematika, e zvogëluar për 1 në raport me notën ekzistuese. Fjala është mbi një shembull mesazhi pa argument, që kryen ndryshimin e gjendjes së objektit, për dallim nga mesazhi paraprak, i cili ka treguar vlerën e gjendjes pa pësuar ndryshim.

Qëllimi i shembullit paraprak është të tregohet se si komunikohet me objektet dhe drejtohet me gjendjen e tyre, si dhe të theksohet lidhshmëria e botës reale me botën imagjinare të objekteve në kompjuter. Mënyra e këtillë e ndërveprimit ndërmjet dy botëve është e domosdoshme që programi të ketë kuptim dhe të jetë praktik.

Pyetje dhe detyra kontrolli

1. Cili është dallimi midis klasës dhe instancës?
2. Sqaro çfarë është gjendja e objektit?
3. Çfarë janë atributet dhe metodat e klasës?

Puno vetë

1. Në skenarin e mëposhtëm nga jeta reale identifiko klasat e nevojshme për paraqitjen e informatave të dhëna. Për secilën klasë jepi elementet e saj (atributet dhe metodat). Cilat instanca të klasave të dhëna mund të identifikohen në tekst?

a) Kontrollit të fluturimit në aeroport në Podgoricë i nevojitet paraqitja e të dhënave relevante për aeroplanë dhe pozitat e tyre. Më saktë, për çdo aeroplan është i njohur numri unik i evidencës, marka, tipi dhe viti i prodhimit. Në çdo moment është e njohur nëse është aeroplani në tokë apo ndodhet në hapësirën ajrore, ose është jashtë rrezes të kontrollorit në Podgoricë. Nëse ndodhet në hapësirën ajrore, janë të njohura koordinatat e tij: x, y, z. Kontrollorët e fluturimit në mënyrë periodike (çdo disa sekonda) bëjnë kontrollin e informacioneve mbi koordinatat e aeroplanit dhe evidentojnë aterrimin dhe fluturimin e tij, si dhe daljen nga hapësira ajrore e Podgoricës.

b) Nxënësit të klasës së gjashtë i nevojitet ndihma për vizatimin dhe njehsimet e elementare të figurave të rregullta në rrafsh. Për çdo figurë është i njohur numri i kulmeve (3 – trekëndëshi, 4 – katërkëndëshi, 6 – gjashtëkëndëshi, 8 – tetëkëndëshi) dhe gjatësia e brinjës, në bazë të cilave njehsohen këndi i brendshëm, perimetri dhe sipërfaqja. Krahas vizatimit të figurës përkatëse, duhet të mundësohet vizatimi i figurës me rreth të brendashkruar dhe të jashtëshkruar. Gjithashtu nxënësit duhet t'i mundësohet që mbas ndryshimit të vlerës së gjatësisë së brinjës të rinovojë të dhënat mbi elementet përkatëse të n-këndëshit të dhënë.

c) Për çdo person të punësuar janë të njohur: emri, mbiemri, NUIQ (numri unik i identifikimit të qytetarit), reparti në të cilin punon, vitet e stazhit punues dhe paga (rroga). Ekzistojnë lloje të veçanta të punësuarish: menaxherët dhe punonjësit në zyre. Në rast se i punësuari është menaxher, për të njihet informata shtesë, numri i të punësuarve në repart në të cilin ai është menaxher, si dhe shtesa në pagë. Në anën tjetër, për punëtorin në zyre është e njohur gjatësia e pauzës gjatë kohës së punës që mund ta përdorë, kurse në sasinë e pagës nuk ndikojnë (llogariten) vitet e stazhit të punës, siç vlen për punëtorët e tjerë. Për secilin punëtor nevojitet të sigurohet përftimi i informacioneve mbi rrogën dhe stazhin e punës, si dhe mundësitë e korrigjimeve të tyre.

V.

KLASAT DHE METODAT

Në kapitullin paraprak jeni njoftuar me shembuj të klasave dhe metodave të ndryshme që u përgjigjen shembujve nga jeta e përditshme. Tani do të mësoni se si krijohen klasat dhe metodat në Javë dhe çfarë është e mundur të bëhet me to.

Në këtë kapitull do të mësoni të krijoni:

- Klasat dhe metodat brenda klasave,
- Konstruktorët për klasa dhe objektet me ndihmën e tyre,
- Metodat *get* dhe *set*,
- Metodën *main*.

Kjo do t'u mundësojë që:

- Të shkruani programin që do të simulojë ndërveprimin ndërmjet objekteve po të njëjtës klasë dhe objekteve të klasave të ndryshme,
- Të shkruani programet, që do të përfshijnë konceptet e trashëgimit (d.m.th. përmbajnë hierarkinë e klasave).

5.1. Forma e përgjithshme e klasës

Klasa përkufizohet shumë thjesht – duke përkufizuar të gjitha elementet e saj që e përbëjnë: të dhënat, gjegjësisht ndryshoret (***atributet***) dhe funksionet që me ato të dhëna diçka veprojnë (***metodat***). Edhe pse janë të mundshme klasat që përmbajnë vetëm ndryshoret ose vetëm metodat, shumica e klasave përmbajnë zakonisht ndryshoret dhe metodat.

Forma e përgjithshme e përkufizimit të klasave bëhet nëpërmjet të fjalës së rezervuar *class* në mënyrën e mëposhtme:

Krijimi i klasës

```
class emriKlases {
        tipi_atributit  emri_atributit1;
        tipi_atributit  emri_atributit2;
        // ...
        tipi_atributit  emri_atributitN;

        tipi_vleres_kthyese_metodes  emri_metodes1(lista_parametrave) {
           // trupi i metodës
        }
        // ...
        tipi_vleres_kthyese_metodes  mri_metodesM(lista_parametrave) {
           // trupi i metodës
        }
}
```

Përkufizimi i klasës fillon me fjalën kyçe *class*, dhe mbas hapjes së kllapës së madhe "{" vazhdojnë ndryshoret që do të përkufizohen në atë klasë. ***Atributet*** përkufizohen në mënyrë plotësisht të njëjtë si ndryshoret lokale, gjë që keni mësuar në kapitullin 3.

Është me rëndësi të përmendet se na Javë përdoret rregulla e pashkruar që emrat e klasave të fillojnë me shkronja të mëdha. Kështu klasa dallohet lehtë nga ndryshoret dhe metodat, emrat e të cilave zakonisht shënohen me shkronja të vogla. Gjatë emërtimit të ndryshoreve, metodave etj., për arsye praktike, në vend të shkronjave ë dhe ç do të përdorim shkronjat e dhe c sipas radhës.

Të shkruajmë tani pjesën e kodit për krijimin e klasës *Nxenesi* nga kapitulli paraprak ku kemi përmendur atributet e mëposhtme: *numriEvidences*, *emri*, *mbiemri*, *paralelja*, *emriShkolles* dhe *notaNgaMatematika*.

```
class Nxenesi {
    int     numriEvidences;
    String  emri;
    String  mbiemri;
    int     klasa;
    String  emriShkolles;
    int     notaNgaMatematika;
}
```

Me anë të atributeve mund t'u shoqërohen edhe vlerat fillestare (të nënkuptueshme) gjatë përkufizimit të vetë klasës. Në shembullin tonë mund të përkufizojmë vlerën e nënkuptuar 1 për atributet *numriEvidences* dhe *klasa*, në mënyrën e mëposhtme:

```
class Nxenesi {
       int     numriEvidences=1;
       String  emri;
       String  mbiemri;
       int     klasa=1;
       String  emriShkolles;
       int     notaNgaMatematika;
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/klasa/Nxenesi* në CD.

Në vazhdim japim përshkrimet e projekteve që do të zhvillohen gjatë këtij teksti paralelisht me përvetësimin e elementeve të reja të lëndës.

Përgatitu që në grup të punosh projektin

Projekti *LojtarëtProjekti*. Krijo projektin *LojtarëtProjekti* dhe brenda tij krijo klasën *Lojtari* që përmban:

- atributin *emriMbiemri*, që paraqet emrin dhe mbiemrin e lojtarit;
- atributin *ekipiFutbollit*, që paraqet emrin e ekipit të futbollit për të cilin lojtari luan;
- atributin *pozicioni*, që paraqet pozicionin në ekip në të cilën lojtari luan;
- atributin *mesatarjaSezones*, që paraqet numrin mesatar të golave që lojtari ka qëlluar në sezonin paraprak. Vlera fillestare e këtij atributi është 0.00;
- atributin *nrGolave*, që paraqet numrin e golave që lojtari ka realizuar në sezonin aktual. Vlera fillestare e këtij atributi është 0.00.

Projekti *ManarProjekti*. Krijo projektin *ManarProjekti* dhe brenda tij krijo klasën *Manari* që përmban:

- atributin *lloji*, që paraqet llojin e manarit (kafshë e përkëdhelur në shtëpi). Vlera fillestare e këtij atributi është "qen";
- atributin *raca*, që paraqet racën e manarit. Vlera fillestare është "e panjohur";
- atributin *emri*, që paraqet emrin e manarit;
- atributin *mosha*, që paraqet moshën e manarit të shprehur në muaj.

Projekti *QytetetProjekti*. Krijo projektin *QytetetProjekti* dhe brenda tij krijo klasën *Qyteti* që përmban:

- atributin *emri*, që paraqet emrin e qytetit;
- atributin *nrPostar*, që paraqet numrin postar të qytetit;
- atributin *nrBanorve*, që paraqet numrin e banorëve të atij qyteti;
- atributin *kampusiUniversitar* vlera fillestare e të cilit është `false`. Vlera e këtij atributi është `true` në qoftë se qyteti ka kampusin universitar.

5.2. Krijimi i metodave brenda klasës

Metodat përcaktojnë se si disa objekte të caktuara sillen, çfarë ndodh me to gjatë krijimit dhe cilat veprime i kryejnë. Metodat përkufizohen në kuadër të trupit të klasës në bazë të sintaksës së mëposhtme:

Metodat e klasës

```
tipi_vleres_kthyese_metodes emri_metodes(lista_parametrave) {
    // trupi i metodës
}
```

Këtu ***tipi_vleres_kthyese_metodes*** përcakton tipin e të dhënave që i kthen metoda. Mund të jetë secili tip i të dhënave (mbi të cilat kemi folur në kapitullin III), duke kyçur edhe tipat e klasave që i krijon vetë programuesi. Në qoftë se metoda nuk kthen një vlerë të caktuar, tipi i saj i të dhënave kthyese duhet të shënohet me ***void***.

Emri i metodës është përcaktuar me identifikuesin ***emri_metodes***. Mund të jetë cilido identifikues i përkufizuar në pajtim me rregullat e përmendura në kapitulli III. P.sh. në rastin e metodave për klasën *Nxenesi*, që e kemi përmendur më herët në shembull, emrat e tyre mund të jenë, sipas radhës: *shenoNotenMatematika*, *regjistrimiNxenesi*...

Lista e parametrave përmban vargun e çifteve ***tipi_parametrit emri_parametrit*** të ndarë me presje. Parametrat, janë në esencë, ndryshore që marrin vlerën e argumenteve të dërguar metodës në momentin e thirrjes së saj. Në qoftë se metoda nuk ka parametra, lista e parametrave do të jetë e zbrazët, prandaj ceken vetëm kllapat e zbrazëta.

Kur tipi i të dhënave kthyese nuk është void, metoda kthen vlerën e cila në një moment të caktuar, zakonisht në fund të ekzekutimit të metodës, jepet me ndihmën e komandës ***return*** në trajtën:

return vlera;

Në shembullin e klasës *Nxenesi* do të njoftohemi me mundësitë e përmendura gjatë përkufizimit të metodave të ndryshme. Klasës *Nxenesi* i shtojmë:

- metodën *peseMatematika*, që nxënësit i evidenton notën 5 nga matematika;
- metodën *shenoNotenMatematika*, që nxënësit i evidenton notën e re nga matematika, që dorëzohet si argument hyrës i metodës;
- metodën *paraqitNotenMatematika*, e cila me një mesazh përkatës paraqet notën nga matematika;
- metodën *ktheNotenMatematika*, që kthen notën aktuale nga matematika.

Emrat e metodave nuk mund duhet të përputhen me emrat e klasave, atributeve ose ndryshoreve në bllokun e programit në kuadër të të cilit ndodhet edhe metoda. Mirëpo, së shpejti do të shihni që brenda një klase mund të ndodhen më shumë metoda me emër të njëjtë, mirëpo me parametra hyrës të ndryshëm. Kjo mundësi në Javë është e njohur me emrin metodë e ***përputhjes*** (***mbivendosja***) (anglisht: *overriding*), mbi të cilën do të flasim në vazhdim.

Projekti i tërë ndodhet në dosjen *Teksti/src/KrijimiMetodaveBrendaKlasave/Nxenesi* në CD.

```
class Nxenesi {
    int     numriEvidences = 1;
    String  emri;
    String  mbiemri;
    int     klasa = 1;
    String  emriShkolles;
    int     notaNgaMatematika;

    void peseMatematika () {      // metoda për shënimin e notës 5 nga matematika
        notaNgaMatematika = 5;
    }

    void shenimiNotaMatematika (int nota) { // metoda për shënimin e notës  nga matematika
        notaNgaMatematika  = nota;
    }

    void paraqitNotenMatamatika () {
        System.out.println("Nota nga matematika " + notaNgaMatematika);
    }

    int ktheNotenMatematika() {  // metoda për kthimin e notës nga matematika
        return notaNgaMatematika;
    }
}
```

Mbi pajtueshmërinë e tipave kemi folur në kapitullin III. P.sh. `int` është i pajtueshëm me `long` (*gjithmonë është e mundshme që vlera e ndryshores së tipit* `int` *t'i shoqërohet ndryshores së tipit* `long`), kurse `float` është i pajtueshëm me `double`.

Klasa kryesore, *Objekt*, përkufizon disa metoda që janë në dispozicion të të gjitha objekteve në Javë (që trashëgojnë klasën *Object*). Metodat në fjalë janë: *clone*, *equals*, *finalizo*, *getClass*, *hashCode*, *notify*, *notifyAll*, *toString* dhe *wait*. Disa nga këto metoda do t'i njoftojmë më hollësisht në kapitujt e mëposhtëm, mirëpo është me rëndësi që emrat e përmendur të evitohen, përveç në rastet kur qëllimisht dëshironi të përforconi njërën prej atyre metodave (rasti i shpeshtë me metodën *toString* – kapitulli VII).

Metoda *peseMatematika* shëno për vlerën e atributit *notaNgaMatematika* numrin 5, prandaj ka tipin `void` dhe nuk ka argumente hyrëse.

Metoda e mëposhtme *shenoNotenMatematika* ka gjithashtu tipin vod, mirëpo ka argumentin hyrës – notën që duhet t'i shoqërohet atributit *notaNgaMatematika*. Vërejtje e rëndësishme është që parametri *nota* duhet të ketë tip të njëjtë (ose pajtueshmëri) si dhe atributi *notaNgaMatematika*, në të kundërtën paraqitet gabimi.

Metoda *shenoNotenMatematika* vetëm paraqet vlerën aktuale të atributit *notaNgaMatematika*, duke thirrur *System.out.println*, prandaj nuk ka parametra hyrës as vlera kthyese.

Në fund, metoda *ktheNotenMatematika* kthen notën aktuale, d.m.th. vlerën e atributit *notaNgaMatematika*. Është me rëndësi të përmendet se tipi kthyes i metodës duhet të jetë i njëjtë (ose i pajtueshëm) si edhe tipi i atributit, vlera e të cilit kthehet, në të kundërtën paraqitet gabimi.

Pyetje dhe detyra kontrolli

1. Cila është forma e përgjithshme e krijimit të klasës?
2. Çfarë do të ndodhë nëse tek përkufizimi i klasës në vend të fjalës kyçe *class* vendoset fjala *Class*?
3. Cila është forma e përgjithshme e krijimit të metodave?
4. Çfarë domethënie ka fjalia "metoda ka vlerën kthyese të tipit `void`"?
5. A mund të përkufizohen më shumë metoda me emër të njëjtë brenda një klase?

Përgatitu që në grup të punosh projektin

Projekti *LojtarëtProjekti*. Në klasën e krijuar më herët *Lojtari* shto:

- Metodën *ndryshimiEkipit*, që për argument të ri ka vlerën e re të ekipit në të cilin lojtari luan. Metoda vlerën e dhënë e vendos si vlerë të re të atributit *ekipiFutbollit* dhe jep informatën mbi ndryshimin e kryer në formën:

  '*Lojtari Emri_Mbiemri* ka kaluar në ekipin e ri *Emri_ekipit*'.

- Metodën *zmadhoMesataren*, e cila si argument hyrës pranon vlerën me të cilën duhet zmadhuar numri mesatar igolave të lojtarëve, d.m.th. vlerën e atributit *mesatarjaSezones*.
- Metodën *goliRi*, që evidenton golin e posa shënuar duke zmadhuar numrin e përgjithshëm të golave të shënuar në sezon (atributi *nrGolave*).
- Metodën *printimi*, që i paraqet të dhënat mbi lojtarin në një rresht.

Projekti *ManarProjekti*. Në klasën *Manari* të krijuar më herët shto:

- Metodën *ndryshoMoshen*, e cila si parametër hyrës pranon moshën e manarit që duhet vendosur si vlerë të re të atributit *mosha*;
- Metodën *zmadhoMoshen*, që e zmadhon moshën e manarit për një muaj;
- Metodën *printimi*, që i paraqet të dhënat mbi manarin dhe racën, nga një vlerë në rresht.

Projekti *QytetetProjekti*. Në metodën e krijuar më parë *Qyteti* shto:

- Metodën *nderrimiNrPostar*, e cila për shkak të ndryshimeve në tërë shtetin e ndërron numrin postar në numrin që paraqitet si argument hyrës i metodës.
- Metodën *behetKampusiUniversitar*, e cila evidenton (ose përforcon) se qyteti ka kampusin universitar.
- Metodën *zmadhoNrBanoreve*, e cila numrin e banorëve e zmadhon për vlerën e argumentit hyrës dhe tregon mesazhin mbi vlerën e re.
- Metodën *printimi*, që paraqet të dhënat mbi qytetin dhe numrin e përgjithshëm të banorëve në një rresht.

5.3. Krijimi i objekteve dhe puna me objektet

Më parë kemi sqaruar se klasat paraqesin vetëm shabllone për krijimin e objekteve, d.m.th. objekti paraqet shembull (instancë) të një klase përkatëse. Në Javë objektet deklarohen në mënyrë të ngjashme si dhe ndryshoret. Në fillim jepet emri i klasës, pastaj emri i objektit konkret:

***emriKlases emriObjektit*;**

Është me rëndësi të përmendet se deklarimi **nuk formon** objektin, por vetëm deklaron referencën që referohet në objekt të klasës përkatëse. Që objekti të mund të përdoret (të thirren metodat e tij, të ndryshohen vlerat e atributeve, etj.), ai fillimisht duhet të krijohet nëpërmjet të operatorit ***new*** në mënyrën e mëposhtme:

***emriKlases* = *new emriObjektit*();**

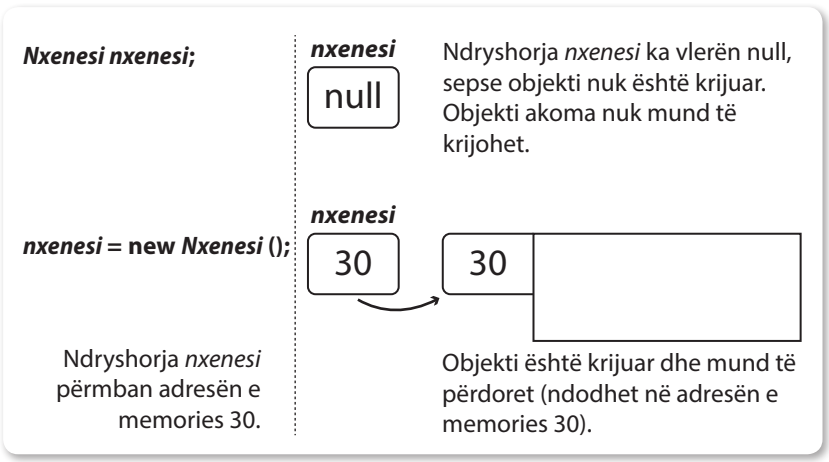


Figura 5.1. Deklarimi dhe krijimi i objektit

Objekti i klasës

Në praktikë përdoret më shpesh krijimi i objektit me deklaratë, në mënyrën e mëposhtme:

***emriKlases emriObjektit = new emriKlases*();**

që është e njëvlershme me:

emriKlases emriObjektit;
***emriObjektit = new emriKlases*();**

Në shembullin tonë të klasës *Nxenesi*:

```
Nxenesi nxenesi = new Nxenesi();
// formohet objekti i tipit Nxënës me emrin nxenesi.
```

Mbasi që komanda paraprake të ekzekutohet, objekti ***nxenesi*** do të jetë një mostër e klasës ***Nxenesi***. Fjala është për objektin që do të ekzistojë "fizikisht". Në këtë mënyrë, prej një klase mund të formohet numri i dëshiruar i objekteve, si në shembullin e mëposhtëm.

```
Nxenesi nxenesi1 = new Nxenesi();
Nxenesi nxenesi2 = new Nxenesi();
```

Gjatë secilit krijim të objektit të ri ai i përfton kopjet e veta të atributeve të përkufizuar me klasën. Prandaj, çdo objekt i tipit Nxenesi, do të përmbajë kopjet e veta të atributeve *emrin*, *mbiemrin*, *numriEvidences*, *emriShkolles*, *paralelja*, *notaNgaMatematika*. Që t'i qasemi këtyre atributeve përdoret operatori pikë (.), në mënyrën e mëposhtme:

***emriObjektit.emriAtributit*;**

Operatori pika e lidh emrin e objektit me emrin e atributit të instancës (mostrës).

Kështu mund të ndryshoni vlerat e notave nga matematika për të dy nxënësit në mënyrën e mëposhtme:

```
nxenesi1.notaNgaMatematika = 5;
nxenesi2.notaNgaMatematika = 3;
```

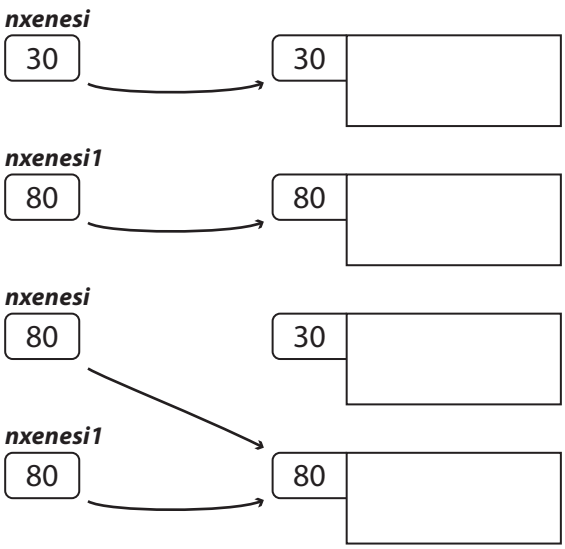
Kjo komandë i kumton përkthyesit që kopjes së atributit *notaNgaMatematika*, që ndodhet brenda objektit ***nxenesi1***, t'i shoqërohet vlera 5, kurse kopjes së atributit *notaNgaMatematika*, që ndodhet brenda objektit ***nxenesi2***, t'i shoqërohet vlera 3. Është me rëndësi të theksohet se ndryshimi i atributit të një objekti nuk ndikon në asnjë mënyrë në atributin me emër të njëjtë të objektit tjetër.

Referenca në objektin *nxenesi* mund të përftojë referencën në cilindo objekt që i takon klasës *Nxenesi* (dhe klasave që janë nxjerrë nga klasa *Nxenesi*, me çfarë do t'ju njoftojmë më poshtë). P.sh.

Nxenesi nxenesi = new Nxenesi();

Nxenesi nxenesi1 = new Nxenesi ();

nxenesi = nxenesi1;



Duhet të theksohet se *nxenesi* përfton referencën në objekt, në të cilën tregon *nxenesi1*. Nuk bëhet kopje e veçantë për *nxenesi* që ka vlerat si atributet në të cilat tregon *nxenesi1*.

Për objektet *nxenesi* dhe *nxenesi1* thuhet se janë ***objekte referuese***, sepse ata përmbajnë referencën, gjegjësisht adresën e objektit.

Për fshirjen e objektit mjafton që ndryshorja, e cila tregon në të, të thirret me komandën null (shiko figurën 5.2) ose të krijohet një objekt tjetër në të cilin do të tregojë po ajo ndryshore (rishtas të shikohet shembulli paraprak në të cilin është fshirë objekti *nxenesi*). Lokacioni në memorie në të cilën nuk tregon asnjë tregues (pointer) do të lirohet së shpejti nëpërmjet mekanizmit "garbage collection", siç është paraqitur në figurën 5.2.

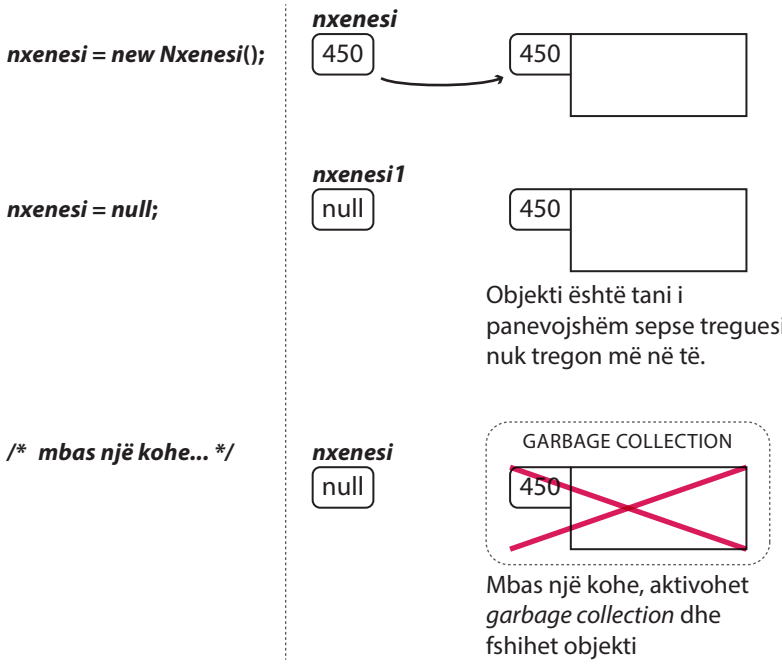


Figura 5.2. Fshirja e objekteve në Javë

Cili mekanizëm te Java siguron zhdukjen e objektit që nuk përdoret më?

Mekanizmi i përmendur quhet ***garbage collection*** (grumbullim mbeturinash), që punon sipas principit të mëposhtëm: analizohen objektet e programit dhe kërkohen ato objekte në të cilat nuk tregojnë më objektet referente. Koha e nisjes së mekanizmit të përmendur nuk është i lidhur me daljen e objektit nga fusha e rrethit të vet, por kjo bëhet kohë mbas kohe dhe nuk ka kontroll të drejtpërdrejtë të atij që përpunon programin.

Shembull. Që të demonstrojmë punën me objekte, të shkruajmë një program të tërë (që përmban metodën *main*, me detaje të të cilës do të njiheni pak më vonë). Në program nevojitet:

- të krijohen dy objekte të klasës *Nxenesi* për paraqitjen e të dhënave mbi nxënësit: Mark Markaj, nota 5 nga matematika dhe Ana Malaj, nota 4 nga matematika;
- të paraqiten të dhënat mbi të gjithë nxënësit e një rreshti;
- duke e thirrur metodën e krijuar më herët *shenoNoten* të shënohen notat e reja nga matematika, Markut nota 4, kurse Anës nota 5.
- të njehsohet vlera mesatare e notave nga matematika të nxënësve Markut dhe Anës dhe të paraqitet në daljen standarde me mesazh përkatës;
- Të supozojmë se gjatë hyrjes është bërë gabimi dhe notat e Markut dhe të Anës janë ndërruar. Të shënohet komanda që do të përmirësojë gabimin dhe si rezultat të tregojë notën e re të Anës duke thirrur metodën e krijuar më herët *ktheNotenMatematika*.

Projekti i tërë ndodhet në dosjen *Teksti/src/KrijimiKlasave/Nxenesi* në CD.

```
public class TestNxenesi {
   public static void main(String[] args) {
      Nxenesi nxenesi1 = new Nxenesi(); // krijimi i objekteve
      Nxenesi nxenesi2 = new Nxenesi();

      nxenesi1.emri = "Mark"; // hyrja e të dhënave per nxënësit

      nxenesi1.mbiemri = "Markaj";
      nxenesi1.notaNgaMatematika = 5;

      nxenesi2.emri = "Ana";
      nxenesi2.mbiemri = "Malaj";
      nxenesi2.notaNgaMatematika = 4;

      System.out.println("Nxenesi: " + nxenesi1.emri + " " + nxenesi1.mbiemri
              + "ka notën nga matematika: " + nxenesi1.notaNgaMatematika);
      // printimi i të dhënave për nxënësin

      System.out.println("Nxenesi: " + nxenesi2.emri + " " + nxenesi2.mbiemri
              + " ka notën nga matematika: " + nxenesi2.notaNgaMatematika);

      nxenesi1.shenimiNotaMatematika(4); // thirrja e metodës
      nxenesi2.shenimiNotaMatematika(5);

      int shuma;
      shuma = nxenesi1.ktheNotenMatematika() + nxenesi2.ktheNotenMatematika();
      // shuma e notave nga matematika
      double mesatarja;
      mesatarja = shuma / 2; // njehsimin e mesatares, printimi
      System.out.println("Nota mesatare nga matematika është " + mesatarja);

      /*  Nota e Anës (nxenesi2) dhe Markut (nxenesi1) do t'i ndërrojmë në mënyrën e mëposhtme:
         1. Ndryshores notaA i shoqërojmë vlerën e notës së Anës
         2. Nota e re të Anës e merr vlerën e notës së Markut
         3. Nota e re e Markut e merr vlerën e notës paraprake të Anës
            (të cilën e kemi mbajtur në mend me emrin notaA) */

      int notaA;
      notaA = nxenesi2.ktheNotenMatematika();

      nxenesi2.shenimiNotaMatematika(nxenesi1.ktheNotenMatematika());
      nxenesi1.shenimiNotaMatematika(notaA);

      System.out.println("Nota e re e Anës është: "
            + nxenesi2.ktheNotenMatematika()); // printimi i notës
   }
}
```

Në fillim kemi krijuar dy objekte *nxenesi1* dhe *nxenesi2*, të cilëve u kemi vendosur vlerat e atributeve përkatëse për *emri*, *mbiemri* dhe *notaNgaMatematika*. Është me rëndësi të theksohet se ndryshoret *nxenesi1* dhe *nxenesi2*, që paraqesin dy objekte të ndryshme, duhet të jenë të ndryshme, d.m.th. në rast se së pari kemi krijuar objektin *nxenesi1* për nxënesin Mark Markaj dhe pastaj kemi përdorur të njëjtën ndryshore për objektin që paraqet nxënësën Anën Malaj, të dhënat për Markun do të fshihen.

Komandat pasuese janë për paraqitjen e të dhënave për nxënësit: në tekst "Nxenesi" me ndihmën e operatorit + është lidhur emri i njërit prej nxënësve (p.sh. *nxenesi1.emri*), pastaj një karakter i zbrazët (një space), dhe mbas është shtuar mbiemri dhe teksti "ka notën nga matematika: " dhe nota e tij nga matematika (vlera numerike). Të theksojmë se rezultati i njëjtë ka mundur të arrihet duke përdorur disa komanda ***System.out.println*** që do kishin paraqitur vlerat e secilit atribut një mbas tjetrit pa kalimin në rreshtin e ri.

Në rreshtin vijues thirret funksioni për shënimin e notës *shenoNotenMatematika* e tipit `void`, që nuk ka vlera kthyese, kurse si argument hyrës pranon notën që duhet shënuar. Prandaj, thirrjen e metodës e kryejmë duke cekur emrin e objektit që ndërlidhet në të dhe mbas operatorit (`.`) jepet emri i metodës. Parametri hyrës i metodës jepet brenda kllapave. Si parametër hyrës ka mundur të përdoret edhe vlera e një ndryshoreje dhe në atë rast brenda kllapave gjatë thirrjes së metodës jepet emri i ndryshores, si në shembullin e mëposhtëm:

```
int nota = 5;
nxenesi1.shenoNotenMatematika(nota);
```

Domethënë, thirrja *nxenesi1.shenoNotenMatematika*(4) e fton metodën *shenoNotenMatematika* të përkufizuar me objektin *nxenesi1*, kurse *nxenesi2.shenoNotenMatematika*(5) e fton metodën *shenoNotenMatematika* të përkufizuar me objektin *nxenesi2*, pavarësisht nga objekti *nxenesi1*.

Kur kryhet komanda *nxenesi1.shenoNotenMatematika*(4), sistemi ekzekutiv i Javës e dorëzon kontrollin e kodit të përkufizuar brenda metodës *shenoNotenMatematika*() brenda objektit *nxenesi1*. Mbas që ekzekutohen komandat brenda metodës *shenoNotenMatematika*(), kontrolli i kthehet ftuesit të metodës, kurse ekzekutimi i programit vazhdon nga rreshti i programit që vijon mbas thirrjes.

Më tutje, për njehsimin e vlerës mesatare të notës së Markut dhe Anës nga matematika nevojitet të njehsojmë shumën e notave të tyre dhe ta pjesëtojmë me 2. Kështu, përkufizojmë dy ndryshore: *shuma* dhe *mesatarja*, ndryshoren e parë të tipit `int` (tipit të njëjtë si atributi *notaNgaMatematika*), kurse ndryshoren e dytë të tipit `double` (sepse gjatë pjesëtimit me 2, rezultati është numër `real`). Ndryshorja *shuma* duhet të paraqesë shumën e vlerave të notave, ashtu që nëpërmjet.

***nxenesi1.ktheNotënMatematika*() + *nxenesi2.ktheNotënMatematika*()**

për të dy objektet thirret metoda *ktheNotenMatematika*() që i kthen (jep si rezultat) notat e tyre nga matematika dhe i mbledh. Komandat e ardhshme njehsojnë vlerën mesatare dhe e paraqesin në daljen standarde.

Për ndërrimin e vlerave të notave nga matematika të Markut dhe të Anës (Marku i paraqitur me objektin *nxenesi1* ka notën 4, kurse Ana e paraqitur me objektin *nxenesi2* e ka notën 5) nevojitet të kryhen veprimet e mëposhtme:

- Të mbahet në mend nota e Anës në një ndryshore ndihmëse (në rastin tonë bëhet fjalë mbi ndryshoren *notaA* që përfton vlerën 5 si vlerë kthyese të metodës *ktheNotenMatematika*() të objektit *nxenesi2*).

- Nota e re e Anës është nota aktuale e Markut: duke thirrur metodën *shenoNotenMatematika*() mbi objektin *nxenesi2* është vendosur vlera e re e atributit *notaNgaMatematika*. Argumenti hyrës i kësaj metode është nota e Markut, vlerën e të cilës e merr nota e Anës, dhe ky është rezultati i metodës *ktheNotenMatematika*() mbi objektin *nxenesi1* (në rastin tonë bëhet fjalë për notën 4). Vlera e notës së Markut ka mundur të ruhet në një ndryshore ndihmëse, p.sh. *notaM*, e cila mbas kësaj dorëzohet si argument hyrës; d.m.th.

```
nxenesi2.shenoNotenMatematika(nxenesi1.ktheNotenMatematika());
```

është ekuivalente (e njëvlershme) me

```
int notaM = nxenesi1.ktheNotenMatematika();
nxenesi2.shenoNotenMatematika(notaM);
```

- Nota e re e Markut merr vlerën e notës së vjetër të Anës, që e kemi mbajtur në mend paraprakisht (në ndryshoren *notaA* të cilës i është shoqëruar vlera 5 si nota paraprake e Anës nga matematika).

Është me rëndësi të përmendet, se në rast se paraprakisht nuk është mbajtur në mend nota e Anës, dhe nëse nota e re e saj është ajo e Markut, do të zhduket informacioni mbi notën paraprake të Anës dhe nuk do të mund t'i shoqërohet Markut. Kështu me komandat e mëposhtme, Anës i shoqërohet nota e Markut, kurse nota e Markut mbetet e pandryshuar:

```
nxenesi2.shenoNotenMatematika(nxenesi1.ktheNotenMatematika());
nxenesi1.shenoNotenMatematika(notaA);
```

Paraqitja ilustruese e gjendjes në memorie mbas ekzekutimit të komandave të përmendura është paraqitur në tabelën e mëposhtme.

	nxenesi1 emri: Mark mbiemri: Markaj notaNgaMatematika: 5

`nxenesi1.emri = "Mark";`
`nxenesi1.mbiemri = "Markaj";`
`nxenesi1.notaNgaMatematika = 5;`	

	nxenesi2 emri: Ana mbiemri: Malaj notaNgaMatematika: 4

`nxenesi2.emri = "Ana";`
`nxenesi2.mbiemri = "Malaj";`
`nxenesi2.notaNgaMatematika = 4;`	

	notaA
`int notaA;`	
	notaA 4
`notaA = nxenesi2.ktheNotenMatematika();`	
	nxenesi2 emri: Ana mbiemri: Malaj notaNgaMatematika: 5
`nxenesi2.shenoNotenMatematika(nxenesi1.ktheNotenMatematika());`	
	nxenesi1 emri: Mark mbiemri: Markaj notaNgaMatematika: 4
`nxenesi1.shenoNotenMatematika(notaA);`	

Në fund, vlera e notës së Anës paraqitet në daljen standarde duke thirrur *System.out.println* të cilës si parametër hyrës i jepet mesazhi "Nota e re e Anës është: " dhe vlera kthyese e metodës *ktheNotenMatematika*().Vlera kthyese e metodës *ktheNotenMatematika*() mund të ruhet në një ndryshore (p.sh. *ndryshNota*) dhe kjo t'i dorëzohet *System.out.println*, d.m.th.

```
System.out.println("Nota e re e Anës është: " +
                   nxenesi2.ktheNotenMatematika());
```

është ekuivalente me

```
int ndryshNota = nxenesi2.ktheNotenMatematika();
System.out.println("Nota e re e Anës është: " + ndryshNota);
```

Pyetje dhe detyra kontrolli

1. Cili është dallimi ndërmjet klasës dhe objektit?
2. Çfarë është rezultati i deklarimit: `Nxenesi nxenesi = new Nxenesi ();` ?
3. Si u qasemi elementeve të objekteve (të jepet një shembull)?
4. A mund të krijohen më shumë objekte (*nxenesi1*, *nxenesi2*) në mënyrën e mëposhtme:

```
Nxenesi nxenesi1, nxenesi2 = new Nxenesi();
...
nxenesi1.shenoNotenMatematika(5);
```

Çfarë do të jetë rezultati i ekzekutimit të komandës së fundit?

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Krijo klasën *TestLojtari* që përmban vetëm metodën *main* e cila:

- Krijon dy objekte për paraqitjen e të dhënave mbi lojtarët: Lionel Messi, Barcelona, numri mesatar i golave 1,6 dhe David Bekam, Milan, numri mesatar i golave 1,03.
- I paraqet të dhënat mbi lojtarët: emrin, mbiemrin dhe emrin e ekipit në një rresht, kurse numrin mesatar të golave në rreshtin tjetër.
- Mesi mori një ofertë më të mirë në ekipin e futbollit Real Madrid, të cilën e pranoi. Duke thirrur metodën përgjegjëse të evidentohet kalimi i tij në ekipin e ri.

Projekti *ManarProjekti*. Krijo klasën *TestManari* që përmban vetëm metodën *main* e cila:

- I krijon dy objekte për paraqitjen e të dhënave mbi macen Xheri me moshë një vit e gjysmë dhe qenin Arçi me moshë një vit.
- I tregon të dhënat mbi manarët paraprakë.
- Duke thirrur metodën *zmadhoMoshen* evidenton që kanë kaluar dy muaj dhe paraqet moshën e re të manarëve.

Projekti *QytetetProjekti*. Krijo klasën *TestQyteti* që përmban vetëm metodën *main* e cila:

- Krijon objektet për paraqitjen e të dhënave mbi qytetet: Podgorica që ka 139 100 banorë, ka kampusin universitar dhe numrin postar 81 000; Bijello Pole, 17 100 banorë, nuk ka kampusin universitar.
- Nga Bijello Pole janë shpërngulur 151 banorë në Podgoricë, që duhet të evidentohet.
- I paraqet të dhënat mbi numrin e banorëve të Podgoricës dhe Bijello Poles dhe për sa dallojnë ata numra.

5.4. Kontrolli i qasjes (*public, private, protected*)

Një prej përparësive themelore të përdorimit të objekteve manifestohet në faktin se objekti nuk ka detyrim të zbulojë të gjitha atributet dhe sjelljet (metodat). Në softuerin e projektuar mirë të OO, objekti duhet të zbulojë vetëm detajet e domosdoshme për komunikim, kurse ato që nuk janë të rëndësishme për përdorim duhet të fshehen nga objektet e tjera. Kjo quhet ***enkapsulacion***. Për shembull, objekti që njehson katrorin e një numri duhet të sigurojë metodat për përftimin e rezultatit. Mirëpo, atributet e brendshme dhe algoritmet, që përdoren për njehsimin e katrorit, nuk duhet të jenë në dispozicion të objektit që bën thirrjen.

Kjo sigurohet me kontrollin e qasjes së elementeve të klasës që jepen gjatë deklarimit. Specifikuesit e qasjes së Javës janë ***public*** (publik), ***private*** (privat) dhe ***protected*** (i mbrojtur). Specifikimi i protected ka kuptim vetëm gjatë trashëgimit. Përshkrimi i të gjitha specifikimeve është dhënë në tabelën 5.1.

Tabela 5.1. Nivelet Java të qasjes

Niveli i qasjes	Modifikuesi i qasjes (fjala kyçe)	Përshkrimi
Publik	`public`	Klasa, atributi ose metoda që janë të shënuara në këtë mënyrë janë të dukshme kudo dhe kanë qasje të qartë.
Privat	`private`	Atributi ose metoda që janë shënuar në këtë mënyrë janë të dukshëm në kuadër të klasës në të cilën janë shënuar.
I nënkuptuari (me paketë)	(nuk shkruhet asgjë)	Klasa, atributi ose metoda që janë të shënuar në këtë mënyrë janë të dukshëm vetëm në kuadër të paketës në të cilën janë shënuar.
I mbrojturi	`protected`	Është i njëjtë si niveli i paketës i qasjes, vetëm që dukshmëria zgjerohet në të gjitha klasat nga paketat e tjera që trashëgojnë klasën e cila është e mbrojtur ose përmban elementin mbrojtës.

Specifikuesi i qasjes i paraprin specifikimit të tipit të anëtarit. Me të duhet të fillojë deklarimi i anëtarit të klasës.

Për shembull:

public int i;

private double j;

private int metodaIme(int a, char b) { //…}

Që të mund të kuptojmë rezultatet e qasjes publike dhe private, të veprojmë si më poshtë:

Shembulli 2. Klasës *Nxenesi* i shtojmë atributin publik *username* dhe atributin privat *password*, si dhe metodat *kthePaswword* dhe *shenoPasword* për kthimin dhe shënimin e vlerës së atributit *password*.

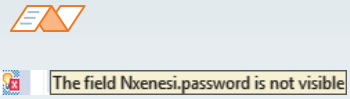
Më tutje, përkufizojmë klasën *testDrejtaQasjes* e cila përmban vetëm metodën *main*. Brenda metodës *main* të krijohet objekti që paraqet nxënësin Mark Markaj, *username* i të cilit është mark.Markaj, kurse *password*-i XXX.

Të përpiqemi t'i qasemi *password*-it të nxënësit Mark Markaj.

```
class Nxenesi {

    String         emri;
    String         mbiemri;
    public String  username;
    private String password;

    String kthePassword() {
        return password;
    }

    void shenoPassword(String newPassword) {
        password = newPassword;
    }
}

class TestDrejtaQasjes {

    public static void main(String[] args) {

        Nxenesi nxenesi = new Nxenesi();

        nxenesi.emri  = "Mark"; /* atributeve emri, mbiemri dhe username
                                   mund t'u qaset drejtpërdrejt */

        nxenesi.mbiemri = "Markaj";
        nxenesi.username = "mark.markaj";

        nxenesi.password = "XXX"; /* kjo komandë nuk është në rregull,
                                     do të shkaktojë gabim */
```

Çdo klasë që fillon me fjalën kyçe ***public*** duhet të ruhet në skedar (fajll) që ka emrin e njëjtë si dhe klasa si dhe ekstensionin *.java*.

Deklarimi i më shumë se një klase `public` në të njëjtin skedar sjell deri te gabimi gjatë përkthimit të programit.



Kompajleri Java na ka tërhequr vëmendjen se ekziston gabimi, kurse për llojet e gabimeve dhe deklarimin e tyre automatik do të flitet në kapitullin X.

```
        nxenesi.shenoPassword("XXX"); /* ndryshores password duhet t'i
                                         qaset nëpërmjet metodës
                                         shënoPassword  */

        System.out.println("Nxenesi: " + nxenesi.emri + " "
                           + nxenesi.mbiemri + " "
                           + nxenesi.username + " "
                           + nxenesi.kthePassword());
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/KontrolliQasjes/Nxenesi* në CD.

Nga shembulli i dhënë shihet se atributit privat *password* nuk mund t'i qaset drejtpërdrejtë, por vetëm nëpërmjet metodave përkatëse *kthePassword* dhe *shenoPassword*.

5.4.1. Konstruktorët

Konstruktorët

Kemi cekur më herët se gjatë inicializimit të objektit të ri rezervohet hapësirë në memorie në të cilën do të vendoset objekti i klasës së dhënë dhe vlerat e inicializuara të atributeve të tyre. Në rastin e përgjithshëm, mund të jetë shumë e lodhshme të inicializohen vlerat e të gjitha atributeve në vlerat e dëshiruara, prandaj në Javë lejohet përdorimi i ***konstruktorëve***, detyra e të cilëve është pikërisht inicializimi i atributit instance.

Konstruktori është sipas mënyrës së shënimit i ngjashëm me metodën, sepse mund të përmbajë komanda të çfarëdoshme, mirëpo, për dallim nga metoda, nuk mund të thirret drejtpërdrejt, por këtë e bën Java në mënyrë automatike gjatë krijimit të objektit të ri me komandën `new`.

Konstruktorët në pamje të parë duken shumë të çuditshëm, sepse krahas tyre nuk jepet tipi privat i të dhënave, madje as tipi `void`. Ai tip i të dhënave nuk është i nevojshëm, sepse konstruktorët në të vërtetë kthejnë tipin e të dhënave që i përgjigjet vetë klasës.

Me fjalë të tjera, ***detyra e konstruktorit*** është që të inicializojë objektin në procesin e krijimit të tij, d.m.th. mbas komandës new menjëherë ekziston objekti i gatshëm për përdorim.

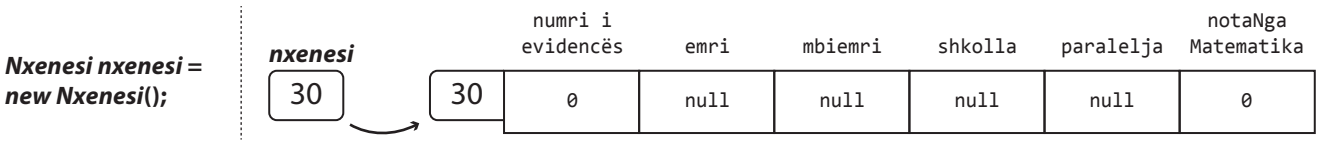
Gjithashtu që Java të njohë konstruktorin, ai duhet të ketë emrin e njëjtë si edhe vetë klasa. Klasa mund të ketë më shumë konstruktorë dhe ata dallohen vetëm sipas parametrave hyrës (numrit dhe/ose tipit).

Konstruktorët e nënkuptuar

Në qoftë se konstruktori nuk shënohet në mënyrë eksplicite, Java krijon ***konstruktorin e nënkuptuar*** i cili nuk pranon asnjë parametër hyrës dhe u shoqëron atributeve, vlerat fillestare të nënkuptuara, të cilat për tipat primitive `byte`, `char`, `short`, `int`, `long`, `float` dhe `double` janë të barabarta me 0, për tipin `boolean` është `false`, kurse për tipat e referencës vlera është `null`.

Në shembullin tonë të klasës *Nxenesi*, rezultati i krijimit të objektit duke thirrur konstruktorin e nënkuptuar është paraqitur në figurë.

```
class Nxenesi {
    int    numriEvidences;
    String emri;
    String mbiemri;
    String shkolla;
    String paralelja;
    int    notaNgaMatematika;
}
```



Shembulli 3. Të ndryshohet kodi paraprak për krijimin e klasës *Nxenesi*, ashtu që të përmbajë konstruktorin i cili për vlerat e nënkuptuara të atributeve *emri* dhe *mbiemri* të vendosë "i panjohur", për atributin *shkolla* të vendosë "Gjimnazi", kurse për atributin *paralelja* "IV 1". Vlerat e nënkuptuara të *numriEvidences* dhe *notaNgaMatematika*, janë si më herët, 1 dhe 5 (sipas radhës).

```
class Nxenesi {
    int    numriEvidences;
    String emri;
    String mbiemri;
    String shkolla;
    String paralelja;
    int    notaNgaMatematika;

    public Nxenesi() { // konstruktori
        numriEvidences = 1;
        emri = "i panjohur";
        mbiemri = "i panjohur";
        shkolla = "Gjimnazi";
        paralelja = "IV-1";
        notaNgaMatematika = 5;
    }
}
```

Ngjashëm si metodat, konstruktorët gjithashtu mund të pranojnë parametrat, d.m.th. argumentet hyrëse. Një prej mangësive të qarta të shembullit paraprak është që të gjithë nxënësit gjithmonë kanë pasur emrin e njëjtë të evidencës dhe notën 5 nga matematika. Prandaj mund të formojmë konstruktorin që pranon parametrat (të ashtuquajturit ***konstruktorët parametrik***):

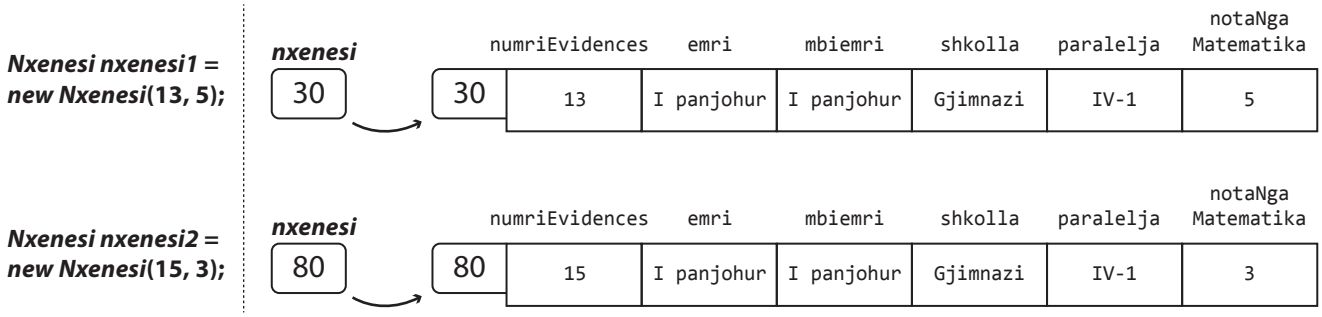
Konstruktori parametrik

```
class Nxenesi  {
   int    numriEvidences;
   String emri;
   String mbiemri;
   String shkolla;
   String paralelja;
   int    notaNgaMatematika;
```

```
   public Nxenesi(int nrEv, int notaMat) {   // konstruktori
      numriEvidences = nrEv;
      emri  = "i panjohur";
      mbiemri  = "i panjohur";
      shkolla  = "Gjimnazi";
      paralelja  = "IV-1";
      notaNgaMatematika  = notaMat;
   }
}
```

kurse më vonë gjatë krijimit të objekteve do të përmendim menjëherë vlerat fillestare të atributeve të dhëna të instancave të caktuara:

```
Nxenesi nxenesi1 = new Nxenesi(13, 5);
Nxenesi nxenesi2 = new Nxenesi(15, 3);
```



5.4.2. Fjala kyçe *this*

Fjala kyçe THIS

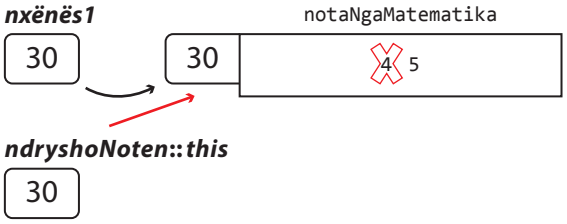
Nganjëherë është e nevojshme që metoda të tregojë në objektin që e ka thirrur. Java e mundëson një gjë të tillë nëpërmjet fjalës kyçe ***this***. Fjala kyçe `this` mund të përdoret brenda cilësdo metodë që të tregohet në objektin aktual. Kjo domethënë se `this` është gjithmonë referencë në objektin për të cilin është thirrur metoda.

Të shikojmë, klasën *Nxenesi*, pak të ndryshuar:

```
class Nxenesi {
   int    notaNgaMatematika = 4;
   public void ndryshoNoten(int ndryshimi) {
      notaNgaMatematika  = notaNgaMatematika  + ndryshimi;
      /* kemi mundur të shkruajmë
         this.notaNgaMatematika = this.notaNgaMatematika + ndryshimi;
         që nënkupton se this tregon në objektin aktual i cili
         ka thirrur metodën ndryshoNoten  */
   }
   public static void main(String[] args) {
      Nxenesi nxenesi1 = new Nxenesi();
      Nxenesi nxenesi2 = new Nxenesi();
      nxenesi1.ndryshoNoten(1); // linja 1
      nxenesi2.ndryshoNoten(-1); // linja 2
   }
}
```

Metoda *ndryshoNoten* si argument hyrës pranon vlerën për të cilën nxënësi e ka zmadhuar/zvogëluar notën nga matematika dhe vlerën e notës së re e vendos për vlerë të atributit *notaNgaMatematika*. Pamja e memories operative është paraqitur në figurë.

Pamja e memories operative – Vija 1



Pamja e memories operative – Vija 2

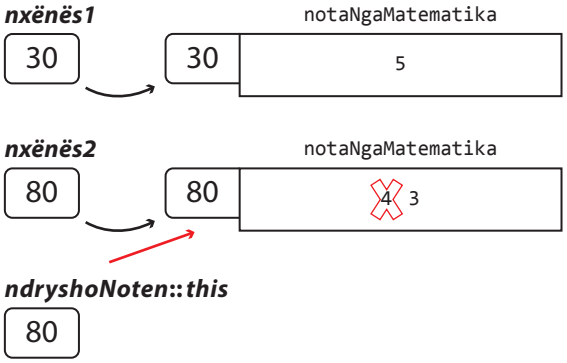


Figura 5.3. Fjala kyçe `this`

Tani shikojmë konstruktorët e mëposhtëm të klasës *Nxenesi*, pak të ndryshuar:

```
// jala kyçe this nuk është e nevojshme të përdoret

Nxenesi(String emriNxenesit, String mbiemriNxenesit, int nrEv) {
   this.emri  = emriNxenesit;
   this.mbiemri  = mbiemriNxenesit;
   this.numriEvidences = nrEv;
}

// Fjala kyçe this është e domosdoshme të përdoret

Ucenik(String emri, String mbiemri, int numriEvidentues) {
   this.emri = emri;
   this.mbiemri = mbiemri;
   this.numriEvidences = numriEvidentues;
}
```

Shumë programues pikërisht me qëllim i japin emrat e njëjtë parametrave dhe atributeve të instancës, duke sqaruar se në atë mënyrë zmadhohet qartësia e programit, mirëpo gjithmonë janë të kujdesshëm dhe përdorin operatorin `this`.

Në konstruktorin e parë, përdorimi i fjalës kyçe this nuk është i nevojshëm, ajo vetëm në mënyrë eksplicite tregon se atributet me të cilat përkufizohen vlerat i takojnë objektit aktual (d.m.th. instancës).

Në anën tjetër, e dimë se në Javë nuk është i lejueshëm deklarimi i dy ndryshoreve lokale me të njëjtin emër brenda mjedisit të njëjtë. Mirëpo, emrat e ndryshoreve lokale dhe argumenteve formale të metodave mund të përputhen me emrat e atributeve të klasave. Në atë rast, ndryshoret lokale dhe parametrat e metodës e fshehin atributin me të njëjtin emër të instancës.

Prandaj në konstruktorin e dytë, ku janë përdorur emrat e njëjtë të parametrave të konstruktorëve si dhe të atributeve të klasës *Nxenesi*, është e domosdoshme përdorimi i operatorit `this` që të bëhet qasja e atributeve përkatëse. P.sh.

```
this.emri = emri;
```

ku *this.emri* paraqet vlerën e atributit të objektit aktual, kurse *emri* është vlera e parametrit hyrës të konstruktorit.

Shembulli 4. Klasës *Nxenesi* t'i shtohet një konstruktor me katër argumente hyrëse që paraqesin sipas radhës vlerat për: numrin e evidencës, emrin, mbiemrin dhe notën nga matematika. Ky konstruktor duhet t'u shoqërojë vlerat atributeve përkatëse, kurse atributeve të tjera t'u shoqërojë vlerat "e panjohur".

Të përkufizohet klasa *TestNxenesi* që do të krijojë objektin *nxenesi* dhe menjëherë nëpërmjet konstruktorit do t'i shoqërojë vlerat emrin Mark, mbiemrin Markaj, notën nga matematika 5, numrin e evidencës 25.

```
class Nxenesi  {
   int numriEvidences;
   String emri;
   String mbiemri;
   String shkolla;
   String paralelja;
   int notaNgaMatematika;

   public Nxenesi(int numriEvidences, int notaNgaMatematika) {

      /* konstruktori i krijuar në shembullin 3 është ndryshuar me
         qëllim që të përdor operatorin this */
      this.numriEvidences = numriEvidences ;
      this.emri = "e panjohur";
      this.mbiemri = "e panjohur";
      this.shkolla = "Gjimnazi";
      this.paralelja = "IV-1";
      this.notaNgaMatematika = notaNgaMatematika;
   }

   public Nxenesi(int numriEvidences, String emri, String mbiemr,
         int notaNgaMatematika) { // konstruktori i ri
      this.numriEvidences = numriEvidences;
      this.emri = emri;
      this.mbiemri = mbiemr;
      this.shkolla = "e panjohur";
      this.paralelja = "e panjohur";
      this.notaNgaMatematika = notaNgaMatematika;
   }
}

class TestNxenesi  {
   public static void main(String[] args) {
      Nxenesi nxenesi = new Nxenesi(25, "Mark", "Markaj", 5);
   }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/KontrolliQasjes_kontruktor/Nxenesi* në CD.

Siç është sqaruar më herët, për shkak të emrit të njëjtë të parametrave hyrës në konstruktorin e dytë, si emra të atributeve përkatëse, është i domosdoshëm përdorimi i fjalës së rezervuar `this`. Për shoqërimin e vlerës atributeve *klasa* dhe *lendaPreferuar*, përdorimi i fjalës `this` nuk është i domosdoshëm.

Klasa *Nxenesi* tani përmban dy konstruktorë, njërin pa vlera hyrëse dhe të dytin me dy vlera hyrëse. Siç e shihni, në Javë është e mundshme të krijohen më shumë konstruktorë të së njëjtës klasë, mirëpo duhet të përmbajnë numër të ndryshëm parametrash, ose numër të njëjtë parametrash, por të tipave të ndryshme. Kjo mundësi është e njohur me emrin ***mbivendosja e konstruktorëve***. Gjatë krijimit të objekteve, sipas parametrit hyrës përcaktohet se cili konstruktor thirret për inicializimin e tyre.

```
// thirrja e konstruktorit të parë
Nxenesi nxenesi1 = new Nxenesi(13, 5);

// thirrja e konstruktorit të dytë
Nxenesi nxenesi2 = new Nxenesi(15, "Ana", "Malaj", 3);
```

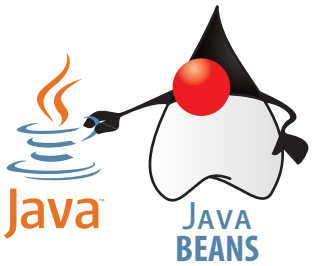
Nocion më i gjerë se mbivendosja e konstruktorëve është ***mbivendosja e metodave*** (anglisht *method overlapping*). Në Javë është e mundshme të krijohen dy ose më shumë metoda që kanë emra plotësisht të barabartë dhe njëkohësisht ndodhen në të njëjtën klasë. Mirëpo, duhet të kenë numër të ndryshëm parametrash ose numër të barabartë parametrash por të tipave të ndryshme. **Shembuj të këtyre metodave mund të gjeni në Përmbledhje.**

5.4.3. Metodat *Get* dhe *set*

Metodat get dhe set

Siç është sqaruar paraprakisht, disa atributeve të klasës u shoqërohet e drejta e qasjes `private` me qëllim që të "fshehen" dhe të mbrohen nga qasja prej klasave të tjera. Për ato atribute të klasës, të cilave megjithatë dëshirojmë t'u mundësojmë leximin ose ndryshimin e vlerave jashtë klasës konkrete, përdoren metodat *get* dhe *set* ose, më popullore *geterët* dhe *seterët*.

> ***Metoda get*** përdoret për leximin e vlerave të atributit të dhënë (kthen vlerën e atributit), kurse ***metoda set*** ka për parametër vlerën që e vendos si vlerë të re të atributit.



Gjatë krijimit të metodës, e cila do të kthejë vlerën e atributit të dhënë, shumë programues do të përdornin emërtime të ndryshme, të tipit "*merre...*", "*ktheje...*". Mirëpo që të zgjidhet ky problem dhe të realizohet uniteti në emërtimin e metodave të cilat kthejnë vlerën e një atributi, *JavaBeans specifikimi* përkufizon rregullën e krijimit të emrave në formën ***getEmriAtributit***. Përjashtimi i vetëm janë *get* metodat për atributet *Boolean*, emërtimi i të cilave është ***isEmriAtributit***. Në mënyrë të ngjashme, *set* metodat kanë emërtimin ***setEmriAtributit***, që vlen për atributet e çdo tipi.

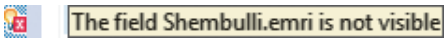
Shembulli 5. Në pjesën e mëposhtme të kodit të sqarohet dallimi midis mënyrave //1 dhe //2 për qasjen e atributit emri.

```
public class Shembulli {
   private String emri;

   public String getEmrin() {
      return emri;
   }

   public void setEmri(String emri) {
      this.emri = emri;
   }
}
```

```
public class TestShembulli {
   public static void main(String[] args) {
      Shembulli objekti = new Shembulli();
      objekti.setEmri("Hana");      // 1
      objekti.emri = "Mirani";      // 2
   }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/KontrolliQasjes_SetGetMetoda* në CD-në.

Në klasën *TestShembulli* bëhet përpjekja t'i qaset vlerës së atributit *emri* të instancës *objekti* të klasës *Shembulli*. Meqenëse niveli i qasjes së atributit *emri* është private, rrjedh se vlerën e re të këtij atributi mund ta vendosim vetëm duke thirrur *set* metodën, *setEmri*. Domethënë me komandën //1 për vlerë të atributit *emri* vendoset Maja, kurse komanda //2 paraqet gabim për shkak të drejtës së qasjes:



Shembulli 6. Të formohet klasa publike *Tëpunesuarit* që ka:

- Atributin privat *emriMbiemri*.
- Atributin privat *rroga*, që paraqet rrogën e të punësuarit.
- Atributin privat *drejtori*, që mund të ketë vlerën `true`, nëse i punësuari është drejtor, në të kundërtën është `false`.
- Metodat përkatëse *set* dhe *get* për të gjitha atributet.

```
public class Punesuarit {
   String  emriMbiemri;
   double  rroga;
   Boolean drejtori;

   public String GetEmriMbiemri() {
      return emriMbiemri;
   }

   public void setEmriMbiemri(String emriMbiemri) {
      this.emriMbiemri = emriMbiemri;
   }

   public double getRroga() {
      return rroga;
   }

   public void setRroga(double rroga) {
      this.rroga = rroga;
   }

   public Boolean isDrejtori() {
      return drejtori;
   }

   public void setDrejtori(Boolean drejtori) {
      this.drejtori = drejtori;
   }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/KontrolliQasjes_SetGetMetode/Punesuarit* në CD-në.

Pyetje dhe detyra kontrolli

1. Kur thirren konstruktorët?
2. Cili është dallim kryesor ndërmjet konstruktorit të nënkuptuar dhe atij parametrik?
3. Të sqarohet domethënia e fjalës kyçe `this`.
4. Çfarë domethënë mbivendosja e metodave?
5. A mund të përputhen konstruktorët?

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Klasës *Lojtari* shtoji:

- Konstruktorin që si argument pranon vetëm emrin dhe mbiemrin e lojtarit dhe e vendos për vlerë të atributit *emriMbiemri*, kurse për atributet *ekipiFutbolit* dhe *nrGolave* vendos sipas radhës vlerat "e panjohur" dhe 0.
- Konstruktorin që për argument pranon vlerat për të gjitha atributet dhe i vendos për vlerat e atributeve përkatëse.
- Metodat publike *set* dhe *get* për secilin prej atributeve.

Projekti *ManariProjekti*. Klasës *Manari* shtoji:

- Konstruktorin që për argument pranon llojin dhe racën e manarit dhe i vendos për vlerë të atributeve *lloji* dhe *raca*, pastaj për atributin *mosha* vendos vlerën 0.
- Konstruktorin që si argument i pranon vlerat për të gjitha atributet dhe i vendos për vlerat e atributeve përkatëse.
- Metodat publike *set* dhe *get* për secilin prej atributeve.

Projekti *QytetetProjekti*. Klasës *Qyteti* shtoji:

- Konstruktorin që për argument pranon emrin e qytetit dhe numrin postar dhe i vendos për vlerë të atributeve *emri* dhe *nrPostar*, pastaj për atributin *nrBanorve* dhe *kampusiUniversitar* vendos vlerat 0 dhe `false`, sipas radhës.
- Konstruktorin që si argument i pranon vlerat për të gjitha atributet dhe i vendos për vlerat e atributeve përkatëse.
- Metodat publike *set* dhe *get* për secilin prej atributeve.

5.5. Metoda *main*

Në shumë shembuj deri tani kemi përdorur metodën ***main***, që tani do ta sqarojmë më hollësisht. Në fillim kemi cekur, se para se të fillojë përkthimi i programit të Javës, ai duhet të ruhet në skedarin që ka emrin e njëjtë si dhe klasa e programit të Javës që përmban metodën `static` *main*() që niset e para gjatë ekzekutimit të programit.

Fjala e rezervuar static

Së pari të sqarojmë çfarë domethënë ka fjala e rezervuar ***static***.

Kur paraqitet nevoja që njëri prej elementeve të klasës të mund t'i shoqërohet, si për objekte të klasës të cilës i takon, po ashtu edhe për objekte të klasave të tjera, para tij vendoset fjala kyçe `static`. Kështu `static` elementi në qoftë se është:

- ***atribut***, sillet si ndryshore globale,
- ***metodë***, sillet si metodë globale.

Çfarë do të ndodhë nëse përpara *main*() lihet jashtë `static` dhe/ose `public`?

Përgjigjja: Programi do të kompilohet, mirëpo te nisja e tij do të paraqitet mesazhi: *Exception in thread 'main' java.lang. NoSuchMethodError:main*

Sikur metoda *main*() nuk do të ishte `static` metodë, thirrja e programit nuk do të mund të ekzekutohet sepse *jo-static* metodat mund të thirren vetëm mbas krijimit të objektit, i cili do t'i thirrasë.

Shembulli 7. Të krijohet klasa *testNxenesi* e cila ka vetëm metodën *main*. Në metodën main të krijohet një objekt i klasës *Nxenesi* emri i të cilit është Mark, mbiemri Markaj, numri i evidencës (në ditar) 25 dhe ka notën 5 nga matematika. Gjithashtu, të paraqiten të dhënat mbi nxënësin; emri dhe mbiemri në një rresht, kurse nota nga matematika me mesazhin përkatës në rreshtin e dytë.

```
public class TestNxenesi {

   public static void main(String[] args) {

      Nxenesi nxenesi1 = new Nxenesi(25, "Mark", "Markaj", 5);
      /* krijimi i objektit nëpërmjet konstruktorit të
         krijuar në seksionin 5.4.1 5.4.1  */

      System.out.println("Nxenesi: " + nxenesi1.emri + " "
                                     + nxenesi1.mbiemri);

      System.out.println("Nota nga matematika: "
                         + nxenesi1.notaNgaMatematika);
   }
}
```

Metodave brenda objektit i qasemi gjithashtu me ndihmën e operatorit (`.`); në qoftë se metoda ka vlerën kthyese, ajo fillimisht duhet të vendoset në ndryshoren ndihmëse, dhe më tutje mund të përdoret. Ndryshorja ndihmëse së pari përkufizohet ashtu që të ketë tip të njëjtë si vlera kthyese e metodës.

Projekti i tërë ndodhet në dosjen *Teksti/src/metodaMain* në CD-në.

Në fund, kur programi niset, në daljen standarde do të paraqitet:

```
Nxenesi: Mark Markaj
Nota nga matematika: 5
```

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Krijo klasën *TestLojtari* që përmban metodën *main* për testim të metodave të krijuara.

Projekti *ManariProjekti*. Krijo klasën *TestManari* që përmban metodën *main* për testim të metodave të krijuara

Projekti *QytetetProjekti*. Krijo klasën *TestQyteti* që përmban metodën *main* për testim të metodave të krijuara.

5.6. Bazat e trashëgimit

Më herët, në kapitullin IV kemi sqaruar konceptin e trashëgimit si një nga konceptet më të rëndësishme të OO. Tani do sqarojmë se si implementohet (realizohet) hierarkia e klasave.

Trashëgimi i klasës

Sintaksa themelore për deklarimin e nënklasës që trashëgon mbiklasën është:

```
class Nenklasa extends Mbiklasa {
   // trupi i klasës
}
```

Përparësia e trashëgimit nuk përfshihet vetëm me faktin se koha gjatë shënimit të kodit zvogëlohet, sepse përdoret pjesa e shënuar e kodit në fillim më shumë herë, por edhe në faktin se në atë mënyrë shkurtohet koha e testimit!

Principi i trashëgimit më së thjeshti mund të paraqitet në shembull, prandaj le t'i kthehemi shembullit të klasës sonë *Nxenesi* dhe klasës *NxënosiIT* që e trashëgon atë. Në kapitullin IV kemi përmendur se:

- klasa *Nxenesi* përmban atributet *numriEvidences*, *emri*, *mbiemri*, *emriShkolles*, *paralelja*, *notaNgaMatematika*, si dhe metodat *perfaqesimiProfesionit* me të cilën paraqitet mesazhi: "Mbas përfundimit të shkollës së mesme kemi kualifikimin e mesëm profesional të njohur nga Enti për punësim i Malit të Zi."
- klasa *NxenesiIT*, krahas atributeve të trashëguara, ka edhe atributin *notaProgramimi*, kurse metoda *perfaqesimiProfesionit* paraqet mesazhin: "Në shkollën e mesme jam takuar me shumë fusha të zbatimit të ICT-se, kurse nota ime nga programimi është " *notaProgramimi*.

Tani do të krijojmë të dy klasat, si dhe klasën *TestNxenesit* që përmban metodën main në kuadër të së cilës do të krijojmë objektet e klasës *Nxenesi* dhe *NxenesiIT* dhe tregojmë punën e metodës *perfaqesimiProfesionit*. Për shkak të hapësirës së kufizuar për kod, klasës *Nxenesi* do t'i përkufizojmë vetëm atributet *emri*, *mbiemri*, *notaNgaMatematika*. Kodi i tërë do të jetë publik, prandaj nuk do të shënojmë metodat përkatëse *get* dhe *set*.

```
class Nxenesi { // krijimi i mbiklasës Nxenesi
   String emri;
   String mbiemri;
   int    notaNgaMatematika;

   void perfaqesimiProfesionit() {
      System.out.println("Mbas përfundimit të shkollës së mesme kemi " +
                         "kualifikimin e mesëm profesional të njohur " +
                         "nga Enti për punësim i Malit të Zi!");
   }
}

class NxenesiIT  extends Nxenesi {
   // krijimi i klasës NxenesiIT që trashëgon klasën Nxenesi
   int notaProgramimi;

   void perfaqesimiProfesionit() {
      System.out.println("Gjatë shkollës së mesme jam njoftuar me "
                      + "fushat e ndryshme të zbatimit të ICT-së, kurse "
                      + "nota ime nga programimi është: "
                      + this.notaProgramimi);
   }
}

class TestNxenesit {

   public static void main(String[] args) {
      Nxenesi nxenesi1 = new Nxenesi();
      NxenesiIT nxenesi2 = new NxenesiIT();

      nxenesi1.emri  = "Mark"; // mbiklasa mund të përdoret në mënyrë të pavarur
      nxenesi1.mbiemri  = "Markaj";
      nxenesi1.perfaqesimiProfesionit();

      nxenesi2.emri  = "Ana";
      nxenesi2.mbiemri = "Gjokaj";
      /* nënklasa ka qasje të gjithë elementeve të trashëguar nga mbiklasa */
      nxenesi2.notaProgramimi = 5;
      nxenesi2.perfaqesimiProfesionit();
   }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/BazatTrashegimit* në CD-në.

Në fund, kur programi niset, në daljen standarde do të paraqitet:

```
Mbas përfundimit të shkollës së mesme kemi kualifikimin e mesëm
profesional të njohur nga Enti për punësim i Malit të Zi!
Në shkollën e mesme jam takuar me shumë fusha të zbatimit të ICT-se, kurse
nota ime nga programimi është: 5
```

Në mënyrë analoge me mbivendosjen (mbulesën) e metodave, nënklasa mund të mbulojë edhe konstruktorin e nënklasës. Madje, kjo ndodh shpesh meqenëse mbiklasat kanë atribute të veçanta (të qenësishme) që duhet të inicializohen. Në qoftë se në metodën/konstruktorin vetëm dëshirohet të zgjerohet metoda/konstruktori origjinal nga mbiklasa, përdoret fjala kyçe super. Kjo nuk ka ndodhur në shembullin paraprak *NxenesiIT*, ku metoda *perfaqesimiProfesionit* ka pasur trup plotësisht të ndryshëm nga metoda në mbiklasën *Nxenesi*.

Që është fjala mbi mbivendosjen e metodave, Eclipse vetë e njeh dhe e shënon

```
void perfaqesimiProfesionit() {
```

Detyra 1. Klasave *Nxenesi* dhe *NxenesiIT*, që i kemi krijuar në shembullin paraprak:

- T'u shtohen konstruktorët për vendosjen e vlerave të atributeve përkatëse.
- Klasës *Nxenesi* t'i shtohet metoda *paraqitja* e cila i paraqet të gjitha vlerat e atributeve.
- Në klasën *NxenesiIT* të krijohet metoda *paraqitja* ashtu që metoda me të njëjtin emër nga mbiklasa zgjerohet duke paraqitur notën nga programimi.

```
class Nxenesi { // krijimi i mbiklasës Nxenesi
    String emri;
    String mbiemri;
    int    notaNgaMatematika;

    Nxenesi(String emri, String mbiemri, int notaNgaMatematika) {
       this.emri = emri;
       this.mbiemri = mbiemri ;
       this.notaNgaMatematika = notaNgaMatematika ;
    }

    void paraqitja() {
       System.out.println("Nxenesi: " + this.emri + " "
                          + this.mbiemri
                          + ", ka notën nga matematika:"
                          + this.notaNgaMatematika);
    }
}

class NxenesiIT extends Nxenesi {
    int notaProgramimi;

    NxenesiShkollaProfesionale(String emri, String mbiemri,
                       int notaNgaMatematika,
                       int notaProgramimi) {
       super(emri, mbiemri, notaNgaMatematika);
       this.notaProgramimi = notaProgramimi;
    }
```

Projekti i tërë ndodhet në dosjen *Teksti/src/BazatTrashëgimit* në CD

```
    void paraqitja() {
       super.paraqitja();
       System.out.println("Nota nga programimi është: "
             + this.notaProgramimi);
    }
}
```

Metoda *main* e klasës *TestNxenesi* bën thirrjen e metodave të krijuara në mënyrën e mëposhtme:

```
public class TestNxenesi {

  public static void main(String[] args) {

    Nxenesi nxenesi = new Nxenesi("Ana", "Malaj", 3);
    NxenesiIT nxenesiIT = new NxenesiIT("Mark", "Markaj", 5, 5);

    nxenesi.paraqitja();
    nxenesiIT.paraqitja();
  }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Nxenesi: Ana Malaj, ka notën nga matematika: 3
Nxenesi: Mark Markaj, ka notën nga matematika: 5
Nota nga programimi është: 5
```

Pyetje dhe detyra kontrolli

1. Çfarë është trashëgimi?
2. Kur përdoret fjala e rezervuar `super`?
3. Si thirren konstruktorët në hierarkinë e klasave?
4. Cilat janë përparësitë e konceptit të trashëgimit?

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Krijo klasën *Perfaqesues* e cila, krahas elementeve të klasës *Lojtari*, përmban

- Atributin *kombetarja*, që paraqet emrin e ekipit kombëtar në të cilin lojtari luan.
- Atributin *nrGolaveKombetarja*, që paraqet numrin e golave që lojtari i ka shënuar duke luajtur për ekipin kombëtar (kombëtare).
- Metodën *golatKombetarja*, që paraqet të dhënat mbi numrin e golave në kombëtare.
- Metodën *printimi*, që paraqet të dhënat mbi lojtarin në mënyrën e mëposhtme: në rreshtin e parë të dhënat mbi lojtarin, në rreshtin vijues të dhënat mbi numrin e golave të shënuar për kombëtaren dhe në rreshtin e tretë numrin e golave për ekipin në të cilën luan.

Krijo klasën *TestKombetarja*, që përmban vetëm metodën *main* në të cilën krijohen dy objekte: objekti i parë i klasës *Lojtari* dhe objekti i dytë i klasës *Perfaqesues*. Testo metodat përkatëse.

Projekti *ManariProjekti*. Krijo klasën *ManariPerGara* e cila, krahas elementeve të klasës *Manari*, përmban:

- Atributin *kategoriaGara*, që paraqet kategorinë e garave në të cilin manari merr pjesë.
- Atributin *cmimiPrestigjioz*, që paraqet çmimin më prestigjioz të cilin manari e ka arritur deri tani.
- Metodën *tDhenatGara*, që paraqet të dhënat mbi kategorinë dhe çmimin më prestigjioz që manari e ka arritur.
- Metodën *printimi*, që paraqet të dhënat mbi manarin në mënyrën e mëposhtme: në rreshtin e parë të dhënat mbi manarin, kurse në rreshtin vijues të dhënat mbi kategorinë dhe çmimin më prestigjioz që manari ka arritur.

Krijo klasën *TestManariGara*, që përmban vetëm metodën *main* në të cilën krijohen dy objekte: objekti i parë i klasës *Manari* dhe objekti i dytë i klasës *ManariPerGara*. Testo metodat përkatëse.

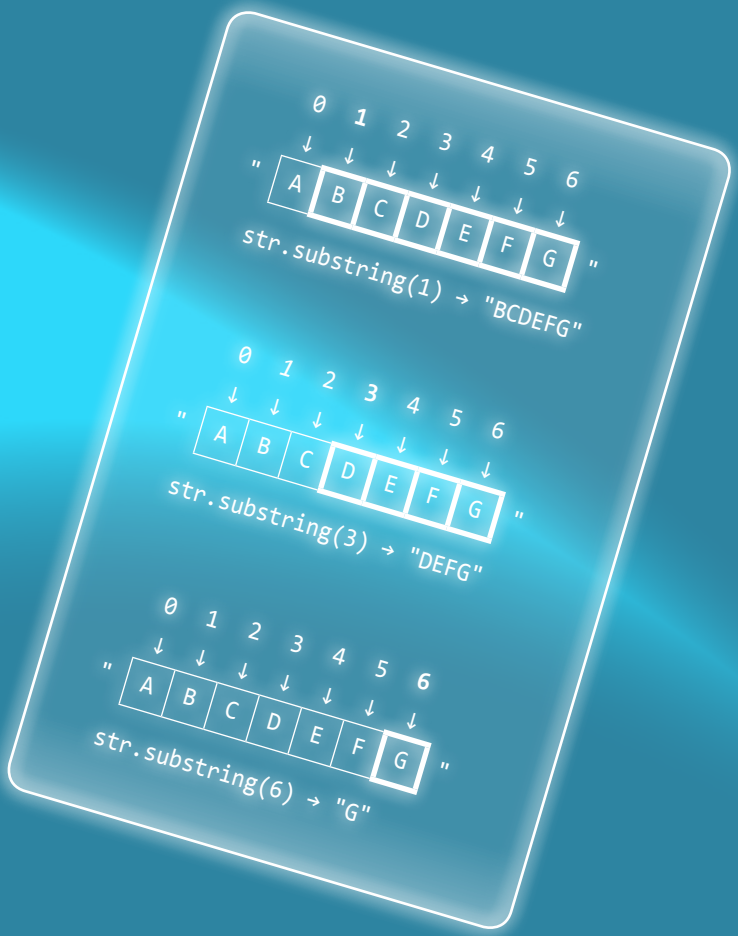
Projekti *QytetetProjekti*. Krijo klasën *KampusiUniversitar* e cila, krahas elementeve të klasës *Qyteti*, përmban:

- Atributin *numriStudentave*, që paraqet numrin e përgjithshëm të studentëve në qytet.
- Atributin *numriFakulteteve*, që paraqet numrin e përgjithshëm të fakulteteve që ekzistojnë në qytet.
- Metodën *tDhenatKampusi*, që paraqet të dhënat mbi numrin e përgjithshëm të studentëve dhe numrin e fakulteteve në qytet.
- Metodën *printimi*, që paraqet të dhënat mbi qytetin në mënyrën e mëposhtme: në rreshtin e parë të dhënat mbi qytetin kurse në rreshtin e dytë të dhënat numrin e studentëve dhe numrin e fakulteteve në atë qytet.

Krijo klasën *TestKampusiUnivesitar*, që përmban vetëm metodën main në të cilën krijohen dy objekte: objekti i parë i klasës *Qyteti* dhe objekti i dytë i klasës *KampusiUniversitar*. Testo metodat përkatëse.

VI.

JAVA BIBLIOTEKA E KLASAVE



Një nga aspektet karakteristike të zhvillimit të softuerit në mënyrën e programimit të orientuar në objekte është mundësia e përdorimit të bibliotekës së klasave. Biblioteka e klasave është bashkësia e klasave që përkrahin zhvillimin e programit.

Në Javë ekziston biblioteka standarde e klasave që mund të përdoret kur është nevoja. Klasat që ndodhen në këto biblioteka janë shumë të rëndësishme për programuesit, sepse përmbajnë funksione speciale. Programuesit shpeshherë bëhen të varur nga këto biblioteka dhe fillojnë të mendojnë mbi to si pjesë e gjuhës. Në aspektin teknik, këto biblioteka nuk janë pjesë e gjuhës, por paraqesin zgjerimin dinamik të funksionalitetit të vetë gjuhës.

Në këtë kapitull janë sqaruar klasat që paraqesin mbështjellat e tipave të thjeshtë të dhënave *int* dhe *double*:

- *Integer* (*java.lang. Integer*),
- *Double* (*java.lang. Double*).

dhe klasat që përdoren më shpesh në Javë:

- *String* (*java.lang. String*),
- *Math* (*java.lang. Math*),
- *Calendar* (*java.util.Calendar*).

Bibliotekat e klasave

Biblioteka e klasave përbëhet nga më shumë pjesë. Secila pjesë paraqet klasa të lidhura ndërmjet vete që së bashku përbëjnë ***Java API*** (anglisht *Application Programming Interface*). Për shembull, ekzistojnë Java API për qasje të bazave të të dhënave në të cilat ekzistojnë klasat që përdoren për punë me bazat. Ekziston edhe *Java Swing API*, në të cilën ndodhen klasat që paraqesin komponentë speciale grafike, që përdoren për krijimin e interfejsit të përdoruesit. Nganjëherë biblioteka komplete standarde e klasave quhet Java API.

Klasat në kuadër të bibliotekës së klasave standarde janë grumbulluar në paketa (*package*). Secila klasë është pjesë e një pakete. Klasa *String* është, për shembull, pjesë e paketës *java.long*. Kësaj pakete i takojnë edhe klasat *Integer* dhe *Double*. Klasat që ndodhen në paketën *java.lang* mund të përdoren në mënyrë automatike gjatë shënimit të programit. Këto klasa janë bazike dhe mund të konsiderohen si pjesë e gjuhës.

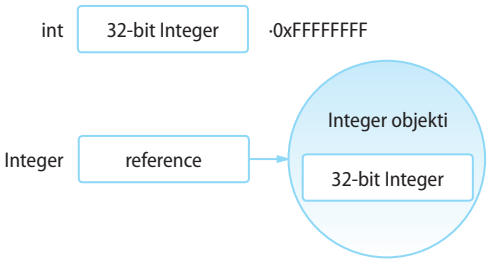
Të gjitha klasat në Javë janë të organizuara në paketa. Paketat në Javë janë të ngjashme me direktoriume (skedarë) që përdoren për organizimin e të dhënave në disk. Edhe klasat që i kemi përdorur deri tani ndodhen në paketa.

Krahas paketës *java.lang* në bibliotekën e klasave standarde të Javës ekzistojnë edhe paketa të tjera. Nëse dëshirohet të përdoret ndonjë nga këto paketa, ato duhet të kyçen paraprakisht në program nëpërmjet deklarimit `import`. Disa nga paketat më të rëndësishme në bibliotekën e klasave standarde të Javës janë paraqitur në tabelën 6.1.

Tabela 6.1. Disa nga paketat në Java bibliotekën standarde:

Paketa	Objektivi
`java.applet`	Përdoret për zhvillimin e apletëve
`java.awt`	Përdoret për punë me grafikë dhe interfejsin grafik të përdoruesit
`java.beans`	Përdoret për punë me komponentë të softuerëve
`java.io`	Përmban klasat që kryejnë funksione të ndryshme të lidhura me hyrje dhe dalje
`java.lang`	Përkrahje e përgjithshme. Kjo paketë është në mënyrë automatike e importuar në të gjithë programet
`java.math`	Përdoret për llogaritje matematike
`java.net`	Përdoret për komunikim nëpërmjet rrjetit
`java.rmi`	Përmban klasat për shënimin e programit që mund të shpërndahen në më shumë kompjuterë (RMI – shkurtesa për *Remote Method Invocation*)
`java.security`	Përmban klasat që korrespondojnë me sigurinë
`java.sql`	Përmban klasat për punë me baza të të dhënave
`java.text`	Përmban klasat për formatimin e tekstit
`java.util`	Përmban klasat shërbyese
`javax.swing`	Përmban klasat për krijim e interfejsit grafik të përdoruesit. Zgjerimi i klasës nga paketa AWT

6.1. Klasa *Integer*



Ndryshorja `int` paraqet numër të plotë, kurse *Integer* paraqet referencën në objektin që përmban numrin e plotë.

Për të gjitha tipat bazike në Javë, ekzistojnë klasat mbështjellat (anglisht *wrapper class*), që përmbajnë tipin bazik dhe metodat shtesë për manipulimin me atë tip bazik. Klasa ***Integer*** paraqet mbështjellësin e vlerave të tipit primitiv int. Përmban atributet e thjeshta të tipit `int` dhe një numër të madh të metodave të dobishme për manipulim me vlerat e tipit primitiv `int`. Klasa *Integer* nuk ka metoda që mundësojmë ndryshimin e vlerës e cila është dhënë njëherë. Prandaj ato janë ***objekte të pandryshueshme***.

Për shembull, gjatë krijimit të objektit të ri të klasës *Integer*, me vlerën 40, ajo vlerë më nuk mund të ndryshohet. Një gjë e tillë ndodh edhe me klasat e tjera që janë mbështjellëse të tipave primitivë.

Në tabelat 6.2., 6.3. dhe 6.4. janë paraqitur atributet, konstruktorët dhe metodat më të rëndësishme të klasës *Integer*.

Tabela 6.2. Atributet më të shpeshta të klasës *Integer* që përdoren

Ndryshorja	Deklarimi	Domethënia
`MAX_VALUE`	`public static int MAX_VALUE`	Konstantja që përmban vlerën maksimale që int mund të ketë $2^{31}-1$
`MIN_VALUE`	`public static int MIN_VALUE`	Konstantja që përmban vlerën minimale që int mund të ketë -2^{31}
`SIZE`	`public static int SIZE`	Numri i bitave që përdoret për paraqitje

Tabela 6.3. Najčešće korišćeni konstruktori klase *Integer*

Konstruktori	Domethënia
`Integer (int value)`	Krijon *Integer*-in e ri që ka vlerën e përkufizuar me vlerën value të tipit `int`.
`Integer (String s)`	Krijon objektin e ri *Integer* që ka vlerën e përkufizuar me stringun *s*.

Shembulli 1. Krijimi i objektit klasa *Integer* duke përdorur dy konstruktorë të ndryshëm.

```
public class ShembulliInteger1 {
    public static void main(String[] args) {

        // 1. Krijimi i objektit Integer nga vlera integer.
        Integer objekti1 = new Integer(10);

        /* 2. Krijimi i objektit Integer nga stringu. Vini re se ky
              konstruktor kërkon që parametri string të ketë interpretimin
              numerik, kurse në të kundërtën mund të shkaktojë gabimin në
              program. */

        Integer objekti2 = new Integer("10");

        //    paraqitja e vlerës së objektit Integer
        System.out.println("Në mënyrën e parë është krijuar: " + objekti1);
        System.out.println("Në mënyrën e dytë është krijuar: " + objekti2);
    }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Në mënyrën e parë është krijuar: 10
Në mënyrën e dytë është krijuar: 10
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/IntegerShembulli1* në CD

Tabela 6.4. Metodat e klasës *Integer* që përdoren më shpesh

Metodat	Deklarimet	Domethënia
doubleValue	`public double doubleValue()`	Kthen vlerën e numrit të plotë në tipin `double`.
compareTo	`public Boolean compareTo(Integer numri)`	Krahason objektin origjinal me objektin *numri* të tipit *Integer*.
floatValue	`public float floatValue()`	Kthen vlerën e numrit të plotë në tipit `float`.
getInteger	`public static Integer getInteger(String vlera)`	Kthen numrin e plotë në bazë të stringut *vlera*
getInteger	`public static Integer getInteger(String vrijednost, int numri)`	Kthen numrin e plotë, kurse në rastin kur vlera string nuk ekziston, do të kthejë vlerën e përmendur *numri*.
intValue	`public int intValue()`	Kthen vlerën e numrit të plotë si tip `int`.
longValue	`public long longValue()`	Kthen vlerën e numrit të plotë si tip `long`.
parseInt	`public static int parseInt(String vlera)`	Kthen vlerën int duke supozuar se *vlera* string e përmendur paraqet numër të plotë (zbërthen string në numër të plotë).
toString	`public String toString()`	Kthen *String*-un e ri që përmban vlerën e numrit të plotë.
toString	`public String toString(int numri)`	E përkthen numrin e plotë *numri* në *String* objekt.
valueOf	`public static Integer valueOf(String vlera)`	Duke supozuar se stringu i përmendur *vlera* paraqet numër të plotë, kthen objektin e ri *Integer* që përmban atë vlerë.

Shembulli 2. Shpeshherë në programe paraqitet nevoja e konvertimit të tipave të të dhënave. Të tregojmë në disa mënyra se si mund të kryhet konvertimi i ndryshoreve të tipit *String* në ndryshoret e tipit *Integer*.

```
public class StringToIntegerShembulli2 {

    public static void main(String[] args) {

    // 1. Krijimi i Integer duke përdorur konstruktorët.
    Integer objekti1 = new Integer("100");
    System.out.println("Në mënyrën e parë është krijuar: " + objekti1 );
```

```
    // 2. Krijimi me ndihmën e metodës valueOf të klasës Integer.
    String str = "100";
    Integer objekti2 = Integer.valueOf(str);
    System.out.println("Në mënyrën e dytë është krijuar: " + objekti2);

    // 3. Krijimi me ndihmën e metodës parseInt të klasës Integer.
    Integer objekti3 = Integer.parseInt(str);
    System.out.println("Në mënyrën e tretë është krijuar: " + objekti3);

    /* Vini re te tri metodat mund të shkaktojnë gabimin
       në qoftë se përdoret string që nga aspekti logjik
       nuk paraqet numër.  */
  }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/Integer/Shembulli2* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Në mënyrën e parë është krijuar: 100
Në mënyrën e dytë është krijuar: 100
Në mënyrën e tretë është krijuar: 100
```

Pyetje dhe detyra kontrolli

1. Cili është dallimi kryesor midis tipit primitiv int dhe klasës *Integer*?

2. Mbi çfarë duhet të kihet kujdes kur krijohet objekti i klasës *Integer* në bazë të një stringu?

3. Cilat nga komandat e mëposhtme janë korrekte:

   a) `x = new Integer(250);`

   b) `x = new Integer("Numri 150");`

   c) `x = new Integer("150");`

   d)
   ```
   x = new Integer("20");
   x = x + 5;
   int y = x + 5;
   ```

6.2. Klasa *Double*

Klasa ***Double*** paraqet mbështjellësin e vlerave të tipit primitiv dhe përmban atribute të thjeshta të tipit `double` dhe një numër të madh metodash të dobishme për trajtimin e vlerave të tipit primitiv `double`.

Double

Në tabelat 6.5., 6.6. dhe 6.7. janë paraqitur atributet më të rëndësishme, konstruktorët dhe metodat e klasës *Double*.

Tabela 6.5. Atributet që përdorën më së shpeshti në klasën *Double*

Ndryshoret	Deklarimi	Domethënia
`MAX_VALUE`	`public static double MAX_VALUE`	Konstanta që përmban vlerën maksimale që një vlerë `double` mund të ketë $(2-2^{-52})\cdot 2^{1023}$.
`MIN_VALUE`	`public static double MIN_VALUE`	Konstanta që përmban vlerën minimale që një vlerë `double` mund të ketë -2^{1074}.

Tabela 6.6. Najčešće korišćeni konstruktori klase *Double*

Konstruktori	Domethënia
`Double (double numri)`	Krijon objektin e ri të klasës *Double* që ka vlerën *numri*.
`Double (String vlera)`	Krijon objektin e ri të klasës *Double* që ka vlerën e përkufizuar me stringun *vlera*.

Tabela 6.7. Metodat e klasës *Double* që përdoren më shpesh

Metodat	Deklarimet	Domethënia
`doubleValue`	`public double doubleValue()`	Kthen numrin real në formën e `double`.
`compareTo`	`public boolean compareTo(Double numri)`	Krahason objektin origjinal me *Double* objektin *numri*.
`intValue`	`public int intValue()`	Përkthen vlerën e numrit real në numër të plotë të tipit `int`.
`longValue`	`public long longValue()`	Përkthen vlerën e numrit real në numër të plotë të tipit `long`.
`toString`	`public String toString()`	Kthen *String* objektin që përmban vlerën `double`.
`toString`	`public String toString(double numri)`	Përkthen numrin `double` *numri* në *String* objekt.
`valueOf`	`public static double valueOf(String vlera)`	Kthen numrin e tipit `double` i cili është paraqitur nëpërmjet stringut të përmendur *vlera*.
`parseDouble`	`public static double parseDouble(String vlera)`	Kthen vlerën `double` duke supozuar që stringu i përmendur *vlera* paraqet numër decimal.

Shembulli 1. Krijimi i objektit të tipit *Double* duke përdorur dy konstruktorë të ndryshëm.

```
public class ShembulliDouble1 {

   public static void main(String[] args) {
      /* 1. Krijimi i objektit të klasës Double
            nga tipi primitiv double.    */
      double d = 10.10;
      Double objekti1 = new Double(d);
      System.out.println("Në metodën e parë është krijuar: " +
objekti1);

      /* 2. Krijimi i objektit të klasës Double nga stringu. Vini re
            se ky konstruktor kërkon që parametri string të ketë
            interpretimin numerik, sepse në të kundërtën mund të
            shkaktojë gabim në program.  */
      Double objekti2 = new Double("25.34");
      System.out.println("Në metodën e dytë është krijuar: " +
objekti2);
   }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/ JavaBibliotekaKlasave/ Double/ Shembulli1* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Në metodën e parë është krijuar: 10.1
Në metodën e dytë është krijuar: 25.34
```

Shembulli 2. Në mënyrë të ngjashme si në shembullin 2 nga seksioni paraprak, të shkruhet programi që krijon objektit e tipit *Double* në bazë të parametrit hyrës në formën e *Stringut*.

```
public class ShembulliDouble2 {
    public static void main(String[] args) {
     /* 1. Krijimi i objektit të klasës Double duke përdorur
           konstruktorin. */
      Double objekti1 = new Double("100.564");
      System.out.println("Në metodën e parë është krijuar: " + objekti1);

      // 2. Duke përdorur metodën ndihmëse valueOf të klasës Double.
      String str1 = "100.476";
      Double objekti2 = Double.valueOf(str1);
      System.out.println("Në metodën e parë është krijuar: " + objekti2);

      /* Vini re se të dy metodat mund të shkaktojnë gabim në program
         në qoftë se paraqitet stringu që në aspektin logjik nuk
         duket si numër.  */
      String str2 = "76.39";
      double d = Double.parseDouble(str2);
      System.out.println("Në metodën e tretë është krijuar: " + d);
   }
}
```

Kur programi të niset, në daljen standarde do të paraqitet:

```
Në metodën e parë është krijuar: 100.564
Në metodën e dytë është krijuar: 100.476
Në metodën e tretë është krijuar: 76.39
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/Double/Shembulli2* në CD.

Pyetje dhe detyra kontrolli

1. Cili është dallimi kryesor midis tipit primitiv double dhe klasës *Double*?

2. Cilat nga komandat e mëposhtme janë korrekte:

a)
```
x = new Double(250.56);
```

b)
```
x = new Double("150,78");
```

c)
```
x = new Double(150);
```

d)
```
int x = 2;
y = new Double(2.3);
double z = x + y;
```

e)
```
double a = Double.parseDouble("150.78");
a++;
```

6.3. Klasa *String*

Stringjet në Java janë paraqitur nëpërmjet objekteve të klasës ***String*** dhe kanë veti dhe sjellje të përkufizuara në mënyrë shumë precize.

String

Klasa *String* përdoret në punë me stringje me gjatësi konstante, kurse klasa *StringBuffer* përdoret për punë me stringje të gjatësive të ndryshme. Klasa që pëdoret më shpesh nga Java biblioteka e klasave është me siguri klasa *String*, sepse çdo varg simbolesh (d.m.th. vargu i simboleve *Unicode*) në Javë më shpesh paraqitet si objekt i këtij tipi (p.sh. "ABCDEFG"). *Stringjet janë konstante, vlera e të cilave nuk mund të ndryshohet mbasi që krijohen*.

Për shembull:

```
System.out.println("ABCDEFG");
String cde = "EFG";
System.out.println("ABCD" + cde);
```

Java siguron përkrahjen speciale për operatorin e bashkimit (+). Në tabelën 6.8. janë dhënë disa shembuj.

Tabela 6.8. Shembuj të përdorimit të operatorit +

Komandat	Vlera e ndryshores rez mbas ekzekutimit të komandave
`x = "Marku";` `rez = x + " e mëson Javën";`	`"Marku e mëson Javën"`

`x = "Marku";`
`y = "Teuta";`
`rez = x + " dhe " + y;`
	`"Marku dhe Teuta"`

Klasa *String* përmban metodat për manipulimin me disa karaktera (shkronja) brenda stringut, metodat për krahasim dhe kërkimin e stringjeve, metodat për pjesëtimin (ndarjen) dhe krijimin e kopjeve të stringjeve, si dhe metodat për konvertimin e karaktereve të stringjeve në shkronja të vogla ose të mëdha. Konvertimi i objekteve të tjera (që programuesit mund t'i krijojnë edhe vetë) në *String* bëhet nëpërmjet metodës *toString*, të përkufizuar në klasën *Object*.

Shembulli 1. Në klasën *Nxenesi* nga kapitulli paraprak të shtohet metoda *toString* që kthen vlerat e të gjithë atributeve në trajtën e një stringu.

```
public class Nxenesi {
   int    numriEvidentues;
   String emri;
   String mbiemri;
   String shkolla;
   String paralelja;
   int    notaNgaMatematika;

   public String toString() {
      return "Nxenesi  [numriEvidentues=" + numriEvidentues +
             ", emri=" + emri + ", mbiemri=" + mbiemri +
             ", shkolla=" + shkolla + ", paralelja=" + paralelja +
             ", notaNgaMatematika=" + notaNgaMatematika + "]";
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/String/Shembulli1* në CD.

Ndryshores së tipit *String* mund t'i shoqërohet një varg karakteresh, si në shembullin e mëposhtëm:

```
String noviString = "Ky është stringu i ri";
```

Rezultati i njëjtë mund të përftohet në mënyrën e mëposhtme:

```
String noviString = new String("Ky është stringu i ri");
```

Në këtë mënyrë është krijuar objekti i ri i klasës *String*, kurse vlera është shoqëruar duke përdorur konstruktorët e klasës String. Konstruktorët e klasës që përdoren më shpesh janë rreshtuar në tabelën 6.9. Vini re se disa konstruktorë krijojnë objektin e klasës *String* mbi parametrin e tipit char ose *Integer*.

Tabela 6.9. Konstruktorët e klasës *String* që përdoren më shpesh

Konstruktori	Deklarimi	Domethënia
String	`public String()`	Krijon objektin e ri të klasës *String*, vlera e të cilit është stringu i zbrazët.
String	`public String(String vlera)`	Krijon objektin e ri të klasës *String*, vlera e të cilit inicializohet në vlerën e stringut *vlera*.
String	`public String(char[] vlera)`	Krijon objektin e ri të klasës *String*, vlera e të cilit është vargu i simboleve të përkufizuara me parametrin *vlera*.
String	`public String(StringBuffer baferi)`	Krijon objektin e ri të klasës String, vlera e të cilit është përcaktuar me parametrin *baferi*.

Disa nga konstruktorët (p.sh. konstruktori në rreshtin e dytë të tabelës paraprake) mundësojnë që gjatë krijimit të objektit në mënyrë automatike të shoqërohet vlera që i përgjigjet një stringu tjetër (ose pjesës së tij). Ekziston edhe mundësia që së pari të krijohet objekti i ri, dhe mbas t'i shoqërohet vlera.

```
String stringuIRi = new String();
stringuIRi = "Ky është stringu i ri";
```

Kopja e stringut mund të krijohet në mënyrën e mëposhtme:

```
String edheNjeString = new String(stringuIRi);
```

dhe kështu krijohet stringu i ri *edheNjeString* i klasës *String*, vlera e të cilit është e barabartë me vlerën e parametrit të *stringuIRi*.

Duke pasur parasysh se bëhet fjalë mbi objektet e tipit të njëjtë, vlera e një stringu mund t'i shoqërohet edhe në mënyrën e mëposhtme:

```
edheNjëString = stringuIRi;
```

Vlera e objektit *stringuIRi* të klasës *String* i shoqërohet objektit *edheNjeString* i klasës *String* (i cili paraprakisht duhet të jetë krijuar).

Shembulli 2. Janë dhënë fjalët "Dallëndyshet" dhe "pranverën". Duke përdorur operatorin + të formohet fjalia: "Dallëndyshet paralajmërojnë pranverën".

```
public class ShembulliString2 {

    public static void main(String[] args) {

        String str1 = "Dallëndyshet";  // ndryshorja str1 vlera e të cilës është "Dallëndyshet"
        String str2 = "pranverën";     // ndryshorja str2 vlera e të cilës është "pranverën"
        String str3 = str1 + " paralajmërojnë  " + str2;

        /* Shoqërimi i vlerës së ndryshores str3 duke bashkuar vlerat e  ndryshoreve str1,
           stringut "paralajmëron" dhe vlerës së ndryshores "pranverën */

        System.out.println("Fjala e parë: " + str1);
        System.out.println("Fjala e tretë: " + str2);
        System.out.println("Fjalia: " + str3);
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/String/Shembulli* në CD.

Kur programi të niset, në daljen standarde do të paraqitet:

```
Fjala e parë: Dallëndyshet
Flala e tretë: pranverën
Fjalia: Dallëndyshet paralajmërojnë pranverën
```

Në tabelën e mëposhtme janë rreshtuar metodat e klasës *String* që përdoren më shpesh, dhe fill mbas vijojnë shembujt për ilustrim.

Tabela 6.10. Metoda që përdoren më shpesh të klasës *String*

Metoda	Deklarimi	Domethënia
`length`	`public int length()`	Kthen numrin e karaktereve të stringut.
`charAt`	`public char charAt(int indeksi)`	Kthen shenjën që ndodhet në pozicionin *indeksi* të stringut.
`equals`	`public boolean equals(String stringuIDyt)`	Krahason vlerën string të objektit aktual me vlerën string të parametrit *stringulDyt*. Metoda kthen `true`, në qoftë se stringjet janë të barabarta (kanë gjatësi të barabartë dhe simbole të barabarta në pozicione përkatëse).
`equalsIgnoreCase`	`public boolean equalsIgnoreCase(String stringuIDyt)`	Krahason vlerën string me vlerën string të parametrit *stringulDyt*, duke mos përfillur shkronjat e vogla apo të mëdha. Shkronjat e mëdha zbërthehen në shkronja të vogla para krahasimit.
`compareTo`	`public int compareTo(String stringuIDyt)`	Krahason vlerën string me vlerën string të parametrit *stringulDyt*. Metoda kthen numrin e plotë i cili mund të jetë negativ, pozitiv ose zero, varësisht nga raporti midis tyre (më i madh, më i vogël, i barabartë).
`startsWith`	`public boolean startsWith(String stringuIDyt, int indeksi)`	Kontrollon nëse stringu fillon me prefiksin e përmendur apo jo nga pozicioni *indeksi*.
`startsWith`	`public boolean stratsWith(String stringuIDyt)`	Kontrollon, nëse stringu fillon me prefiksin e përmendur ose jo.
`endsWith`	`public boolean endsWith(String stringuIDyt)`	Kontrollon nëse stringu përfundon me prefiksin e përmendur ose jo.
`indexOf`	`public int indexOf(int karakteri)`	Kthen numrin që paraqet pozicionin brenda stringut në të cilën herën e parë paraqitet *karakteri* i përmendur.
`indexOf`	`public int indexOf(int karakteri, int indeksi)`	Kthen numrin që paraqet pozicionin brenda stringut në të cilën herën e parë paraqitet *karakteri* i përmendur, mbas pozicionit të përkufizuar me argumentin *indeksi*.
`lastIndexOf`	`public int lastIndexOf(int karakteri)`	Kthen numrin që paraqet pozicionin brenda stringut në të cilën herën e fundit paraqitet *karakteri* i përmendur

lastIndexOf	`public int lastIndexOf(int karakteri, int indeksi)`	Kthen numrin që paraqet pozicionin brenda stringut, në të cilën herën e fundit paraqitet karakteri, mirëpo mbas pozicionit *indeksi*.
substring	`public String substring(int indeksi1, int indeksi2)`	Kthen pjesën e stringut, duke filluar nga pozicioni *indeksi1deri* te pozicioni *indeksi2*.
replace	`public String replace(char karakteri1, char karakteri2)`	E zëvendëson karakterin *karakteri1* në string me karakterin e dytë *karakteri2*.
toLowerCase	`public String toLowerCase()`	Nga stringu i vjetër formon stringun e ri duke i shndërruar të gjitha shkronjat e stringut në shkronja të vogla.
toUpperCase	`public String toUpperCase()`	Nga stringu i vjetër formon stringun e ri duke i shndërrua të gjitha shkronjat e stringut në shkronja të mëdha.
trim	`public String trim()`	Nga stringu i vjetër formon stringun e ri duke duke i hequr hapësirat (space) eventuale në fillim dhe në fund të stringut.
toCharArray	`public char[] toCharArray()`	E përkthen stringun në një varg shenjash.

Objektivi i disa metodave është mjaft i qartë nga përshkrimi i tyre, meqenëse shumica e tyre mundëson qasjen e disa karaktereve të caktuara brenda stringut, ndryshimi i vlerave të pjesëve të caktuara të stringut, edhe operacione të tjera të ngjashme. Ato metoda kthehen si metoda të çdo klase (çfarë kemi sqaruar më herët në seksionin 5), d.m.th. me ndihmën e operatorit. (pika) shënohet objekti mbi të cilin metoda thirret.

Në vazhdim jepen disa shembuj të thjeshtë:

Metoda *length* kthen (jep) gjatësinë e stringut mbi të cilin vepron, kurse vlera e përftuar mund të paraqitet si pjesë përbërëse e mesazhit "Gjatësia e stringut është: ". Shikoni pjesën e kodit që bën implementimin përkatës.

```
System.out.println("Gjatësia e stringut është: " +
                   string1.length());
```

Metoda *charAt* si rezultat jep karakterin që ndodhet në pozicionin përkatës në string, ashtu që numërimi fillon nga 0; d.m.th. karakteri fillestar ndodhet në pozicionin 0, kurse karakteri i dytë në pozicionin 1 e kështu me radhë. Shikoni pjesën e kodit me të cilin paraqitet karakteri i tretë i stringut *string2*.

```
System.out.println("Karakteri i tretë i stringut " +
                   string2 + "është: " +
                   string2.charAt(2));
```

Në tabelën 6.11. janë dhënë disa shembuj të thjeshtë të përdorimit të metodave të klasës *String*.

Tabela 6.11. Shembujt e përdorimit të disa metodave të klasës *String*

Komanda	Vlera e ndryshores rez mbas ekzekutimit të komandës

`String string1 = "Programimi në Javë";`
`String string2 = "programimi në Javë";`
`Boolean rez = string1.equals(string2);`
	`false`

`String string1 = "Programimi në Javë";`
`String string2 = "programimi në Javë";`
`Boolean rez = string1.equalsIgnoreCase(string2);`
	`true`

`String string1 = "Programimi në Javë";`
`String string2 = "Java";`
`int rez = string1.compareTo(string2);`
	`6`
	Metoda *compareTo* krahason dy stringje. Në qoftë se janë të barabartë, metoda jep 0. Në të kundërtën, kthen numrin që paraqet ndryshesën në vlerat int të karaktereve të parë në string që janë të ndryshëm. (P.sh. P ka vlerën 80, kure J ka vlerën 74, prandaj ndryshesa e tyre është 6)

`String string="Marash";` `String rez = string.replace('a', 'i');`	`"Mirash"`
`String string1 = "Programimi në Javë";` `int rez=string1.length();`	`18`
`String string1 = "Programimi në Javë";` `int rez = string1.indexOf(' ');`	`10`
`String string1 = "Programimi në Javë";` `String rez = string1.substring(0, 10);`	`"Programimi"`
`String string1 = "   Programimi në Javë   ";` `String rez = string1.trim();`	`"Programimi në Java"`

Shembulli 3. Të verifikohen rezultatet e zbatimit të metodave *equals* dhe *equalsIgnoreCase* për kontrollimin e barazimit të stringjeve "JAVA" dhe "Java".

```
public class ShembulliString3 {
    public static void main(String[] args) {
        String stringu1 = "JAVA";
        String stringu2 = "Java";
        // metoda equals i krahason dy objektet të klasës String
        System.out.println("Rezultati i metodës equals: " +
                           stringu1.equals(stringu2));
        /* metoda equalsIgnoreCase i krahason dy objektet pavarësisht
           nga madhësia e shkronjave  */
        System.out.println("Rezultati i metodës equalsIgnoreCase: " +
                           stringu1.equalsIgnoreCase(stringu2));
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/String* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Rezultati i metodës equals: false
Rezultati i metodës equalsIgnoreCase: true
```

Shembulli 4. Për fjalinë e dhënë "*Dallëndyshet paralajmërojnë pranverën*" të gjendet gjatësia e përgjithshme dhe gjatësitë e fjalëve të saj.

```
public class ShembulliString4 {

    public static void main(String[] args) {
        String fjalia = "Dallëndyshet paralajmërojnë pranverën";
        int gjatesia;

        gjatesia = fjalia.length();
        System.out.println("Gjatësia e fjalisë është: " + gjatesia);

        int pozicioni;
        pozicioni = fjalia.indexOf(' ');
        String fjala;
        fjala = fjalia.substring(0, pozicioni);

        System.out.println("Fjala e parë është: " + fjala  +
                           ", kurse gjatesia e saj është: " + fjala.length());
    }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Gjatësia e fjalisë është: 37
Fjala e parë është: Dallëndyshet, kurse gjatesia e saj
është: 12
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/String* në CD-në

Shembulli 5. Në fjalën "David" shkronja D të zëvendësohet me H dhe shkronja v të zëvendësohet me m dhe të paraqitet fjala e përftuar në atë mënyrë. Fjala e re të paraqitet me shkronja të mëdha.

```
public class ShembulliString5 {

    public static void main(String[] args) {
        String fjala  = "David";
        fjala  = fjala .replace('D', 'H');
        fjala  = fjala .replace('v', 'm');

        System.out.println("Mbas ndryshimeve është përftuar fjala: " + fjala);
        System.out.println("PARAQITJA ME SHKRONJA TË MËDHA: " + fjala.toUpperCase());
    }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Mbas ndryshimeve është përftuar fjala: Hamid
PARAQITJA ME SHKRONJA TË MËDHA: HAMID
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/String* në CD.

Shembulli 6. Të paraqitet shkronja e parë dhe e fundit e fjalës "JAVA".

```
public class ShembulliString6 {

    public static void main(String[] args) {
        String fjala = "JAVA";
        String shkronjaEParë;
        String shkronjaEFundit;

        shkronjaEParë = fjala.substring(0, 1);
        // shkronja e parë ndodhet në pozicionin 0, deri te pozicioni 1

        shkronjaEFundit =fjala.substring(fjala.length()-1, fjala.length());
        /* shkronja e fundit ndodhet në pozicionin gjatesia-1,
           deri te pozicioni gjatesia  */

        System.out.println("Shkronja e parë është: " + shkronjaEParë);
        System.out.println("Shkronja e fundit është: " + shkronjaEFundit);
     }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/String* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Shkronja e parë është: J
Shkronja e fundit është: A
```

Pyetje dhe detyra kontrolli

1. Cili është dallimi midis klasave *String* dhe *StringBuffer*?

2. Plotëso tabelën me rezultatet e ekzekutimeve të blloqeve të komandave të përmendura, ashtu që ndryshorja stringu është përkufizuar me:

```
String stringu = "Mësoj stringjet në Javë";
```

Në qoftë se komanda do të shkaktojë gabim, të shënohet "gabim" dhe të sqarohet shkaku i shfaqjes së tij.

Komanda	Vlera e ndryshores rez mbas ekzekutimit të komandës
`int rez = string.length();`	
`char rez = string.charAt(50);`	
`String rez = string.substring(0, 5) + "Algoritmet dhe programimi";`	
`String stringu2 = "të programoj";` `String rez = string.replace("stringjet", stringu2);`	
`char rez = stringu.charAt(stringu.length());`	
`char rez = stringu.charAt(stringu.length()-1);`	
`Boolean rez = stringu.startsWith('P');`	
`Boolean rez = stringu.endsWith('I');`	

Puno vetë

1. Të shkruhet programi i cili, emrin dhe mbiemrin e nxënësit e shkruan me shkronja të mëdha.

2. Të shkruhet programi që nga fjalia e dhënë printon fjalën e dytë.

3. Të shkruhet programi që nga emri dhe mbiemri i nxënësit të krijojë adresën e e-mailit, që përbëhet nga shkronja e parë e emrit të nxënësit, pikës (.), mbiemrit të nxënësit dhe domenit "shkolla.edu".

4.* Të shkruhet programi që nga fjalia e dhënë printon fjalën e fundi.

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Në klasën *Lojtari* shto:

- Metodën *infoLojtari* që kthen të dhënën mbi ekipin e futbollit dhe pozicionin në të cilin lojtari luan në mënyrën vijuese: *EKIPI_FUTBOLLIT*; *pozicioni* (p.sh. për lojtarin që është portier në ekipin e futbollit "Buduqnost", metoda kthen: *BUDUQNOST*; *portier*).
- Metodën *shkurtesaEmrit* që paraqet shkurtesën e emrit të ekipit të futbollit të krijuar në mënyrën vijuese: *FC – tri shkronjat e para të emrit* (p.sh. për ekipin e futbollit "Buduqnost", shkurtesa është: *FC-BUD*).

Projekti *ManariProjekti*. Në klasën *Manari* shto:

- Metodën *infoManari* që paraqet të dhënat mbi manarin në mënyrën e mëposhtme: *Manari LLOJI*: *raca* (p.sh. për manarin te i cili lloji = "macja", raca = "macja persiane", metoda paraqet: *Manari MACJA*: *macja persiane*).
- Metodën *krijimiShenjesEvidences*, që kthen simbolin që mund të përdoret në kartonin e evidencës së manarit. Simboli përftohet duke lidhur dy shkronjat e para të emrit të llojit, linjës së poshtme (_), shkronjës së parë dhe të fundit të racës, linjës së poshtme (_) dhe moshës së manarit (p.sh. për manarin me këto karakteristika lloji = "macja", raca = "macja persiane", mosha = 5, simboli i krijuar është: *MA_ME_5*.

Projekti *QytetetProjekti*. Në klasën *Qyteti* shto:

- Metodën *krijimiIShenjes*, që kthen shenjën për qytetin që përftohet duke bashkuar dy shkronjat e para të emrit të qytetit me numrin postar (p.sh. për qytetin Podgorica, numri postar i të cilit është 81 000, shkurtesa (shenja) është *PO – 81 000*).
- Metodën *shkronjaNgaEmri*, që paraqet shkronjën e parë, shkronjën e mesme dhe shkronjën e fundit të emrit të qytetit (p.sh. për qytetin Cetinja, metoda kthen *C T A*, kurse për "Podgorica", metoda kthen *P O A*).

6.4. Klasa *Math*

Math

Klasa ***Math*** përmban metodat nëpërmjet të cilave janë paraqitur veprimet dhe funksionet themelore matematike që përdoren në gjeometri dhe trigonometri. Klasa *Math* i përkufizon dy konstante të tipit double: *Numrin e Ojlerit* (E) (përafërsisht i barabartë me 2,72) dhe *numrin e Ludolfit PI* (përafërsisht i barabartë me 3,14). Në tabelat 6.12. dhe 6.13. janë paraqitur disa nga metodat e klasës *Math* që u përgjigjen funksioneve trigonometrike që përdoren më shpesh, gjegjësisht funksioneve eksponenciale.

Tabela 6.12. Metodat e klasës *Math* që përdoren më së shpeshti që paraqesin funksionet trigonometrike

Metoda	Deklarimi	Domethënia
`toRadians`	`static double toRadians(double shkallët)`	Shkallët i kthen në radianë
`toDegrees`	`static double toDegrees(double radianet)`	Radianët i kthen në shkallë
`sin`	`static double sin(double argument)`	Kthen sinusin e këndit *argument* të dhënë në radianë
`cos`	`static double cos(double argument)`	Kthen kosinusin e këndit *argument* të dhënë në radianë
`tan`	`static double tan(double argument)`	Kthen tangjentin e këndit *argument* të dhënë në radianë
`asin`	`static double asin(double vlera)`	Kthen këndin në radianë, sinusi i të cilit është *vlera*
`acos`	`static double acos(double vlera)`	Kthen këndin në radianë, kosinusi i të cilit është *vlera*
`atan`	`static double atan(double vlera)`	Kthen këndin në radianë, tangjenti i të cilit është *vlera*

Tabela 6.13. Metodat e klasës *Math* që përdoren më shpesh dhe që paraqesin funksionet eksponenciale

Metoda	Deklarimi	Domethënia
`pow`	`static double pow(double y, double x)`	Kthen y^x.
`sqrt`	`static double sqrt(double arg)`	Kthen $\sqrt{arg}$.
`exp`	`static double exp(double arg)`	Kthen e^{arg}.
`cbrt`	`static double cbrt(double arg)`	Kthen $\sqrt[3]{arg}$.
`log`	`static double log(double arg)`	Kthen ln(*arg*).
`log10`	`static double log10(double arg)`	Kthen $\log_{10} arg$.

Klasa *Math* përmban shumë metoda për rrumbullakimin dhe krahasimin e numrave.

Tabela 6.14. Metodat e përdorura më shpesh të klasës *Math*

Metoda	Deklarimi	Domethënia
abs	`static int abs(int arg)`	Kthen vlerën absolute të numrit *arg*, |*arg*|.
abs	`static float abs(float arg)`	Kthen vlerën absolute të numrit *arg*, |*arg*|.
max	`static int max(int x, int y)`	I krahason numrat *x* dhe *y* dhe kthen numrin më të madh.
max	`static double max(double x, double y)`	I krahason numrat *x* dhe *y* dhe kthen numrin më të madh.
min	`static int min(int x, int y)`	I krahason numrat *x* dhe *y* dhe kthen numrin më të vogël.
min	`static double min(double x, double y)`	I krahason numrat *x* dhe *y* dhe kthen numrin më të vogël.
round	`static long round(double arg)`	Kthen numrin më të afërt të tipit long në raport me vlerën *arg*.

Në tabelën 6.15. janë dhënë disa shembuj të thjeshtë të zbatimit të metodave të klasës *Math*.

Tabela 6.15. Shembujt e zbatimit të disa nga metodat e klasës *Math*

Komanda	Vlera e ndryshores rez mbas ekzekutimit të komandave
`long rez = Math.round(3.45);`	3
`long rez = Math.round(3.75);`	4
`long rez = Math.abs(-4);`	4
`double x = 3/4;` `double rez = Math.pow(2,x);`	$2^{3/4} = \sqrt[4]{2^3}$
`Double rez= Math.max(4, 3.5);`	4.0
`Double r = 2.0;` `Double rez = Math.pow(r, 2)*PI;`	12.566370614359172

Shembulli 1. Llogaritja e rrënjëve të numrave të dhënë duke përdorur metodat *sqrt* të klasës *Math*.

```
public class ShembulliMath1 {

    public static void main(String[] args) {
        int i = 9;
        double d = 25.5;

        System.out.println("Rrënja katrore e numrit " + i +
                           " është: " + Math.sqrt(i));

        System.out.println("Rrenja katrore e numrit " + d +
                           " është: " + Math.sqrt(d));
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/Math* në CD.

Kur niset programi, në daljen standarde do të paraqitet:

```
Rrenja katrore e numrit 9 është: 3.0
Rrenja katrore e numrit  25.5  është: 5.049752469181039
```

Shembulli 2. Llogaritja e vlerës absolute të ndryshoreve të tipit int dhe double, duke përdorur metodat *abs* dhe *Math*.

```
public class ShembulliMath2 {

    public static void main(String[] args) {

        int i = 8;
        int j = -5;

        System.out.println("Vlera absolute e numrit " + i +
                           " është: " + Math.abs(i));

        System.out.println("Vlera absolute e numrit " + j +
                           " është: " + Math.abs(j));

        double d1 = 43.324;
        double d2 = -349.324;

        System.out.println("Vlera absolute e numrit " + d1 +
                           " është: " + Math.abs(d1));

        System.out.println("Vlera absolute e numrit " + d2 +
                           " është: " + Math.abs(d2));
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/Math* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Vlera absolute e numrit  8 është: 8
Vlera absolute e numrit -5 është: 5
Vlera absolute e numrit  43.324 është: 43.324
Vlera absolute e numrit -349.324 është: 349.324
```

Pyetje dhe detyra kontrolli

1. Plotëso tabelën me rezultatet e ekzekutimit të blloqeve të përmendura të komandave. Në qoftë se komanda do të shkaktojë gabimin, të shënohet "gabim" dhe të sqarohet shkaku i paraqitjes si tij.

Komandat	Vlera e ndryshores rez mbas ekzekutimit të komandave
`Rez = Math.ln(-3.45);`	
`Rez = Math.max(Math.max(3,6),2);`	
`Rez = Math.abs(Math.min(-2, 5));`	
`Rez = Math.sin(Math.toRadians(30));`	

Puno vetë

1. Të shkruhet programi për njehsimin e diametrit dhe perimetrit të rrethit në bazë të vlerës së njohur të sipërfaqes së tij.

2. Të shkruhet programi për njehsimin e perimetrit dhe sipërfaqes së trekëndëshit në bazë të vlerave të njohura të brinjëve të trekëndëshit.

3.* Të shkruhet programi me të cilin përcaktohet numri i plotë x që është zgjidhje e ekuacionit $x^a = n$. Në qoftë se nuk ekziston një numër i tillë i plotë, të përcaktohet numri i plotë i përafërt me rrënjën e ekuacionit.

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Në klasën *Lojtari* shto:

- Metodën *ndryshimiNumritGolave*, që si argument hyrës pranon dy objekte të klasës *Lojtari* që luajnë në pozicionet e sulmuesve. Metoda paraqet ndryshimin e numrit të golave të lojtarit që ka shënuar më shumë gola dhe lojtarit që ka shënuar më pak gola (duke marrë parasysh lojtarin e dhënë dhe dy lojtarë që paraqiten si parametra hyrës). P.sh. për lojtarin që ka shënuar 3 gola, në qoftë se metoda bën thirrjen e argumenteve hyrëse: *lojtari1* (që ka shënuar 6 gola) dhe *lojtari2* (që ka shënuar 4 gola), metoda kthen vlerën $6 - 3 = 3$.
- Metodën *numriPriturGolave*, që si argument hyrës pranon numrin e përgjithshëm të ndeshjeve në të cilat është planifikuar që lojtari të luajë. Metoda kthen numrin e golave që përftohet ashtu që mesatarja e golave të shënuar në sezonin paraprak shumëzohet me numrin e pritur të ndeshjeve që lojtari do të luajë në sezonin aktual (të ardhshëm). Numri i përftuar i golave të rrumbullakohet në numrin më të afërt të plotë (p.sh. për lojtarin që deri tani ka shënuar mesatarisht 0.75 gola për ndeshje, dhe do të luaj 5 ndeshje, numri i pritur është 4).

Projekti *ManariProjekti*. Në klasën *Manari* shto:

- Metodën *ndryshimiManaret*, e cila si argument hyrës e pranon objektin e klasës *Manari*. Metoda kthen vlerën absolute të ndryshimit në moshë ndërmjet manarëve (p.sh. për manarin në moshën 3 vjeç, metoda në rastin kur argumenti hyrës, që paraqet manarin me moshë 5 vjeç, do të kthejë vlerën 2).
- Metodën *cmimiManarit*, e cila si argument hyrës pranon çmimin aktual të manarit dhe vlerën *p* që paraqet përqindjet, dhe kthen numrin e plotë që paraqet vlerën e çmimit të manarit mbas

rritjes p % (p.sh. për manarin me çmim aktual 150 €, mbas rritjes prej 9%, çmimi i rrumbullakuar në numra të plotë është 163 €).

Projekti *QytetetProjekti*. Në klasën *Qyteti* shto:

- Metodën *ndryshimiQytetet*, e cila si argument hyrës e pranon objektin e klasës *Qyteti*. Metoda kthen vlerën absolute të ndryshimit të numrit të banorëve midis qyteteve (p.sh. për qytetin me 360 000 banorë, në rast se argumenti hyrës i metodës paraqet qytetin me 140 000 banorë, rezultati i metodës do të jetë 220 000).
- Metodën *nrBanorve*, e cila si argument hyrës pranon vlerën p që paraqet përqindjet, dhe kthen numrin e banorëve që pritet në rastin e shtimit të popullsisë për p % (p.sh. në qytetin me 1 000 999 banorë mbas shtimit prej 7%, numri i banorëve do të rritet në 1 071 069).

6.5. Klasa *Calendar*

Calendar

Klasa ***Calendar*** e cila i siguron metodat për punën me njësitë kohore: vitet, muajt, javët, ditët, orët, minutat, sekondat dhe pjesët e sekondave (milisekondat). Ndodhet të paketën *java.util*, dhe që të përdoret, klasa *java.util.calendar* duhet të importohet në mënyrë eksplicite

```
import java.util.Calendar;
```

Koha shënohet në formatin e mëposhtëm:

January 1, 1970 18:53:24.123 GMT (Gregorian).

Klasa Calendar ka metodën *getInstance*, për gjenerimin e objekteve të tipit *Calendar*, vlerat e ndryshoreve të të cilit inicializohen me vlerat aktuale (momentale) të vitit, muajit, ditës, orës, minutës dhe sekondës

```
Calendar dataESotme = Calendar.getInstance();
```

Objektit të klasës *Calendar* mund t'i shoqërohen vlerat për nocione kohore në pajtim me emërtimet nacionale. Kështu ditët në javë mund të quhen: e hënë, e martë, e mërkurë... Muajt e vitit mund të quhen: janar, shkurt, mars... Kështu mund të krijohen kalendarët kombëtarë (shiko shembullin, në të cilin përdoren vargjet me të cilët do të takohemi në njërin prej kapitujve të mposhtëm).

Më shumë informata mbi klasën *Calendar* mund të gjeni në adresën: http://dosc.oracle.com/javase/1.5.0/docs/api/java/calendar.htlm

```
String months[] = {
  "Janar", "Shkurt", "Mars", "Prill",
  "Maj", "Qershor", "Korrik", "Gusht",
  "Shtator", "Tetor", "Nëntor",
  "Dhjetor"};

Calendar calendar =
    Calendar.getInstance();
System.out.print("Muaji aktual: ");
System.out.print(months[calendar
    .get(Calendar.MONTH)]);
```

Tabela 6.16. Disa nga atributet e deklaruara nëpërmjet klasës *Calendar*

Tipi	Atributi	Përshkrimi
`static int`	**`MILISECOND`**	Vlera që paraqet një të mijtën pjesë të sekondës
`static int`	**`SECOND`**	Vlera që paraqet sekondën
`static int`	**`MINUTE`**	Vlera që paraqet minutën
`static int`	**`HOUR`**	Vlera që paraqet orën

`static int`	**`DAY_OF_WEEK`**	Vlera që paraqet ditën në javë.
`static int`	**`DAY_OF_MONTH`**	Vlera që paraqet ditën në muaj.
`static int`	**`DAY_OF_YEAR`**	Vlera që paraqet ditën në vit.
`static int`	**`WEEK`**	Vlera që paraqet javën.
`static int`	**`MONTH`**	Vlera që paraqet muajin. (0 – janari, 1 – shkurti, etj)
`static int`	**`YEAR`**	Vlera që paraqet vitin.
`static int`	**`MONDAY`**	Vlera për ditën që paraqet ditën e parë në javë.
`static int`	**`TUESDAY`**	Vlera për ditën që paraqet ditën e dytë në javë.
`static int`	**`JANUARY`**	Vlerën për muajin që paraqet muajin e parë në vit.
`static int`	**`FEBRUARY`**	Vlerën për muajin që paraqet muajin e dytë në vit
`static int`	**`time`**	Koha momentale në formatin January 1, 1970 18:53:24.123 GMT (Gregorian).

Tabela 6.17. Metodat që përdoren më shpesh të klasës *Calendar*

Metoda	Deklarimi	Përshkrimi
`getDate`	`int getDate()`	Kthen vlerën momentale të datës dhe kohës
`add`	`void add(int argumenti, int vlera)`	I shton vlerës *vlera* argumentit të dhënë *argumenti* në pajtim me rregullat e kalendarit.
`get`	`int get(int argumenti)`	E kthen vlerën momentale të atributit kalendarik *argument* (vitin, muajin, ditën, orën, minutën, sekondën etj.).
`getInstance`	`Calendar getInstance()`	Kthen objektin e klasës *Calendar* me datën momentale.
`getTime`	`Data getTime()`	E kthen objektin e tipit *Date* në bazë të vlerave të ndryshoreve (viti, muaji, ...)
`setTime`	`void setTime(Date data)`	Vendos vlerën për datë.
`set`	`void set (int viti, int muaji, int dita)`	Vendos vlerën për vitin, muajin dhe ditën.
`set`	`void set (int viti, int muaji, int dita, int ora, int minuta, int sekonda)`	Vendos vlerën për vitin, muajin dhe ditën, orën, minutën dhe sekondën.
`toString`	`String toString()`	E zbërthen datën në string.

Në tabelën 6.18. janë dhënë disa shembuj të thjeshtë të përdorimit të metodave të klasës *Calendar*.

Tabela 6.18. Shembuj në të cilët përdoren disa nga metodat e klasës *Calendar*

Komanda	Vlera e ndryshoreve data dhe rez mbas ekzekutimit të komandave
`Calendar data = Calendar.getInstance();` `int rez = data.get(Calendar.YEAR);`	Për datën = `'Mon 01 April 2013 20:16:35'`, `rez` është `2013`
`Calendar data = Calendar.getInstance();` `int rez = data.get(Calendar.DAY_OF_WEEK)`	Për datën = `'Mon 01 April 2013 20:16:35'`, `rez` është 2 (Sepse 1 – Sunday, 2 – Monday…)

`Calendar data = Calendar.getInstance();`
`calendar.add(Calendar.MONTH,2);`
`int rez = data.get(Calendar.MONTH);`
	Për datën = `'Mon 01 April 2013 20:16:35'`, vlera e atributit `Calendar.MONTH` je 4. Mbas shtimit të vlerës 2 të atributit `Calendar.MONTH` do të ndryshojë data në `'Sat 01 June 2013 20:16:35'`, prandaj vlera e ndryshores rez do të jetë 6.

`Calendar data = Calendar.getInstance();`
`calendar.add(Calendar.HOUR,-4);`
`int rez = data.get(Calendar.HOUR);`
	Për datën = `'Mon 01 April 2013 20:16:35'`, vlera e atributit `Calendar.HOUR` është 20.
	Mbas shtimit të vlerës -4, data e re do të ketë vlerën `'Mon 01 April 2013 16:16:35'`, prandaj vlera e ndryshores rez do të jetë 16.

Shembulli 1. Paraqitja e datës së sotme nëpërmjet klasës *Calendar*.

```
import java.util.Calendar;

public class ShembulliCalendar1 {

    public static void main(String[] args) {

        /* përdorim metodën getInstance() për
           krijimin e objektit të klasës Calendar */
        Calendar cal = Calendar.getInstance();

        /* me ndihmën e metodës getTime( )
           paraqesim datën dhe kohën aktuale      */
        System.out.println("Sot (tani) është: " + cal.getTime());
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/Calendar* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Sot (tani) është: Mon 01 April 2013 20:16:35
```

Shembulli 2. Të paraqitet data e sotme, data e nesërme dhe data para 10 ditësh.

```
import java.util.Calendar;

public class ShembulliCalendar2 {
    public static void main(String[] args) {

        // Krijimi i instancës së klasës Calendar
        Calendar sot = Calendar.getInstance();

        System.out.println("Data e sotme: "
                + sot.get(Calendar.DAY_OF_MONTH) + "-"
                + (sot.get(Calendar.MONTH) + 1) + "-"
                + sot.get(Calendar.YEAR));

        // Shtimi i një dite në raport me ditën momentale, duke përdorur metodën add
        danas.add(Calendar.DATE, 1);

        System.out.println("Data e nesërme: "
                + sot.get(Calendar.DAY_OF_MONTH)
                + "-" + (sot.get(Calendar.MONTH) + 1)
                + "-" + sot.get(Calendar.YEAR));

        // Kthimi i datës për 10 ditë nga data aktuale
        sot = Calendar.getInstance();
        sot.add(Calendar.DATE, -10);

        System.out.println("Data para 10 ditësh: "
                + sot.get(Calendar.DAY_OF_MONTH)
                + "-" + (sot.get(Calendar.MONTH) + 1)
                + "-" + sot.get(Calendar.YEAR));
    }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Data e sotme: 15-5-2014
Data e nesërme: 16-5-2014
Data para 10 ditësh: 5-5 - 2014
```

Projekti i tërë ndodhet në dosjen *Teksti/src/JavaBibliotekaKlasave/Calendar/Shembulli2* në CD.

Pyetje dhe detyra kontrolli

1. Plotëso tabelën me rezultatet e ekzekutimeve të blloqeve të komandave të rradhitura. Nëse komanda do të shkaktojë gabimin, të shënohet "gabimi" dhe të sqarohet shkaku i shfaqes së tij.

Komandat	Vlera e ndryshores rez mbas ekzekutimit të komandave
`Calendar cal = Calendar.getInstance();` `int rez = cal.get(DATE);`	
`Calendar cal = Calendar.getInstance();` `int rez = cal.get(Calendar.DATE);`	

`Calendar cal = Calendar.getInstance();`
`Cal.add(Calendar.MONTH, -10);`
`int rez = cal.get(Calendar.DATE);`	

Puno vetë

1. Shkruani programin që paraqet datën aktuale dhe e shënon emrin e ditës dhe muajin aktual në vit.

2. Shkruani programin që paraqet datën e sotme, datën mbas 15 ditësh në krahasim me ditën e sotme dhe emrin e asaj dite dhe muaji.

3. Shkruani programin që kthen numrin e ditëve në muajin aktual.

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Në klasën *Lojtari* shto:

- Metodën *skadimKontrate*, e cila si argument hyrës pranon datën e nënshkrimit të kontratës si dhe numrin e muajve të angazhimit të lojtarit. Lojtari duhet të paralajmërohet mbi skadimin e kontratës 7 ditë më herët, prandaj metoda kthen datën kur duhet të dërgohet lajmërimi që paraqitet në daljen standarde (p.sh. për kontratën e nënshkruar me '05-05-2013' në periudhën prej 5 muajsh, lajmërimin "Kontrata juaj skadon për 7 ditë!" duhet të dërgohet në datën '28-09-2013').

Projekti *ManariProjekti*. Në klasën *Manari* shto:

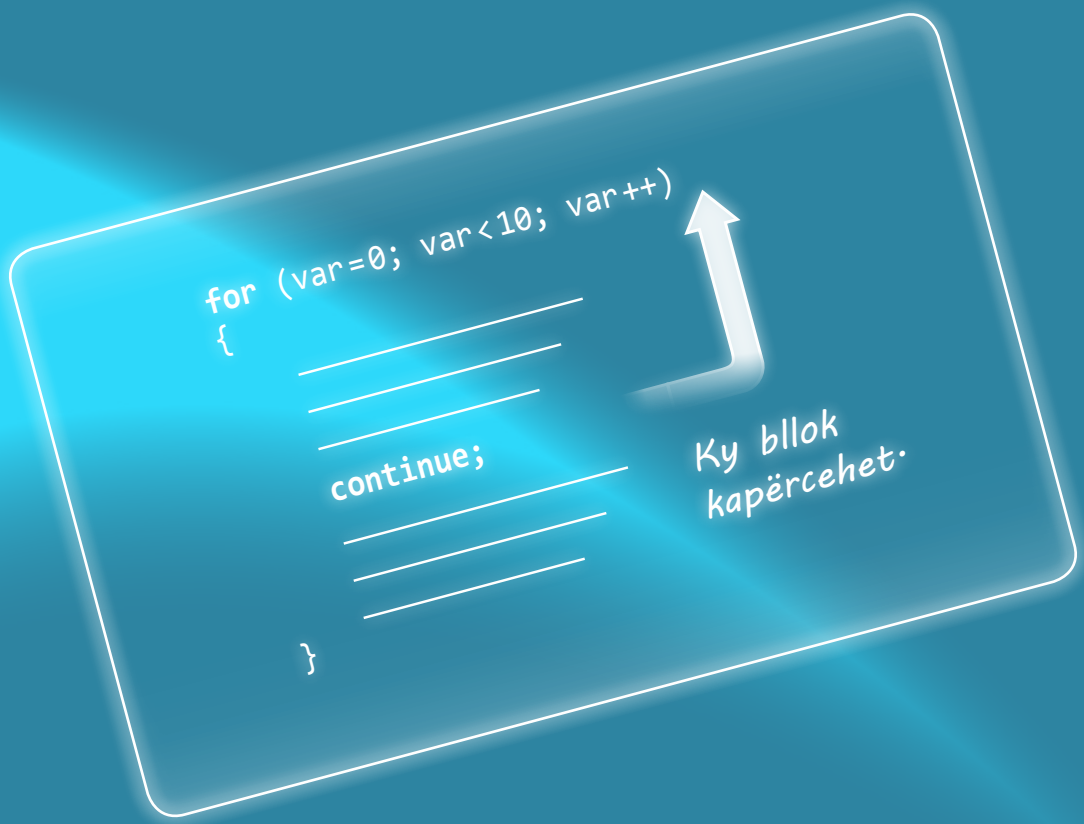
- Metodën *moshaManarit*, që si argument hyrës pranon moshën e manarit në ditën e sotme në formatin *v* vite, *m* muaj dhe *d* ditë. Metoda kthen ditën e lindjes së manarit (p.sh. në datën '05-05-2013' manari ka moshën 1 vjet, 2 muaj dhe 3 ditë, prandaj dita e lindjes së tij është '02-03-2012').

Projekti *QytetetProjekti*. Në klasën *Qyteti* shto:

- Metodën lajmerimiPerDitenEQytetit, që për argumentin hyrës pranon datën të cilën qyteti e feston si Dita e qytetit. Duhet të krijohet lajmërimi mbi festimin 7 ditë më herët, prandaj metoda kthen datën kur duhet të dërgohet lajmërimi që paraqitet në daljen standarde (p.sh. për Ditën e qytetit '05-05-2013', lajmërimin se "Festimi i Ditës së komunës është për 7 ditë!" duhet të dërgohet në datën '28-04-2013').

VII.

KOMANDAT DREJTUESE



Në pjesën e parë të librit jeni njoftuar me skemat algoritmike të kompleksiteteve të ndryshme. Tani mund të shkruani programet që i implementojnë ato skema.

Në këtë kapitull do të njiheni me komandat

- `if`
- `switch`
- `for`
- `while`
- `do-while`

dhe me ndihmën e tyte do të implementoni algoritmet që i keni shkruar në kapitullin I. Gjithashtu, të gjitha programet e shkruara do t'i testoni për vlera të caktuara të parametrave hyrës, të cilat do t'i përmendni në vetë programin. Në kapitullin e mëposhtëm do të mësoni se si këto vlera mund të futen nëpërmjet tastierës ose të lexohen nëpërmjet një skedari hyrës.

Në fillim të Tekstit, në pjesën mbi algoritmet, keni vërejtur se ka shumë pak probleme që mund të zgjidhen në mënyrë sekuenciale, gjegjësisht me një varg të njëpasnjëshëm komandash pa kapërcime në pjesë të tjera të algoritmit. Shpeshherë ndodh që zgjidhja e një probleme të caktuar varet nga plotësimi i kushtit të përkufizuar. Kështu në teorinë e algoritmeve ekzistojnë degëzimet dhe ciklet për kontrollin e rrjedhës së programit (`if`, `while`, `do-while`, `for`). Gjuha programuese Java ka bashkësinë e komandave drejtuese të saj, ashtu që disa prej tyre u përgjigjen komandave të përmendura në teorinë e algoritmeve.

Në këtë kapitull do të njoftoheni me sintaksën e secilës prej tyre dhe shumicën e detyrave për të cilët më herët i kemi përmendur algoritmet, tani do t'i implementojmë dhe testojmë.

7.1. Komanda *if*

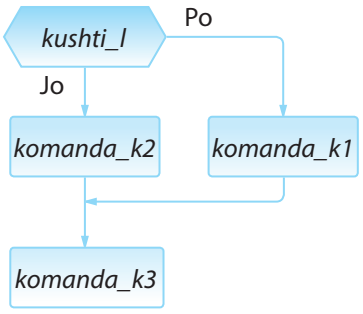
***Komanda* `if`**

Komanda ***if*** në Javë implementon skema algoritmike të degëzuara (të kushtëzuara). Ajo verifikon shprehjen logjike ose vlerën e një ndryshoreje dhe në varësi nga rezultati i tyre vazhdon me ekzekutimin e programit. Sintaksa themelore e komandës `if` është:

```
if(shprehja_logjike) komanda1;
    else komanda2;
```

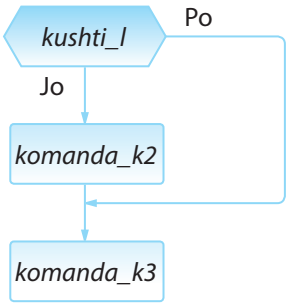
Në qoftë se shprehja logjike është true (e saktë), do të ekzekutohet *komanda1*, në të kundërtën do të ekzekutohet *komanda2*. Fjala kyçe else është fakultative dhe nuk e përcjell doemos komandën `if`. Në figurën 7.1. janë dhënë disa shembuj të implementimit të skemave algoritmike që i kemi përmendur më herët.

a)




```
if (kushti_l)
  komanda_k1;
else
  komanda_k2;
komanda_k3;
```

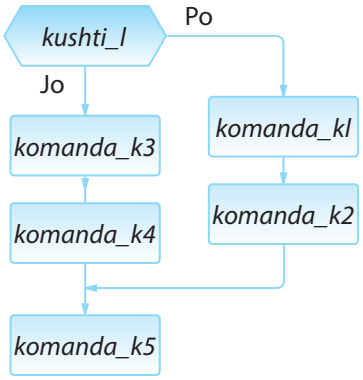
b)




```
if (kushti_l)
   ;
else
  komanda_k2;
komanda_k3;
```

Figura 7.1. Shembuj të implementimit të degëzimit me ndihmën e komandës `if`.

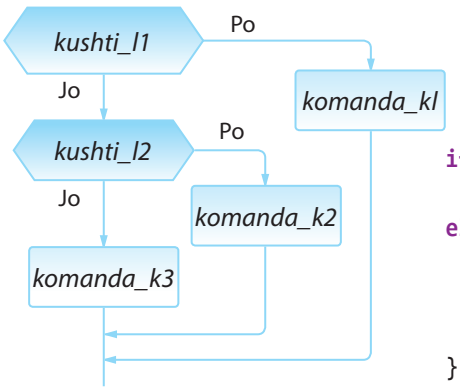
Gjithashtu, është me rëndësi të përmendet se menjëherë mbas `if` dhe `else` komandave mund të vendoset vetëm nga një komandë. Në qoftë se ka më shumë komanda, është e domosdoshme të përdoren blloqet e programit të shënuara me kllapat e mëdha. Shembuj të tillë janë paraqitur në figurën 7.2.




```
if (kushti_l) {
    komanda_k1;
    komanda_k2;
}
else {
    komanda_k3;
    komanda_k4;
}
komanda_k5;
```

Figura 7.2. Shembuj të implementimit duke krijuar blloqet e programit brenda komandës `if`.

Si edhe te zgjidhja algoritmike e problemit, edhe komandat `if` mund të jenë të mbivendosura ndërmjet vete. Në atë rast është i domosdoshëm përdorimi i kllapave të mëdha që në mënyrë të qartë të caktohet se cilës komandë `if` i përket pjesa `else`. Tani do të japim disa shembuj që në mënyrë të qartë tregojnë implementimin e degëzimeve të mbivendosura.




```
if (kushti_l1)
    komanda_k1;
else {
    if (kushti_l2)
        komanda_k2;
    else komanda_k3;
}
```

Figura 7.3. Shembull implementimi të degëzimeve të mbivendosura

Në qoftë se nevojitet të verifikohen më shumë kushte, shpeshherë mund të përdoret konstruksioni `if-else-if`. Fjala është për vargun e mëposhtëm të komandave:

```
if(shprehja_logjike1){
    ...
}
else if(shprehja_logjike2){
    ...
}
else if(shprehja_logjike3){
    ...
}
```

Shembulli 1. Implementimi i **Algoritmit 10** për zgjidhjen e ekuacionit kuadratik $ax^2+bx+c=0$, $a>0$. Programi testohet për vlerat e parametrave $a=1$, $b=2$, $c=1$.

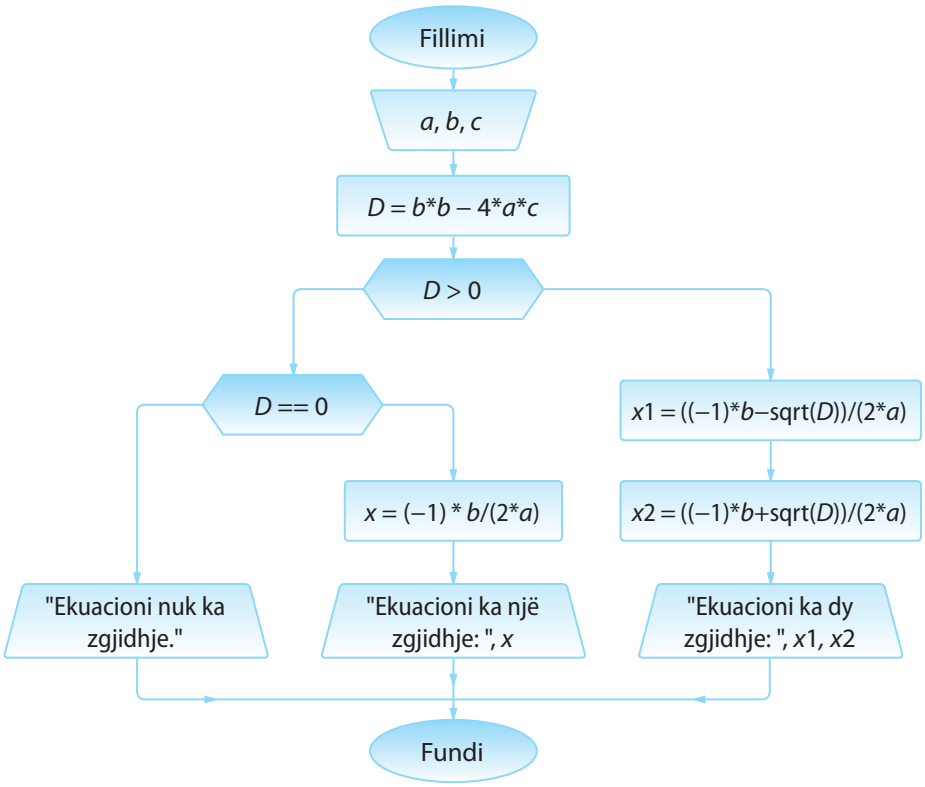
```
if (x > 0)
    if (x == 5) y=5;
else y=x;
```

Komanda paraprake është korrekte nga aspekti sintaksor, mirëpo nuk është përkufizuar në mënyrë korrekte (unike), sepse nuk është e qartë se a është ekzekutimi i komandës y = x nën kontrollin e kushtit x > 0 apo të x == 5. Me qëllim që të zgjidhet kjo moskorrektësi, në vend të konstruksionit të dhënë shënohet

```
if (x > 0) {
    if (x == 5) y=5;
    else y=x;
}
```

Duhet të kihet parasysh se fjala e rezervuar else gjithmonë çiftëzohet me komandën `if` që ndodhet më afër përsipër.

Të implementohen ***algoritmet 8-13*** dhe të testohen për vlera të çfarëdoshme të argumenteve hyrëse.



Algoritmi 10.

```
public class Algoritmi10 {

    public static void main(String[] args) {
        double a = 1.0, b = 2.0, c = 1.0;
        double D = b * b - 4 * a * c;

        if (D < 0)
            System.out.println("Ekuacioni " + a + "*x^2+" + b + "*x+" + c +
                    "=0 nuk ka zgjidhje reale!");
        else
            if (D == 0) {
                double x = ((-1) * b - Math.sqrt(D)) / (2 * a);
                System.out.println("Ekuacioni " + a + "*x^2+" + b + "*x+" + c +
                        "=0 ka saktësisht një zgjidhje: " + x);
            } else {
                double x1 = ((-1) * b - Math.sqrt(D)) / (2 * a);
                double x2 = ((-1) * b + Math.sqrt(D)) / (2 * a);
                System.out.println("Ekuacioni " + a + "*x^2+" + b + "*x+" + c +
                        "=0 ka dy zgjidhje reale, x1=" + x1 + " , x2=" + x2);
            }
    }
}
```

Projekti tërë ndodhet në dosjen *Teksti/src/Komanda_IF* në CD.

Kur programi niset, në daljen standarde do të paraqitet:

```
Ekuacioni 1.0*x^2+2.0*x+1.0=0 a saktësisht një zgjidhje: -1.0
```

Pyetje dhe detyra kontrolli

1. Cilat vlera do t'i kenë ndryshoret *x* dhe *y* mbas ekzekutimit të `if` komandave të mëposhtme? Në qoftë se komanda do të shkaktojë gabimin, të shënohet "gabimi" dhe të sqarohet shkaku i shfaqjes së tij.

Vlera e ndryshores	Komandat	Rezultati i ekzekutimit të komandave
`x = 2;` `y = 4;`	`if (x>y) x=y-1;` `else y--;`	`x=?` `y=?`

`x = 0;`
`y = -5;`
	`if(x+y >= x-y) x=y-1;`
	`else {`
	`  x+=;`
	`  y+=5;`
	`}`
		`x=?`
		`y=?`

`x = 0;` `y = -5;`	`if(x=y) x=x+y;`	`x=?` `y=?`

2. Cila nga linjat e kodit do të ekzekutohen pa gabim?

a)
```
int i=0;
if (i) {
    System.out.println("Hello!");
}
```

b)
```
boolean b=true, b2=true;
if (b==b2) {
    System.out.println("So true");
}
```

c)
```
int i=1, j=2;
if (i==1 || j==2) {
    System.out.println("OK!");
}
```

d)
```
int i=1, j=2;
if (i==1 &| j==2) {
    System.out.println("OK!");
}
```

3. Për cilat vlera të ndryshores *a* ekzekutimi i `if` komandave të mëposhtme nuk do të japë rezultat të njëjtë?

```
if (a>0) b=2;
    else b=5;
if (a<0) b=5;
    else b=2;
```

Puno vetë

1. Të implementohen të gjitha skemat algoritmike që i ke përpunuar më herët në seksionin 1.3.2.

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Në klasën *Lojtari* shto:

- Metodën *shperndaCmimi*, që përcakton nëse lojtari e fiton çmimin (kthen `true`/`false`). Lojtari është fitues i çmimit në qoftë se numri mesatar i golave të shënuar për sezon është më i madh se 1, kurse deri tani ka shënuar jo më pak se 20 gola.
- Metodën *ndihmaSeleksionuesit*, e cila si argument hyrës pranon pozicionin në të cilin seleksionuesit i nevojitet lojtari që të kompletojë ekipin për ndeshje, kurse në daljen standarde paraqet lajmërimin nëse lojtari luan në pozicionin e dhënë apo jo.
- Metodën *kategoriaLojtari*, që kthen kategorinë të cilës i përket sulmuesi:
  - SHËNUES I SHKËLQYESHËM (*mesatarjaSezonit* > 2, *nrGolave* > 15),
  - SHËNUES I MIRË (1 < *mesatarjaSezonit* < 2, *nrGolave* > 10),
  - SHËNUES ME EFIKASITET TË VOGËL (*mesatarjaSezonit* < 1),
  - në rastet e tjera është I PAKATEGORIZUAR.

  Në rast se lojtari nuk është sulmues, të jepet sqarimi përkatës.

Projekti *ManariProjekti*. Në klasën *Manari* shto:

- Metodën *ndryshimiMoshes*, që si argument hyrës pranon vlerën e re që duhet vendosur për moshën e manarit. Në qoftë se vlera e re ndryshon më shumë se 25 % në raport me vlerën paraprake, nuk bëhet ndryshimi, por jepet sqarimi përkatës.
- Metodën *kontrolli*, që kthen `true` në qoftë se vlerat e të gjitha atributeve janë të njohura (d.m.th. në qoftë se vlera e atributit *raca* nuk është e barabartë me "e panjohur"). Në të kundërtën metoda kthen `false`.
- Metodën *kategoriaManari*, që kthen kategorinë të cilës i përket manari nga lloji i qenve:
  - KONE (*mosha* < *6 muaj*),
  - JUNIOR (*6 muaj* <= *mosha* < *16 muaj*),
  - I RRITUR (anglisht ADULT) (*16 muaj* <= *mosha* < *5 vjet*),
  - QEN I VJETËR (*mosha* >= *5 vjet*).

  Në rast se manari nuk është qen, të jepet sqarimi përkatës.

Projekti *QytetetProjekti*. Në klasën *Qyteti* shto:

- Metodën *qytetMasiv*, që kthen `true` në qoftë se qyteti ka më shumë se një milion banorë. Në të kundërtën, kthen `false`.
- Metodën *numriKorrektPostar*, që kthen `true`/`false` varësisht, nëse kodi postar është futur si duhet apo jo. Numri postar është korrekt nëse përmban saktësisht 5 shifra.
- Metodën *kategoriaQyteteve*, që kthen kategorinë në bazë të numrit të banorëve:
-
  - QYTET I VOGËL (numri i banorëve <= 20 000),
  - QYTET I MESËM (20 000 < numri i banorëve <= 50 000),
  - QYTET I MESËM (50 000 < numri i banorëve <= 1 000 000),
  - QYTET MASIV (1 000 000 < numri i banorëve <= 10 000 000),
  - QYTET MEGA MASIV (numri i banorëve > 10 000 000).

Vërejtje: Gjatë zgjidhjes së detyrave paraprake në veçanti të përfshihen rastet kur nuk janë të njohura vlerat e atributeve përkatëse.

7.2. Komanda *switch*

Komanda ***switch*** sipas objektivit të vet është e ngjashme me `if-else` dhe mund të përdoret kur duhet të kontrollohet një numër i madh kushtesh. Sintaksa themelore e komandës `switch` është:

```
switch(shprehja) {
    case konstantja1:
        ...
        break;
    case konstantja2:
        ...
        break;
    case konstantja3:
        ...
        break;
    default:
        ...
        break;
    }
```

Vlera e ndryshores krahasohet, sipas radhës me konstantet dhe kur arrihet te përputhja, ekzekutohen të gjitha komandat që pasojnë. Komanda `break` e ndërpret ekzekutimin e bllokut `switch` dhe kështu siguron që të kryhen vetëm komandat që janë të lidhura me vlerën e dhënë.

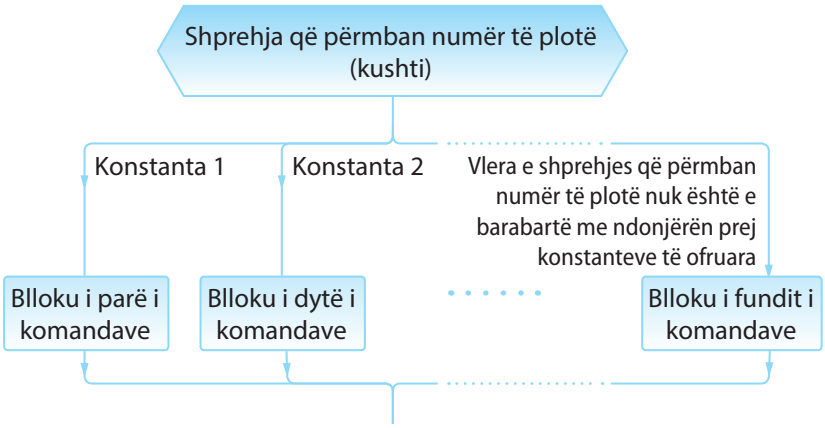


Figura 7.4. Paraqitja grafike e komandës `switch`

Në qoftë se mbas një grupi operatorësh në `switch` evitohet shënimi i operatorit break, atëherë në rast se zgjidhet ai grup, do të kryhen edhe alternativat e tjera deri te paraqitja e break ose fundi i operatorit switch. Shpeshherë thuhet se në atë rast vjen deri te e ashtuquajtura ***rënia e ndryshores*** nëpër bllokun `switch`.

Kështu në shembullin:

```
switch (dita) {
   case 1:
      System.out.println("E HËNE"); break;
   case 2:
      System.out.println("E MARTË"); break;
   case 3:
      System.out.println("E MËRKURË");
```

Komanda SWITCH

Dallimi kryesor ndërmjet komandave ***switch*** dhe ***if*** është në faktin se `switch` mund të kontrollojë vetëm barazimin e një ndryshoreje me konstanten e përmendur, kurse komanda if mund të njehsojë çdo shprehje logjike. Me fjalë të tjera, `switch` vetëm verifikon, nëse vlera e ndryshores së përmendur përputhet me vlerën e ndonjë konstanteje të përmendur në blloqet `case`.

Për shembull, komandën `switch` mund ta përdorim, nëse nevojitet që në bazë të vlerave numerike 1 – 7 të shënohen emrat e ditëve në javë *E hënë – E diel*.

```
switch (dita) {
 case 1:
   System.out.println("E hënë");
   break;
 case 2:
   System.out.println("E martë");
   break;
 case 3:
   System.out.println("E mërkurë");
   break;
 case 4:
   System.out.println("E enjte");
   break;
 case 5:
   System.out.println("E premte");
   break;
 case 6:
   System.out.println("E shtunë");
   break;
 case 7:
   System.out.println("E diel");
   break;
 default:
   System.out.println("VLERË
JOKORREKTE për numrin rendor të
ditës në javë...");
}
```

Përcakto vlerat e ndryshores *numri* mbas ekzekutimit të komandave të përmendura `switch` në qoftë se vlera fillestare e saj është 2.

```
// a)
switch(numri) {
  case 1:
    numri=2;
    break;
  case 2:
    numri=3;
  case 3:
    numri=4;
    break;
  default:
    numri=5;
}

// b)
switch(numri) {
  case 1: case 2:
    numri++;
    break;
  case 3: case 4:
    numri--;
    break;
  default:
    numri=5;
}
```

```
        case 4:
           System.out.println("E ENJTE");
        case 5:
           System.out.println("E PREMTE");
        case 6:
           System.out.println("E SHTUNE"); break;
        case 7:
           System.out.println("E DIEL"); break;
        default:
           System.out.println("VLERË JOKORREKTE për numrin rendor " +
                              "të ditës në javë...");
     }
```

për vlerën e ndryshores `dita=4`, në daljen standarde do të jetë e shënuar:

```
E ENJTE
E PREMTE
E SHTUNË
```

Në qoftë se komanda `default` nuk ekziston, dhe paraprakisht nuk është gjetur përputhja e vlerës së ndryshores me konstantat e dhëna, do të dilet nga `switch` blloku dhe do të vazhdojë ekzekutimi i programit nga komanda vijuese. Në bllokun e komandave `default` nuk është i domosdoshëm përdorimi i komandës `break`, sepse mbas ekzekutimit të bllokut `default` menjëherë do të dilet nga komanda `switch`.

Duhet të ceket, se një blloku të komandave i është e lejueshme t'i shoqërohen më shumë konstante, si në shembullin e mëposhtëm:

```
switch (muaji) {
  case 1: case 3: case 5: case 7: case 8: case 10: case 12:
    System.out.println("Muaji ka 31 ditë!");
    break;
  case 4: case 6: case 9: case 11:
    System.out.println("Muaji ka 30 ditë!");
    break;
  case 2:
    System.out.println("Shkurti ka 28 ditë përveç në rastin e " +
                       "vitit të brishtë kur ka 29 ditë!");
    break;
  default:
    System.out.println("VLERË JOKORREKTE për numrin rendor të " +
                       "muajve në vit...");
}
```

Shembulli 2. Klasës *Nxenesi* t'i shtohet metoda *paraqitjaNota* për paraqitjen e notës nga matematika në gjuhën shqipe:

```
class Nxenesi {
    int    numriEvidences;
    String emri;
    String mbiemri;
    String shkolla;
    String paralelja;
    int    notaNgaMatematika;
```

```
    public void paraqitjaNotes() {
        System.out.print("Nota e nxenesit nga matematika është: ");

        switch (notaNgaMatematika) {
        case 1:
            System.out.println("PAMJAFTUESHËM.");
            break;
        case 2:
            System.out.println("MJAFTUESHËM.");
            break;
        case 3:
            System.out.println("MIRË.");
            break;
        case 4:
            System.out.println("SHUMË MIRË.");
            break;
        case 5:
            System.out.println("SHKËLQYESHËM!");
            break;
        default:
            System.out.println("I PANOTUAR...");
            break;
        }

    }
}
```

Metoda *main* i klasës *TestNxenesi* bën thirrjen e metodave të krijuara në mënyrën e mëposhtme:

```
public class TestNxenesi  {

  public static void main(String[] args) {
    Nxenesi nxenesi = new Nxenesi();
    nxenesi.notaNgaMatematika = 5;
    nxenesi.paraqitjaNotes();
  }

}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Nota e nxenesit nga matematika është: SHKËLQYESHËM!
```

Projekti tërë ndodhet në dosjen *Teksti/src/Komanda_SWITCH/Nxenesi* në CD.

Pyetje dhe detyra kontrolli

1. Të shkruhet komanda `switch` që është ekuivalente me `if` komandën e mëposhtme:

```
if (k==5) r++;
else {
   if (k==4) r--;
   else {
      if (k==3) or (k==2) or (k==1) r=0;
   }
}
```

2. 2. Cilat janë tipat e lejueshëm të të dhënave për ndryshoren *x* në pjesën e mëposhtme të kodit?

1. `byte` 2. `long`
3. `char` 4. `float`
4. `Short` 6. `Long`

```
switch(x){
    default: System.out.println("Hello...");
}
```

a) 1 dhe 3
b) 2 dhe 4
c) 3 dhe 5
d) 4 dhe 6

3. Në cilën nga linjat duhet shënuar komanda break; në mënyrë që në daljen standarde të paraqitet simboli * gjashtë herë (d.m.th. ******).

```
int a = 1;
switch (a) {
   case 0: System.out.print("*");    //linja 3
   case 1: System.out.print("**");   //linja 4
   default: System.out.print("***"); //linja 5
}
```

a) Mbas secilës nga linjat 3, 4 dhe 5.
b) Mbas linjave 3 dhe 4.
c) Nuk është e mundshme që në dalje të përftohen gjashtë *.
d) Nuk duhet të vendoset `break` në ndonjë linje.

Puno vetë

1. Të shkruhet programi që për numrin rendor të muajit në vit të paraqesë emrin e atij muaji.

2. Të shkruhet programi që për veprimin algjebrik të dhëna nga bashkësia (+,-,*,/,%) të futur si string dhe për vlerën e dy operandave të plotë të njehsojë dhe të paraqesë rezultatin.

Testoji të dy detyrat për vlera të çfarëdoshme të parametrave përkatës.

7.3. Cikli *for*

Në teorinë e algoritmeve jeni njohur me ciklin ***for***, që është forma më e shpeshtë e ciklit për punë që duhet përsëritur saktësisht një numër të caktuar herësh. Sintaksa themelore e këtij cikli në gjuhën programuese Java është:

```
for (shprehja1; shprehja_logjike; shprehja2) {
    ...
}
```

Shprehja *shprehja1* i jep instruksione ciklit for se prej cilave vlera fillon kontrolli i ndryshoreve (d.m.th. ndryshorja që përcakton numrin e përsëritjeve të veprimeve brenda ciklit for). *Shprehja_logjike* paraqet kushtin themelor të ciklit. Cikli përsëritet gjithnjë derisa kushti është plotësuar. Shprehja e fundit, *shprehja2*, i tregon ciklit se si ndryshorja e kontrollit ndryshon mbas secilit hap (d.m.th mbas kalimit nëpër cikël).

Cikli FOR

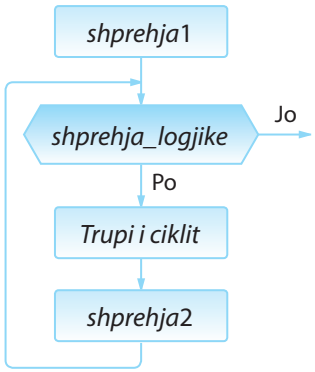


Figura 7.5. Paraqitja algoritmike e ciklit `for`

Për ciklin `for` vlen vërejtja e njëjtë si për komandën `if`: me komandën `for` përfshihet komanda e parë vijuese e programit. Nëse duhet të bëhet ekzekutimi i më shumë komandave, duhet të përdoren blloqet e programit (të shënuar me kllapa të mëdha).

Për përdorimin e ciklit `for` le të japim edhe disa vërejtje dhe disa situata karakteristike me të cilat mund të takoheni gjatë punës:

- Ndryshorja e kontrollit të cilën cikli e përdor duhet të deklarohet paraprakisht. Java mundëson që deklarimi të bëhet në vetë ciklin, d.m.th pjesa e kodit:

  ```
  int i;
  for(i=1; i<=10; i++)
      System.out.println("Une jam maturant...");
  ```

  është ekuivalente me:

  ```
  for(int i=1; i<=10; i++)
      System.out.println("Une jam maturant...");
  ```

- Shprehja që përcakton mënyrën e ndryshimit të ndryshores kontrolluese mund të evitohet. Në atë rast është e domosdoshme që në bllokun e komandave brenda ciklit të përkufizohet mënyra e ndryshimit të ndryshores kontrolluese. Në të kundërtën, do të kishim ciklin që ekzekutohet pafund herë (sepse kushti i daljes nga cikli nuk do të jetë i plotësuar për shkak të vlerës konstante të ndryshores kontrolluese). Domethënë, cikli `for` paraprak është ekuivalent me

  ```
  for (int i=1; i<=10; ) {
      System.out.println("Ja sam maturant...");
      i++;
  }
  ```

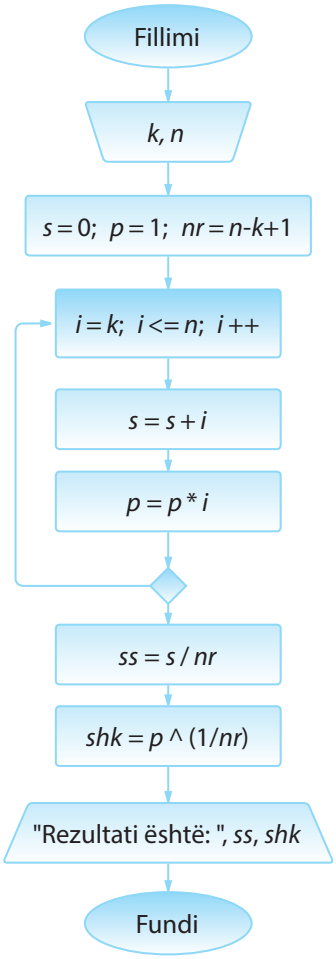
- Është e mundshme që cikli `for` nuk ka shprehjen për inicializimin e ndryshores kontrolluese, mirëpo në atë rast vlera i është shoqëruar para hyrjes në cikël. Cikli paraprak është ekuivalent me

  ```
  int i=1;
  for (; i<=10; i++)
      System.out.println("Une jam maturant...");
  ```

- Ekzekutimi i ciklit mund të kontrollohet edhe me më shumë ndryshore. Në atë rast cikli përmban më shumë se një shprehje për inicializimin e vlerës, si dhe për ndryshimin e vlerave të ndryshoreve kontrolluese (ato shprehje janë të ndara me presje). Kështu në shembullin e mëposhtëm kemi ndryshoret kontrolluese *a* dhe *b*, ashtu që ndryshorja e parë shumëzohet me 2, kurse e dyta pjesëtohet me 2, derisa vlera e ndryshores *b* të bëhet më e vogël se vlera e ndryshores *a*.

```
int a, b;
for (a=10, b=100; a<b; a=a*2, b=b/2)
    System.out.println("Vlera a=" + a +
                       " Vlera b=" + b);
```

Shembulli 3. Implementimi i **Algoritmit 19** për njehsimin e të mesmes aritmetike dhe gjeometrike të numrave natyrorë nga intervali [*k*, *n*], *k* < *n*. Programi testohet për vlerat e parametrave *k* = 5, *n* = 10.



Algoritmi 19.

```
public class Algoritmi19 {

   public static void main(String[] args) {
      int k=5, n=10;
      int s=0, p=1, nr=n-k+1;

      for (int i=k; i<=n; i++) {
          s = s + i;
          p = p * i;
      }

      double ss = s;
      ss = ss / nr;

      double shk = nr;
      shk = 1/shk;
      shk = Math.pow(p, shk);

      System.out.println("E mesmja aritmetike është " + ss +
                ", kurse e mesmja gjeometrike është " + shk);
   }
}
```

Detyra ndodhet në CD, në dosjen *Teksti/src/Komanda_FOR_DO_WHILE*

Kur programi niset në daljen standarde do të paraqitet:

```
E mesmja aritmetike është 7.5, kurse e mesmja gjeometrike është
7.298920475286468
```

Pyetje dhe detyra kontrolli

1. Sa herë do të përsëritet cikli, në qoftë se cikli `for` është përkufizuar në mënyrën e mëposhtme:

a) `for (int i = -1; i <= 1; i++)`

b) `for (int i = 100; i >90; i--)`

c) `for (int i = 100; i >= 90; i--)`

2. Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së kodit?

```
int y=2;
for (int z = 0; z < 3; z++) {
    if (z!=y) {
        System.out.println("z= ",z);
    }
}
```

3. Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së kodit?

```
int shuma=0;
for (int i = 1; i < 5; i++) {
    shuma=shuma+i;
}
System.out.println("shuma= ",shuma);
```

4.* Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të kodit të mëposhtëm? Të zgjidhen tri përgjigje.

```
public class K3 {
    public static void main(String[] args) {
        for (int i = 0; i < 2; i++) {
            for (int j = 2; j >=0; j--) {
                if(i==j) break;
                System.out.println("i="+i+", j="+j);
            }
        }
    }
}
```

a) `i=1, j=2`

b) `i=0, j=1`

c) `i=1, j=2`

d) `i=0, j=2`

e) `i=1, j=1`

f) `i=0, j=2`

g) `i=2, j=2`

h) `i=2, j=1`

Puno vetë

1. Implementoji të gjitha skemat algoritmike që i ke shkruar më parë në seksionin 1.3.3.1.

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Në klasën *Lojtari* shto:

- Metodën *nrPergjithshemGolave*, që si argument hyrës pranon numrin e mbetur të ndeshjeve deri në fund të sezonit, si dhe numrin e golave që pritet që lojtari t'i japë për ndeshje. Metoda paraqet se si do të ndryshojë numri i golave të shënuara sikur në secilën ndeshje nga ndeshjet e mbetura, lojtari do të jepte numrin e pritur të golave.

  P.sh.: Në qoftë se kanë mbetur edhe 5 ndeshje, për lojtarin që ka shënuar 3 gola, dhe pritet që të japë mesatarisht nga një (1) gol, metoda do të paraqesë:

  Mbas ndeshjes së parë, lojtari do të ketë gjithsej 4 gola

  Mbas ndeshjes së dytë, lojtari do të ketë gjithsej 5 gola

  ...

Projekti *ManariProjekti*. Në klasën *Manari* shto:

- Metodën *paraqitjaMoshesQenit*, e përcaktuar për qentë, manaret, ashtu që manarit të caktuar, për secilin muaj të jetës, i paraqet kategorinë përkatëse. Mosha e parë është deri në 6 muaj (kone), mosha e dytë deri në 16 muaj (junior), mosha e tretë deri në 5 vjet (qen i rritur), dhe mosha e katërt (qen i vjetër). Për manarët e tjerë jep lajmërimin përkatës që në këtë mënyrë nuk mund të bëhet kategorizimi.

  P.sh. Për qentë e moshës 18 muaj, metoda do të paraqesë:

  Muaji i parë: qeni i takon kategorisë KONE

  ...

  Muaji i 5-të: qeni i takon kategorisë KONE

  Muaji i 6-të: qeni i takon kategorisë JUNIOR

  ...

  Muaji i 15-të: qeni i takon kategorisë JUNIOR

  Muaji i 16-të: qeni i takon kategorisë QEN I RRITUR

  Muaji i 17-të: qeni i takon kategorisë QEN I RRITUR

  Muaji i 18-të: qeni i takon kategorisë QEN I RRITUR

Projekti *QytetetProjekti*. Në klasën *Qyteti* shto:

- Metodën *krijimiNumraveRiPostar*, që si argument hyrës pranon numrin e tërësive të reja postare për të cilat duhet të gjenerohet numri postar. Numrat postarë nga numrat postarë ekzistues të qytetit krijohen duke shtuar këto vlera... , etj.

P.sh. Për qytetin Podgorica, numri postar i të cilit është 81 000, metoda do të paraqesë numrat postarë të mëposhtëm për 5 kode të reja postare:

Kodi postar 1: 81 005

Kodi postar 2: 80 995

Kodi postar 3: 81 010

Kodi postar 4: 80 990

Kodi postar 5: 81 015

Vërejtje: Gjatë zgjidhjes së detyrave të dhëna në veçanti të shqyrtohen rastet kur nuk janë të njohura vlerat e atributeve përkatëse.

7.4. Cikli *while* dhe *do-while*

Me ciklet ***while*** dhe ***do-while*** jeni takuar gjithashtu në skemat algoritmike, kurse sintaksa themelore e tyre në gjuhën programuese Java është:

```
while(shprehja_logjike) {
    ...
}
```

dhe

```
do {
    ...
} while (shprehja_logjike);
```

Vetia më e rëndësishme e ciklit `while` (që e kemi theksuar edhe në algoritme, por është me rëndësi të përsëritet) është se kushti (d.m.th. shprehja logjike) kontrollohet gjatë hyrjes në cikël, ashtu që mund të ndodhë që cikli të mos ekzekutohet ndonjëherë (në qoftë se vlera e shprehjes logjike është `false`), si në shembullin:

```
int i=10, j=100;
while (i > j) {
    ...
}
```

Për dallim nga cikli `while`, kushti logjik te `do-while` verifikohet mbas ekzekutimit të bllokut të komandave, ashtu që komandat ekzekutohen së paku një herë.

Kështu, në shembullin e mëposhtëm printimi i mesazhit do të kryhet gjatë kalimit të parë nëpër cikël, mirëpo fill mbas atij kalimi, kushti nuk do të jetë i plotësuar, me çfarë ndërpritet ekzekutimi i komandave.

```
int i=10, j=100;
do {
    System.out.print("Printimi... ");
} while (i > j);
```

Cikli WHILE dhe DO-WHILE

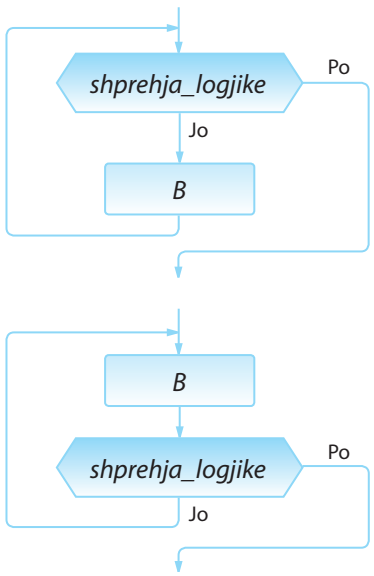
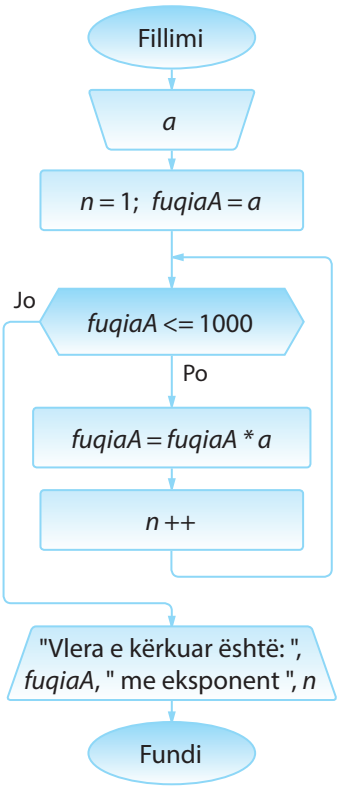


Figura 7.6. Paraqitja algoritmike e cikleve `while` dhe `do-while`.

Implemento **algoritmet 17**, **19** dhe **20** dhe testoji për vlera të çfarëdoshme të argumenteve hyrëse.



Algoritmi 22.

Detyra ndodhet në CD-në në dosjen *Teksti/src/Komanda_FOR_WHILE_DO_WHILE*

Implemento **algoritmet 23 - 26** dhe testoji për vlera të çfarëdoshme të argumenteve hyrëse.

Shembulli 4. Implementimi i **Algoritmit 22** me të cilin për vlerën e dhënë të numrit a përcaktohet numri më i vogël i formës a^n që është më i madh se numri 1000. Programi testohet për vlerën $a = 3$.

```
public class Algoritmi22 {

    public static void main(String[] args) {
        int a = 3, n = 1, fuqiaA = a;

        while (fuqiaA <= 1000) {
            System.out.println("vlera e shprehjes a=" + a +
                               " në fuqinë n=" + n +
                               " është:" + fuqiaA);
            n++;
            fuqiaA = fuqiaA * a;
        }

        System.out.print("Rezultati është: ");
        System.out.println("vlera e shprehjes a=" + a +
                           " në fuqinë n=" + n +
                           " është:"  + fuqiaA);
    }
}
```

Kur programi të niset, në daljen standarde do të paraqitet:

```
vlera e shprehjes a = 3 në fuqinë n = 1 është: 3
vlera e shprehjes a = 3 në fuqinë n = 2 është: 9
vlera e shprehjes a = 3 në fuqinë n = 3 është: 27
vlera e shprehjes a = 3 në fuqinë n = 4 është: 81
vlera e shprehjes a = 3 në fuqinë n = 5 është: 243
vlera e shprehjes a = 3 në fuqinë n = 6 është: 729
Rezultati është: vlera e shprehjes a = 3 në fuqinë n = 7 është: 2187
```

Pyetje dhe detyra kontrolli

1. Plotëso vlerat që mungojnë në tabelën e mëposhtme:

Komanda	Numri i përsëritjeve të ciklit	Vlera mbas ekzekutimit të komandave

`a=1; b=1;`
`while (a<=b) {`
`  a++;`
`  b--;`
`}`		

a=1; b=1;
while (a<=3) {
  a++;
}
b--;		

a=5; p=1;
do {
  a--;
  p=p*a;
} while (a>1);		

2. Çfarë duhet të vendoset në vendin e shënuar me simbolin ??? në mënyrë që cikli i përmendur të ekzekutohet saktësisht 5 herë?

a)
```
a=25; b=5;
while (???) {
    if (a>b) a=a-b;
    else b=b-a;
}
```

1. b<5
2. a!=b
3. b<a
4. a>5
5. Asnjë prej përgjigjeve të ofruara

b)
```
a=10; b=100;
do {
    a++;
    if (b>10*a) b/=5;
} while (???)
```

1. b>a
2. b>5*a
3. b<125
4. a<=15
5. Asnjë prej përgjigjeve të ofruara

3. Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së kodit?

```
int y=2, z=0;
while (z < 3) {
    if (z!=y) {
        System.out.println("z= ",z);
    }
    z++;
}
```

4. Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së kodit?

```
int shuma=0, i=1;
do {
    shuma=shuma+i;
} while (i < 5);
System.out.println("shuma = ",shuma);
```

Puno vetë

1. Implementoji të gjitha skemat algoritmike që i ke shkruar më herët në seksionin 1.3.3.2.

Përgatitu që në grup të punosh projektin

Projekti *LojtartProjekti*. Në klasën *Lojtari* shto:

- Metodën *pozicioniRi*, që si argument hyrës pranon numrin e rrotacioneve që lojtari duhet t'i kryejë. Rrotacionet bëhen duke filluar nga pozicioni momental për një vend djathtas në rendin: *sulmues*, *mbrojtës*, *sulmuesi në krah*, *ndalues*, *lojtar mesfushe*. Portierët nuk rrotullohen. Metoda kthen pozicionin e lojtarit mbas ekzekutimit të të gjitha rrotacioneve dhe njëkohësisht paraqet pozicionin mbas secilit rrotacionit.

  P.sh. Për lojtarin që është momentalisht në pozicionin e sulmuesit në krah, mbas 3 rrotacioneve metoda do të paraqesë:

  Mbas rrotacionit të parë, lojtari është në pozicionin e ndaluesit!

  Mbas rrotacionit të dytë, lojtari është në pozicionin e lojtarit të mesfushës!

  dhe rezultati do të jetë *sulmues*.

Projekti *ManariProjekti*. Në klasën *Manari* shto:

- Metodën *vleresimiVjeteve* që për argument hyrës e pranon objektin e klasës *Manari* dhe përcakton për sa muaj një manar do të jetë dy herë më i vjetër se manari tjetër. Metoda tregon raportin e moshës së manarëve për çdo muaj vijues, deri sa të arrihet kushti i përmendur ose njëri prej tyre të arrijë moshën 20 vjeçare.

  P.sh. Për dy manarë të moshës 2 muaj dhe 7 muaj, metoda do të paraqesë:

  Muaji i parë: Manari I 3 muaj, Manari II 8 muaj, raporti: 2.67

  Muaji i dytë: Manari I 4 muaj, Manari II 9 muaj, raporti: 2.25

  Muaji i tretë: Manari I 5 muaj, Manari II 10 muaj, raporti: 2.0

  dhe do të kthejë vlerën 3.

Projekti *QytetetProjekti*. Në klasën *Qyteti* shto:

- Metodën *qytetiMasivNeTArdhmen*, e cila për një qytet me më pak se një milion banorë paraqet se si do të duket ndryshimi i numrit të përgjithshëm të popullsisë sikur të vazhdojë trendi i rritjes së numrit të banorëve për çdo vit nga 50 000, kurse çdo të pestin vit, numri i banorëve ulet për 1000. Paraqitja përfundon kur qyteti ka të paktën një milion banorë.

  P.sh. Për qytetin që ka 750 000 banorë, metoda do të paraqesë:

  Viti i parë, numri i banorëve: 800 000
  Viti i 2-të, numri i banorëve: 850 000
  Viti i 3-të, numri i banorëve: 900 000
  Viti i 4-të, numri i banorëve: 950 000
  Viti i 5-të, numri i banorëve: 949 000
  Viti i 6-të, numri i banorëve: 999 000
  Viti i 7-të, numri i banorëve: 1049 000

Vërejtje: Gjatë zgjidhjes së detyrave të dhëna në veçanti të shqyrtohen rastet kur nuk janë të njohura vlerat e atributeve përkatëse.

7.5. Komandat *break*, *return* dhe *countinue*

Siç është sqaruar më herët, dalja nga cikli përkufizohet me kushtin logjik, ashtu që cikli përsëritet deri sa kushti të jetë plotësuar. Mirëpo në Javë ekzistojnë edhe komandat ***break***, ***return*** dhe ***continue***, nëpërmjet të cilave mund të kontrollohet dalja nga cikli në mënyrën e mëposhtme.

Ekzekutimi i cikleve mund të ndërpritet duke përdorur komandën break. Në qoftë se në ndonjë iteracion (përsëritje) ekzekutohet komanda ***break***,**cikli në të njëjtin moment ndërpritet dhe vazhdon me ekzekutimin e komandave që pasojnë mbas ciklit**.

Shembulli 5. Ilustrimi i komandës `break`

```
public static void main(String[] args) {
    int i;
    for (i=1; i<=5; i++) {
        if (i==3) break;
        System.out.println("i = " + i);
    }
    System.out.print("Shembull komande BREAK!");
}
```

Kur të niset programi, në daljen standarde do të paraqitet:

```
i=1
i=2
Shembull komande BREAK!
```

Gjithashtu, ndërprerja e ciklit mund të realizohet duke thirrur komandën ***return***. Në qoftë se metoda në të cilën përdoret cikli kthen një vlerë, cikli mund të ndërpritet duke thirrur komandën `return`, që njëkohësisht shkakton ndërprerjen e ekzekutimit të metodës së tërë.

Shembulli 7. Ilustrimi i komandës `return` për ndërprerjen e ciklit

```
public int plotpjesëtueshëmMe9(int a, int b) {
    for (int i=a; i<=b; i++) {
        if (i%9 == 0) return i;
    }
    return -1;
}
```

Në rast të thirrjes së `plotpjesëtueshmeMe9(10, 20)`, dalja është 18, kurse në rastin e thirrjes së `plotpjesëtueshmeMe9(10, 15)`, dalja është `-1`.

Të implementohen algoritmet **27**, **28**, **29**, **31**, **33**, **34** dhe **35** dhe të testohen për vlera të çfarëdoshme të argumenteve hyrëse.

Komanda break

Shembulli 6.
Të përcaktohet vlera e ndryshores *i* mbas ekzekutimit të pjesës së kodit

```
int i=1;
while (i<10) {
   if(i==5) break;
   System.out.println("i="+i);
   i++;
}
```

Zgjidhja. Për vlerat e ndryshores *i* prej 1 deri 4, kushti brenda komandës `if` nuk do të jetë i plotësuar, prandaj në daljen standarde do të paraqiten sipas radhës:

```
i = 1
i = 2
i = 3
i = 4
```

Për `i=5`, kushti i komandës `if` është i plotësuar dhe cikli `while` përfundon me punën e vet.

Komanda return

Shembulli 8.
Është dhënë metoda `max(a,b,c)`:

```
public int max(int a, int b, int c){
   if (a>=b) && (a>=c) return a;
   if (b>=a) && (b>=c) return b;
   return c;
}
```

Të përcaktohet dalja në rastin e thirrjes së `max(10,8,5)` dhe `max(5,10,9)`.

Zgjidhje. Në rastin e thirrjes së `max(10,8,5)`, kushti në komandën e parë IF është i plotësuar dhe metoda kthen vlerën e argumentit të parë (a) 10.

Në rastin e thirrjes së max(5,10,9), kushti në komandën e parë IF nuk është i plotësuar, prandaj verifikohet kushti në komandën e dytë IF. Ky kusht është i plotësuar, prandaj metoda e kthen vlerën e argumentit të dytë (b) 10.

Komanda continue

Në Javë ekziston edhe komanda, ekzekutimi i të cilës siguron që të përsëritet iteracioni momental dhe të vazhdohet me iteracionin pasues në cikël. Fjala është mbi komandën ***continue***. Kur ajo ekzekutohet, të gjitha komandat që do të duhej të ekzekutohen në atë iteracion, dhe janë të shënuara nën këtë komandë, kapërcehen dhe kalohet në iteracionin pasues. Komanda e vetme që ekzekutohet në mënyrë të rregullt në rastin e ciklit `for` është komanda brenda kllapave të ciklit me të cilën zmadhohet ose zvogëlohet vlera e numëruesit (të ndryshores kontrolluese).

Shembulli 9. Ilustrimi i komandës `continue`

```
public class Algoritmi19 {

   public static void main(String[] args) {
      int i;
      for (i=1; i<=5; i++) {
         if (i==3) continue;
         /* Komandat që vijojnë
            nuk ekzekutohen për i=3 */
         System.out.println("i = " + i);
      }
   }
}
```

Kur të niset programi, në daljen standarde do të paraqitet:

```
i = 1
i = 2
i = 4
i = 5
```

Primjer 10.
Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së mëposhtme të kodit:

```
System.out.println("Numra të plotpjesëtueshëm me
5 dhe 9: ");

int i=9;
while (i<100){
   i++;
   if (i%5 != 0) || (i%9 != 0)
      continue;
   System.out.println(i);
}
```

Zgjidhje.
Pjesa e dhënë e kodit do të paraqesë të gjithë numrat dyshifrorë që janë të plotpjesëtueshëm me 5 dhe 9:

```
Numra të plotpjesëtueshëm me 5 dhe 9:
45
90
```

Shembulli 11. Të krijohet klasa *DetyraMatematika* që përmban:

- Metodën *shumefishiMeIVogelIPerbashketNgaIntervali*,e cila duhet të përcaktojë numrin më të vogël të plotë nga intervali 10 deri në 1000 që është i plotpjesëtueshëm me numrat *a*, *b* dhe *c* që jepen si argumente hyrëse. Mbas përfundimit të punës, metoda paraqet mesazhin *Nxenesi Emri-Mbiemri ka zgjidhur detyrën*!
- Metodën *numerIThjesht*, që për vlerë të parametrit hyrës *n* verifikon nëse është numër i thjeshtë apo jo dhe kthen `true`/`false`.
- Metodën *main* për testimin e metodave të krijuara.

```
public class DetyraMatematika {

   public static void intervalSHVP(int a, int b, int c) {
      int i;

      for (i = 10; i <= 1000; i++) {
         if (i % a == 0 && i % b == 0 && i % c == 0) {
            /*
               është gjetur numri dhe dalim nga cikli,
               ekzekutimi vazhdon në (1)  */
            System.out.println("Numri i kërkuar është: " + i);
            break;
         }
      }

      /* (1) kontrolli a kemi mbërri deri në fund të ciklit,
             kurse rezultati nuk është gjetur  */
      if (i == 1001)
         System.out.println("Nuk ekziston numri nga [10, 1000]" +
                            "që është i plotpjesëtueshëm me " + a + ", " +
                             b + ", " + c + " !");
   }

   public static boolean numerIThjesht(int n) {

      for (int i = 2; i <= n/2; i++) {
         if (n % i == 0)
            return false;
         // është gjetur pjesëtuesi, prandaj numri nuk është i thjeshtë
      }
      return true;
   }


   public static void main(String[] args) {

      int a = 20;
      if (numerIThjesht(a))
         System.out.println("Numri " + a + " ËSHTË i thjeshtë!");
      else
         System.out.println("Numri " + a + " NUK ËSHTË i thjeshtë!");
      intervalSHVP(5, 7, 9);
   }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Numri 20 NUK ËSHTË i thjeshtë!
Numri i kërkuar është: 315
```

Detyra ndodhet në CD, në dosjen *Teksti/src/KombinovaniZadaci/ZadaciMatematika*

Pyetje dhe detyra kontrolli

1. Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së kodit?

```
int shuma = 0;
for (int nr=20; nr<=50; nr++) {
    if(nr%7 == 0) break;
    shuma = shuma + nr;
}
System.out.println("shuma = ", shuma);
```

2. Çfarë do të paraqitet në daljen standarde mbas ekzekutimit të pjesës së kodit?

```
public int llogaria(int x){
    for(int y=5; y<=x; y++) {
    if(x%y == 0) return y;
}

public static void main(String[] args) {
    int z;
    z = llogaria(3) + llogaria(15);
    System.out.println("Vlera z=", z);
}
```

Puno vetë

1. Implemento të gjitha skemat algoritmike që i ke shënuar paraprakisht në seksionin 1.3.4.

VIII.

RRJEDHAT DHE SKEDARËT

Në këtë kapitull do të flasim mbi rrjedhat, skedarët dhe elementet e Javës që mundësojnë daljen dhe hyrjen e të dhënave. Rrjedhat shpeshherë përdoren gjatë shënimit të Java programit; duke filluar nga shënimet e thjeshta të të dhënave në monitor, pastaj për ruajtjen e të dhënave në një skedar, dhe te veprimet komplekse siç janë lidhjet e rrjedhave dhe filtrimi i të dhënave që nëpër to kalojnë.

Do të njoftoheni me klasat themelore për paraqitjen e rrjedhave hyrëse dhe rrjedhave dalëse të të dhënave:

- *InputStream*
- *OutputStream,*

dhe klasat e Javës që përdorën më shpesh për punën me rrjedha:

- *FileInputStream*
- *FileOutputStream*
- *PrintStream*
- *BufferedInputStream*
- *BufferedOutputStream*
- *BufferedReader*
- *InputStreamReader.*

Rrjedha e të dhënave

Unix është sistemi operativ për kompjuterët e zhvilluar gjatë viteve 1960 dhe 1970 nga të punësuarit e kompanisë AT&T Bell Labs.

Figura 8.1. Rrjedhat në jetën e përditshme

Rrjedha e të dhënave është një varg të dhënash që programi mund të pranojë nga mjedisi me qëllim që të përpunohet ose t'i dërgohet mjedisit mbas përpunimit. Gjatësia e rrjedhës është e matshme, por nuk është përcaktuar paraprakisht. Emri "rrjedha e të dhënave" është i lidhur me nocionin "lumi", që paraqet rrjedhën e vazhdueshme të ujit prej burimit deri në grykë.

Rrjedha e të dhënave si nocion, për herë të parë është paraqitur gjatë zhvillimit të sistemit operues UNIX, me qëllim që komunikimi i komplikuar me skedarë të bëhet më i thjeshtë. Ruajtja dhe përdorimi i të dhënave kërkon "kujdesin" mbi tipin dhe strukturën e tyre. Përdorimi i skedarëve, në aspektin e rrjedhës së të dhënave, nënkupton shënimin dhe leximin e vargjeve të bajtëve pavarësisht nga mënyra e interpretimit të të dhënave të shkruara dhe të lexuara. Pavarësia e interpretimit të të dhënave mundëson mënyrën unike të punës me rrjedha të të dhënave, pa marrë parasysh se a bëhet fjalë për komunikim në rrjetë, për program apo për një strukturë të dhënash. Konceptim i ngjashëm i punës me rrjedha është pranuar edhe në Javë.

Në figurën 8.1. është paraqitur një shembull i rrjedhave në jetën reale. Udhëtarët presin në rend, një nga një. Kur autobusi mbërrin, dyert hapen për praninë e udhëtarëve dhe udhëtarët hyjnë në autobus. Ekzistojnë rregullat sipas të cilave njerëzit sillen gjatë hyrjes në autobus - personat që janë të parët në rend hyjnë së pari në autobus e kështu me radhë. Kur të gjithë hyjnë në autobus, dyert mbyllen dhe autobusi niset. Në rastin kur plotësohen të gjitha vendet në autobus, ndërpritet hyrja e udhëtarëve në autobus dhe dyert mbyllen pavarësisht se a ka apo nuk ka më udhëtarë, që dëshirojnë të udhëtojnë deri në destinaconin e dëshiruar. Prandaj ata janë të obliguar të presin autobusin tjetër.

Rendi i udhëtarëve u përgjigjet tërësisht rrjedhave në Javë. Udhëtarët në këtë rast paraqesin të dhënat, kurse autobusi paraqet një operacion (veprim) funksional. Në rend për në autobus gjithmonë ndodhet një numër i fundmë udhëtarësh, që rritet ose zvogëlohet në mënyrë permanente, në varësi të shpejtësisë së transportimit të udhëtarëve dhe shpejtësisë së paraqitjes së udhëtarëve të rinj. Ekzistojnë edhe disa lidhje midis udhëtarëve, p.sh. nëna me fëmijë; nuk do të lejohet që nëna të shkojë me një autobus, kurse fëmijët me një autobus të tjetër. Pra, ata, shoferi i autobusit i konsideron si grup, që nënkupton se në autobus do të hyjë grupi i tërë ose asnjë anëtar i grupit.

Në varësi të sistemit operues, përfundimi i të dhënës në një rrjedhë zakonisht paraqitet me Ctrl + D (nën Unix-in) ose Ctrl + Y në PC kompjuterët.

Rrjedhat e të dhënave në Javë kanë të gjitha karakteristikat e përmendura. Në qoftë se të dhënat vijnë në rrjedhë më shpejt sesa mund të përpunohen, ato do të presin përpunimin përkatës. Për shkak të kësaj karakteristike, më shpesh është e parëndësishme kërkesa që secila e dhënë të përpunohet në momentin e paraqitjes së saj. Rrjedhat më shpesh përpunohen në një cikël, i cili do të ekzekutohet gjithnjë deri sa të ketë të dhëna në hyrje, d.m.th. deri sa të mbërrihet deri te fundi i hyrjes (për çfarë zakonisht ekziston një shenjë).

Në Javë ekzistojnë dy lloje rrjedhash: ***hyrëse*** dhe ***dalëse***. Rrjedhat hyrëse mundësojnë leximin e të dhënave nga një burim, kurse rrjedhat dalëse mundësojnë dërgimin e të dhënave mbas përpunimit në një vend tjetër. Prandaj shumë shpesh përdoren shprehjet "leximin nga rrjedha" dhe "shënimi në rrjedhë" që paraqesin rrjedhat hyrëse dhe dalëse.

Në punën me rrjedha, shpeshherë ndodh që programuesi të kërkojë një veprim që nuk mund të kryhet dhe në atë rast paraqitet devijimi nga procedura standarde, që mund të shkaktojë gabimin në kod. Është shumë me rëndësi që programuesi paraprakisht të parashohë të gjitha situatat

që mund të paraqiten në punën me rrjedhat dhe të përkufizojë hapat se çfarë të veprohet në disa raste ashtu që të mos vihet deri te gabimi.

Për shembull, programi e merr të dhënën nga rrjedha hyrëse, e përpunon dhe e dërgon në dalje. Çfarë ndodh me programin kur lexohen dhe përpunohen të gjitha të dhënat hyrëse? Programi akoma përpiqet të lexojë të dhënat hyrëse (të cilat nuk ekzistojnë) dhe kështu paraqitet gabimi. Pra, programuesi duhet të parashohë një situatë të tillë dhe me kod programues të përkufizojë çfarë të veprohet në atë rast.

Analiza e gabimeve gjatë përdorimit të rrjedhave hyrëse dhe dalëse të të dhënave është përpunuar në seksionin "Gabimet në programim".

8.1. Rrjedhat standarde të sistemit

Rrjedha standarde e të dhënave

Klasa *java.lang.System* përmban tri rrjedha kryesore, të cilat deri tani i kemi përmendur disa herë:

- rrjedha standarde hyrëse *in*,
- rrjedha standarde dalëse *out*,
- rrjedha standarde për shënimin e mesazhit mbi gabimin *err*.

Metodat për drejtimin e rrjedhave standarde hyrëse dhe dalëse të klasës *System* janë të përshtatshme gjatë shënimit të Java aplikacioneve të pavarura, kurse për shënimin në ekran në aplete përdoren metodat përkatëse grafike.

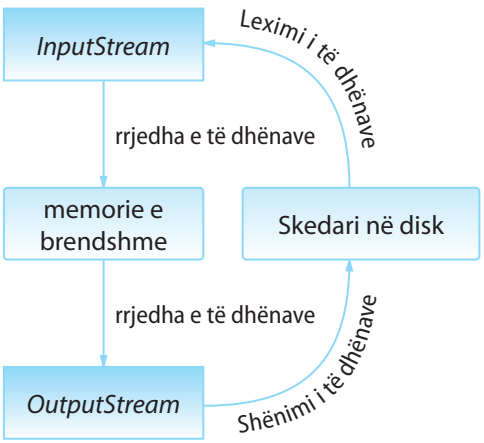
Daljen standarde (*System.out*) e kemi përdorur në të gjitha shembujt e deritashëm. Nga klasa *System.out* përdoren më shpesh dy metoda:

- metoda *print* i shtyp të dhënat duke mos kaluar në rreshtin e ri në fund të printimit,
- metoda *println* i shtyp të dhënat dhe e shton rreshtin e ri mbas printimit.

Dalja standarde (*System.in*) përdoret për hyrje të të dhënave nga tastieri. Nga klasa *System.in* më shpesh përdoret:

- metoda *read*, me të cilën lexohet një simbol nga tastieri, dhe kur mbërrihet në fund të rreshtit, kthehet vlera −1.

Rrjedha standarde për gabim (*System.err*) përdoret për paraqitjen e mesazhit mbi gabimin. Krahas pajisjes standarde dalëse (monitorit), është shumë e shpeshtë nevoja që mesazhet mbi gabimin të ruhen në një skedar. Në qoftë se ekziston kërkesa për analizën e punës së përdoruesve të caktuar, ose analizën e punës së programit, atëherë është e domosdoshme që të gjitha mesazhet mbi gabimet të ruhen në një skedar me qëllim që më vonë të bëhet analiza e tyre. Për një qëllim të tillë, rrjedha standarde për gabime drejtohet në printimin e mesazheve mbi gabimet në një skedar të caktuar.



Rrjedhat standarde të sistemit nuk janë të përshtatshme për zbatime komplekse, sepse përmbajnë vetëm metodat themelore. Prandaj ato as nuk përdoren gjatë punës me skedarë ose gjatë komunikimit nëpërmjet rrjetës. Shumica e rrjedhave të caktuara për hyrje dhe për dalje i takojnë paketës *java.io*. Kjo paketë përmban dy klasa themelore, ***InputStream*** dhe ***OutputStream***, që do t'i sqarojmë në vazhdim.

8.1.1. Klasa *InputStream*

Klasa InputStream

Klasa ***InputStream*** është klasa që përmban metodat me ndihmën e të cilave bëhet drejtimi me rrjedhën hyrëse të të dhënave. Në tabelën 8.1. janë paraqitur metodat themelore të klasës *InputStream*.

Tabela 8.1. Metodat e klasës *InputStream*

Metodat	Deklarimi	Domethënia
`read`	`public abstract int read() throws IOException`	Lexon një bajt të të dhënave, kurse në rastin se takon fundin e rrjedhës kthen –1.
`read`	`public int read(byte vargu[]) throws IOException`	Lexon të dhënat nga rrjedha hyrëse dhe i ruan ato në vargun e përkufizuar të bajtëve. Metoda kthen numrin e bajtëve të lexuar.
`read`	`public int read(byte vargu[], int pozicioni, int gjatesia) throws IOException`	I lexon të dhënat me *gjatësi* të përcaktuar nga rrjedha hyrëse, i vendos në *vargu*-n e përkufizuar, duke filluar nga *pozicioni* i cekur. Metoda kthen numrin e bajtëve të lexuara ose kthen –1, nëse ka mbërri në fund të rrjedhës para të dhënës së kërkuar.
`skip`	`public long skip(long numri) throws IOException`	Kapërcen *numrin* e dhënë të të dhënave hyrëse (bajtëve) nga rrjedha.
`available`	`public int available() throws IOException`	Kthen numrin e të dhënave që mund të lexohen nga rrjedha.
`close`	`public void close() throws IOException`	E mbyll rrjedhën hyrëse. Duhet të thirret në fund të punës me rrjedhën.
`mark`	`public void mark() throws IOException`	E shënon pozicionin momental në rrjedhë në mënyrë që të mundësohet kthimi i mëvonshëm në të.
`reset`	`public void reset() throws IOException`	E vendos pozicionin në rrjedhë në atë që është mbajtur mend mbas thirrjes së metodës *mark*.
`markSupported`	`public boolean markSupported() throws IOException`	Kthen vlerën logjike që cakton a mund të zbatohet kombinimi *mark*/*reset* mbi atë rrjedhë.

http://docs.oracle.com/javase/1.4.2/docs/api/java/io/InputStream.html

Gjatë ekzekutimit të metodave që përdoren në punë me rrjedhat e të dhënave, mund të arrihet deri te disa probleme. Fjala është mbi disa situata të veçanta të cilat mund të prishin ekzekutimin e programit, dhe mbi to do të ketë më shumë fjalë në kapitullin "Gabimet në program". Tani është me rëndësi të pranoni se në Javë përdoret termi ***përjashtimet*** (anglisht *Exceptions*), p.sh. kur është fjala për punë me rrjedha hyrëse-dalëse përdoret ***përjashtimet hyrëse-dalëse*** (anglisht *IOExceptions*).

Në program është e domosdoshme të parashihen mundësitë e paraqitjes së këtyre përjashtimeve dhe të përkufizohet çfarë të veprohet në rastin kur përjashtimi ndodh (i ashtuquajturi përpunimi i përjashtimit). Përpunimi i përjashtimit mundëson që programi të përfundojë punën e vet pa ndërprerje.

Në rast të paraqitjes së përjashtimit, programi ndërpret ekzekutimin e kodit standard të tij dhe fillon ekzekutimin e kodit të përpunimit të përjashtimit. Përpunimi standard i përjashtimit ofron informatën që ka ardhur deri te disa probleme të caktuara gjatë ekzekutimit të programit (gjë që përmendet mbas fjalës kyçe ***catch***). Mbas këtij mesazhi, eliminohet problemi i përmendur dhe përsëri niset ekzekutimi i programit.

8.1.1.1. Metoda *read*

Metoda ***read*** është metoda më e rëndësishme e klasës *InputStream* dhe mundëson leximin e të dhënave nga rrjedha hyrëse. Nga tabela 8.1. lexojmë se ekzistojnë tri metoda *read* që pranojnë parametra të ndryshëm dhe janë të dedikuar për përdorim në situata të ndryshme. Vetia e përbashkët e të gjitha metodave *read* është se gjatë procesit të leximit të të dhënave "bllokohet" programi gjatë pritjes të të dhënave hyrëse. Kjo domethënë se gjatë thirrjes së metodës *read* ndalohet ekzekutimi i programit deri sa të lexohen të dhënat nga rrjedha hyrëse.

Kjo ndërprerje e programit në praktikë nuk paraqet një problem. Fal mundësisë së Javës që në mënyrë paralele të ekzekutojë më shumë fije programit, programi mund të shkruhet ashtu që të ekzekutohet një pjesë e tjetër e kodit deri sa pritet leximi i të dhënave hyrëse.

Mënyra më e thjeshtë e përdorimit të metodës read është leximi i vetëm një bajti të dhëne, siç është paraqitur në shembullin e mposhtëm

```
// Mënyra e përdorimit të metodës read.
InputStream rrjedha = NjeRrjedheHyrese();

if (rrjedha.read() != 0) {
   System.out.println("Nuk është lexuar bajti i kërkuar i të dhënës!");
}
```

Mënyra më e shpeshtë e përdorimit të metodës *read* është leximi i vargut të bajtëve. Edhe pse vetëm në seksionin e ardhshëm do të njoftoheni me vargjet si tipa të të dhënave, le të japim një shembull përdorimi të metodës *read* në atë rast.

```
// Mënyra e përdorimit të metodës read.
InputStream rrjedha = NjeRrjedheHyrëse();
byte[] varguBajtesh = new byte[4096];

if (rrjedha.read(varguBajtesh) != varguBajtesh.length)
    System.out.println("Nuk janë lexuar të gjitha bajtët");
```

Në rastin e paraqitur, metoda *read* lexon të dhënat nga rrjedha dhe plotëson vargun *varguBajtesh*. Metoda kthen numrin e bajtëve të lexuar të të dhënave, ashtu që mund të verifikohet, nëse është plotësuar vargu i tërë.

Metoda *read* mund të përdoret edhe për leximin e pjesës së rrjedhës së të dhënave dhe plotësimit të vetëm një pjese të vargut. Në këtë rast është e domosdoshme të përdoret metoda *read* që pranon parametrat mbi vargun në të cilin do të ruhen të dhënat, mbi pozicionin nga i cili vargu fillon të plotësohet dhe numrin e bajtëve që duhet të lexohen, si në shembullin:

```
rrjedha.read(varguBajtesh, 1024, 500);
```

Metoda e thirrur në këtë mënyrë do të lexojë 500 bajt nga rrjedha dhe do t'i ruajë në vargun *varguBajtesh* nga vendi 1024 (pra do të plotësohet vargu nga pozicioni 1024 deri te pozicioni 1523). Një mënyrë e tillë e përdorimit të metodës *read* është praktike kur dëshirohet të krijohet vargu pjesërisht i plotësuar ose kur në varg dëshirohet të ruhen të dhënat që vijnë nga rrjedhat e ndryshme.

Në praktikë është rast i shpeshtë që leximit të dhënës nga rrjedha t'i shoqërohet një fije e veçantë e programit, kurse të dhënat e përftuara të përpunohen në fijet e tjera të programit.

8.1.1.2. Metoda *available*

Shpeshherë programuesi dëshiron të dijë se sa bajt mund të lexojë nga rrjedha. Për shembull, në rrjedhë ekziston një numër i caktuar i bajtëve që mund të lexohen menjëherë, kurse të dhënat e tjera duhet të priten. Duke thirrur metodën ***available*** është e mundshme të dihet se sa të dhëna mund të lexohen menjëherë nga rrjedha:

```
System.out.println("Menjëherë mund të lexohen " +
                   rrjedha.available() + "bajt!");
```

Megjithatë, ekzistojnë rrjedhat që nuk përkrahin përcaktimin e numrit të bajtëve të përgatitur për lexim, dhe në atë rast thirrja e kësaj metode kthen vlerën 0.

Rekomandim: Evitoni thirrjen e metodës *available*, vetëm kur jeni të sigurt se përdorni tipin e rrjedhës që e përkrah atë!

8.1.1.3. Metoda *skip*

Në rast se dëshirohet të kapërcehet një numër i caktuar të dhënash nga rrjedha dhe të vazhdohet me leximin e rrjedhës nga pozicioni i ri, mund të përdoret metoda ***skip***, si në shembullin

```
rrjedha.skip(500);
```

Me këtë komandë kapërcehen 500 bajt nga rrjedha e të dhënave, kurse me thirrjen vijuese të metodës *read* lexohet bajti i parë mbas vargut të bajtëve të kapërcyer.

8.1.1.4. Metoda *mark* dhe *reset*

Ekzistojnë situata kur programuesi dëshiron të lexojë përmbajtjen e një të dhëne në rrjedhë, mirëpo më vonë dëshiron të kthehet në pozicionin e njëjtë në rrjedhë. Disa rrjedha përkrahin mundësinë e etiketimit të pozicionit momental në rrjedhë, vazhdimin e leximit të të dhënave dhe kthimin e mëvonshëm në pozicionin e etiketuar. Për një gjë të tillë përdoret metoda ***mark*** dhe ***reset***. Duke marrë parasysh se rrjedhat vërtetë paraqesin një rrjedhë konstante e të dhënave, ato duhet të ruajnë të gjitha të dhënat që ndodhen ndërmjet pozicionit momental dhe pozicionit të ri, në mënyrë që sipas nevojës ato të dhëna të mund të lexohen. Prandaj, gjatë thirrjes së metodës *mark* duhet të jepet numri i bajtëve, i cili duhet të mbahet mend.

```
rrjedha.mark(1024);
```

Me këtë komanda mbahet në mend pozicioni momental dhe krijohet hapësira për ruajtjen e 1024 bajtëve shtesë nga rrjedha. Në qoftë se lexohen jo më shumë se 1024 bajt, është e mundshme të kthehet në vendin e shënjuar duke thirrur metodën *reset*.

```
rrjedha.reset();
```

Në qoftë se nga pozicioni i shënjuar janë lexuar më shumë se 1024 bajt të cekur, atëherë thirrja e metodës *reset* do të shkaktojë gabimin.

Rekomandim. Midis metodës *mark* dhe *reset*, gjithmonë lexoni për një bajt më pak se numri i cekur i bajtëve!

8.1.1.5. Metoda *close*

Kur përfundon puna me një rrjedhë, nevojitet që ajo të mbyllet. U propozojmë që këtë gjithmonë ta bëni vetë, edhe pse ekziston mundësia që kjo punë t'i lëshohet sistemit të Javës. Në rast se përdorni sistemin e mbylljes së Javës të rrjedhave të panevojshme, nuk do të keni mundësinë që të hapni përsëri rrjedhën, dhe një gabim i këtillë në program zbulohet shumë vështirë. Prandaj rekomandohet "programim i pastër", që nënkupton mbyllje të pavarur të të gjitha rrjedhave të përdoruara mbas përdorimit të tyre.

Që të mbyllni rrjedhën hyrëse, bëni thirrjen e metodës ***close*** në mënyrën e mëposhtme:

```
rrjedha.close();
```

Duke përdorur këtë komandë, jeni të sigurt se rrjedha është mbyllur dhe, sipas nevojës, mund ta hapni përsëri më vonë dhe nga ajo të lexoni të dhënat.

8.1.2. Klasa *OutputStream*

Klasa OutputStream

Klasa ***OutputStream*** është klasa që përcakton mënyrën me të cilën dërgohen të dhënat në rrjedhat dalëse. Në tabelën 8.2. janë paraqitur metodat themelore të klasës *OutputStream* që përdoren në rrjedhat dalëse.

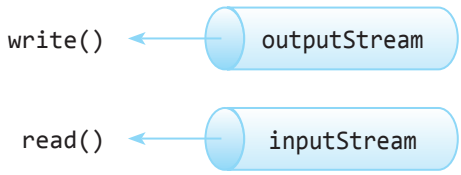
Tabela 8.2. Metodat themelore të klasës *OutputStream*

Metodat	Deklarimi	Domethënia
`write`	`public void write(int b) throws IOException`	Shënon një bajt në rrjedhën dalëse.
`write`	`public void write(byte vargu[]) throws IOException`	Shënon një *varg* bajt në rrjedhën dalëse.
`write`	`public void write(byte vargu[], int pozicioni, int numri) throws IOException`	Shënon *numrin* e bajtëve nga *vargu* në rrjedhën dalëse të të dhënave.
`close`	`public void close() throws IOException`	E mbyll rrjedhën. Metoda *close* duhet të thirret që të lirohen resurset që për rrjedhë kanë qenë përdorur.

8.1.2.1. Metoda *write*

Metoda më e rëndësishme për punën me rrjedha dalëse është ***write*** që mundëson shënimin e të dhënave në rrjedha dalëse. Metoda *write* ka tri trajta:

```
// Mënyra e përdorimit të metodës write.
OutputStream rrjedha = NjeRrjedheDalese();
byte b = merreBajtinVijues;
rrjedha.write(b);
```



Në mënyrë të ngjashme si në rastin e hyrjes së të dhënave, më shpesh takohet puna me vargje bajtësh, prandaj le të përmendim edhe ata shembuj. Shënimi i përmbajtjes së tërë në rrjedhën dalëse kryhet në mënyrën e mëposhtme:

```
// Mënyra e përdorimit të metodës write
OutputStream rrjedha = NjeRrjedheDalese();
byte[] varguBajtesh = new byte [1024];
// ... komanda me të cilën plotësohet vargu varguBajtesh
rrjedha.write (varguBajtesh);
```

Në qoftë se dëshirojmë të dërgojmë vetëm një pjesë të vargut në rrjedhën dalëse, përdoret trajta e tretë e metodës *write* që mundëson shënimin e vetëm një pjese të vargu. Në shembullin e mëposhtëm, kjo komandë në rrjedhën dalëse do të dërgojë 500 bajt nga vargu varguBajtësh, duke filluar nga vendi me indeks 1024.

```
// Mënyra e përdorimit të metodës write
OutputStream rrjedha = NjeRrjedheDalese();
byte[] varguBajtesh = new byte [4096];
// ... komanda me të cilën plotësohet vargu varguBajtesh m
rrjedha.write (varguBajtesh, 1024, 500 );
```

8.1.2.2. Metoda *close*

Si edhe te rrjedhat hyrëse, edhe rrjedhat dalëse duhet të mbyllen kur duk dëshirohet të punohet me to. Metoda *close* përdoret në mënyrën identike si te rrjedhat hyrëse ashtu edhe te rrjedhat dalëse:

```
rrjedha.close();
```

Pyetje dhe detyra kontrolli

1. Cilat janë rrjedhat standarde të klasës *java.lang.System*?
2. Cili është objektivi i klasës *InputStream*?
3. Cili është objektivi i klasës *OutputStream*?
4. Cila metodë lexon një bajt të dhënash, dhe në rastin kur takon fundin e rrjedhës kthen −1?
5. Me cilën metodë kapërcehet një numër i caktuar i të dhënave hyrëse (bajt) nga rrjedha?
6. Me cilën metodë kontrollohet se sa të dhëna mund të lexohen nga rrjedha?
7. Me cilën metodë shënohet një bajt në rrjedhën dalëse?
8. Çfarë veprimi kryen linja e mëposhtme e kodit:

```
if (rrjedha.read(varguBajtesh) != varguBajtesh.length) {...}
```

8.2. Klasat për punën me skedarë

Në këtë kapitull janë paraqitur klasat themelore për punë me skedarë. Klasat *FileInputStream* dhe *FileOutputStream* janë dedikuar për leximin e të dhënave nga skedari dhe shënimin e të dhënave në skedar. Ato trashëgojnë klasat themelore *InputStream*, gjegjësisht *OutputStream*, që nënkupton se përkrahin metodat dhe vetitë që i kemi përshkruar në kapitullin paraprak, por përmbajnë edhe metodat që janë dedikuar zbatimit konkret dhe lehtësimit të punës me disa tipa të të dhënave në rrjedhë.

8.2.1. Klasa *FileInputStream*

Klasa FileInputStream

Leximi i të dhënave nga skedari realizohet duke krijuar rrjedhën hyrëse të të dhënave mbi skedarin si burim të dhënash. Rrjedha hyrëse e skedarit ekzistues krijohet nëpërmjet objektit të klasës ***FileInputStream***. Në atë rast nevojitet të thirret konstruktori i klasës dhe të jepet emri i skedarit nga i cili do të lexohen të dhënat. Emri i skedarit mund të përmendet pavarësisht nga sistemi operues në të cilin programi ekzekutohet, sepse Java në mënyrë automatike do të adaptojë emrin e skedarit platformës në të cilën programi është nisur. Në tabelën 8.3. janë paraqitur konstruktorët që përdoren më shpesh të klasës *FileInputStream*.

Më shumë informata mbi klasën mund të gjeni në sajtin: http://docs.oracle.com/javase/7/docs/index.html

Tabela 8.3. Konstruktorët e klasës *FileInputStream* që përdoren më shpesh

Konstruktori	Domethënia
`FileInputStream(File skedari)`	Krijon rrjedhën hyrëse të të dhënave duke vendosur lidhjen me skedarin e përmendur.
`FileInputStream(String emri)`	Krijon rrjedhën hyrëse të të dhënave duke vendosur lidhjen me skedarin e cila është emërtuar me emrin e dhënë.

Në tabelën 8.4. janë paraqitur metodat që përdoren më shpesh të klasës *FileInputStream*.

Tabela 8.4. Metodat e klasës *FileInputStream* që përdoren më shpesh

Metodat	Deklarimi	Domethënia
read	`public int read()` `throws IOException`	Lexon një bajt të të dhënave ose – 1 nëse takon fundin e rrjedhës.
read	`public int read(byte vargu[])` `throws IOException`	Lexon të dhënat në një varg bajtësh. Parametri vargu paraqet *vargun* e bajtëve që do të lexohet. Metoda kthen numrin e bajtëve të lexuar.
read	`public in read(byte vargu[], int pozicioni, int gjatesia)` `throws IOException`	I lexon të dhënat në vargun e bajtëve *vargu*[] duke filluar nga *pozicioni* i përmendur me *gjatësi* të përmendur. Metoda kthen numrin e bajtëve të lexuar ose kthen -1, në qoftë se ka takuar fundin e rrjedhës para numrit të dëshiruar të të dhënave.
skip	`public long skip(long numri)` `throws IOException`	Kapërcen *numrin* e dhënë të të dhënave hyrëse (bajtëve) nga rrjedha.
available	`public int available()` `throws IOException`	Kthen numrin e të dhënave që nga rrjedha mund të lexohen.
close	`public void close()` `throws IOException`	E mbyll rrjedhën hyrëse. Kjo metodë duhet të thirret në fund të punës me rrjedhë.
finalize	`Protected void finalize()` `throws IOException`	Siguron që metoda close ekzekutohet vetëm kur nuk do të ketë nevojë për përdorimin e rrjedhës.

Metodat read nga klasa *FileInputStream* lexojnë të dhënat binare, prandaj këto metoda nuk mund të përdoren për leximin e karaktereve nga skedari. Për një gjë të tillë përdoren klasat e tjera siç është *FileReader*.

Në shembullin e mëposhtëm do të paraqesim mënyrën e përdorimit të klasës *FileInputStream*.

Shembulli 1. Duke përdorur metodat e klasës *FileInputStream* të krijohet rrjedha hyrëse e të dhënave për leximin e përmbajtjes së skedarit tekstual *LeximiStrigut.txt*, që ndodhet në direktoriumin *C://Shembull* (d.m.th. rruga deri te skedari është *C://Shembull/LeximiStrigut.txt*). Në pajisjen standarde dalëse (monitor) të shënohet përmbajtja e lexuar.

```
import java.io.*;

public class ShembulliFileInputStream {

   public static void main(String[] args) {

      // krijohet objekti i klasës file
      File skedari  = new File("C://Shembull/LeximiStringut.txt");
      int eDhena;
      StringBuffer permbajtja = new StringBuffer("");
      FileInputStream rrjedhahyrese = null;

      try {

         /*
            Krijohet rrjedha hyrëse nëpërmjet të cilës formohet lidhja me skedarin. Në qoftë se
            skedari i përmendur nuk ekziston në sistem, shkaktohet gabimi "skedari nuk ekziston".
          */
         rrjedhahyrese  = new FileInputStream(skedari);

         /*
            Leximi i të dhënës nga rrjedha e krijuar realizohet në mënyrën e mëposhtme:
            1. ndryshorja permbajtja paraqet stringun e krijuar nga simbolet e lexuara nga skedari
            2. me metodën read( ) lexohet një bajt të dhënash nga skedari
            3. metoda read( ) kthen vlerën – 1 në qoftë se të gjitha të dhënat janë të lexuara,
               që përdoret si kusht në ciklin while
            4. bajti i lexuar i të dhënave zbërthehet në karakterin simboli dhe shtohet në
               stringun përmbajtja
          */

         while ((eDhena  = rrjedhahyrese.read()) != -1) {
            char simboli  = (char) eDhena;
            permbajtja.append(simboli);
         }

         rrjedhahyrese.close(); // rrjedha hyrëse mbyllet duke thirrur (zbatuar) metodën close()
      }

      /* Përpunimi i gabimit në qoftë se nuk ekziston skedari */

      catch (FileNotFoundException e) {
         System.out.println("Skedari " + skedari.getAbsolutePath()
               + " nuk mund të gjindet në sistemin skedar!");
      }

      /* Përpunimi i gabimit të paraqitur gjatë leximit të të dhënave nga skedarët */
      catch (IOException ioe) {
         System.out
               .println("Është shfaqur gabimi gjatë leximit të të dhënave nga skedari! "
                     + ioe);

      }
```

```
        /* Printimi i përmbajtjeve të lexuara të skedarit */
        System.out.println("Përmbajtja e skedarit : ");
        System.out.println(permbajtja);
    }
}
```

Projekti i tërë ndodhet në dosjen *Teksti/src/RrjedhatSkedaret/FileInputStream* në CD.

Për nisjen e programit, krijoni dosjen *Shembulli* në *C:* dhe në të skedarin *LeximiStringut.txt* me përmbajtjen e mëposhtme:

```
Ky tekst ndodhet në skedarin e kërkuar, është lexuar nëpërmjet rrjedhës
hyrëse të të dhënave dhe është paraqitur në monitor!
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Përmbajtja e skedarit:
Ky tekst ndodhet në skedarin e kërkuar, është lexuar nëpërmjet rrjedhës
hyrëse të të dhënave dhe është paraqitur në monitor!
```

8.2.2. Klasa *FileOutputStream*

Klasa FileOutputStream

Klasa ***FileOutputStream*** mundëson krijimin e rrjedhës dalëse të të dhënave dhe regjistrimin e të dhënave në skedar. Që të mundësohet regjistrimi i të dhënave nga rrjedha dalëse në skedar, është e domosdoshme të përdoret një prej konstruktorëve të klasës *FileOutputStream*. Në atë rast është e nevojshme të ceket edhe emri i skedarit në të cilin do të kryhet regjistrimi. Emri i skedarit mund të ceket pavarësisht nga sistemi operues, sepse Java në mënyrë automatike do të adaptojë emrin e skedarit platformës në të cilën programi është nisur. Në tabelën 8.5. janë paraqitur konstruktorët e klasës *FileOutputStream* që përdoren më shpesh.

Tabela 8.5. Konstruktorët e klasës *FileOutputStream* që përdoren më shpesh

Konstruktori	Domethënia
`FileOutputStream(File skedari)`	Krijon rrjedhën dalëse të të dhënave për regjistrimin në skedar që është paraqitur me objektin e skedarit të klasës *File*.
`FileOutputStream(File skedari, boolean tShtohet)`	Krijon rrjedhën dalëse të të dhënave për regjistrimin në skedar. Nëse argumenti i dytë *tShtohet* ka vlerën `true`, atëherë të dhënat regjistrohen në fund të skedarit ekzistues.
`FileOutputStream(String naziv)`	Krijon rrjedhën dalëse të të dhënave për regjistrimin në skedar që është e paraqitur me stringun *emri*..
`FileOutputStream(String emri, boolean shto)`	Krijon rrjedhën dalëse të të dhënave për regjistrimin në skedar *emri* i të cilit është paraqitur me një string, kurse në varësi të vlerës së parametrit të dytë (`true`), të dhënat e reja i shton në fund të skedarit.

Në tabelën 8.6. janë paraqitur metodat e klasës *FileOutputStream* që përdoren më shpesh.

Tabela 8.6. Metodat e klasës *FileOutputStream*

Metodat	Deklarimi	Domethënia
write	`public void write(int eDhena) throws IOException`	Regjistron një bajt (*eDhena*) në rrjedhën dalëse.
write	`public void write(byte vargu[]) throws IOException`	Regjistron vargun e bajtëve (*vargu*) në rrjedhën dalëse.
write	`public void write(byte vargu[], int pozicioni, int gjatesia) throws IOException`	Regjistron të dhënat e përkufizuara me gjatësinë *gjatesia* nga vargu i bajtëve *vargu*, nga *pozicioni* i përmendur në rrjedhën dalëse.
close	`public void close() throws IOException`	E mbyll rrjedhën dalëse të të dhënave.
finalize	`protected void finalize() throws IOException`	Siguron se metoda close është ekzekutuar mbas që më nuk ekziston nevoja për përdorimin e rrjedhës.

Më shumë informacione mbi klasën mund të gjeni në sajtin: http://docs.oracle.com/javase/7/docs/index.html

Metoda *write* nga klasa *FileOutputStream* i regjistron të dhënat binare në skedar. Në shembullin e mëposhtëm do të paraqesim mënyrën e përdorimit të klasës *FileOutputStream*.

Shembulli 2. Duke përdorur metodat *write* të klasës *FileOutputStream* në fund të skedarit *ShtimiStringut.txt* që ndodhet në direktorium *C://Shembulli*, shto fjalinë: "Kjo fjali do të shtohet në skedarin e përmendur".

```
import java.io.*;

public class ShembulliFileOutputStream {
   public static void main(String[] args) {

      String skedari = "C://Shembulli/ShtimiStringut.txt";

      try {
         /* Që përmbajtja të shtohet në një skedar, nevojitet:
            1. të krijohet rrjedha dalëse e të dhënave që do të vendosi lidhjen me skedarin
            2. për mundësimin e shtimit të përmbajtjeve në përmbajtje ekzistuese në skedar,
               përdoret konstruktori FileOutputStream(String skedari, Boolean shto), ku argumenti
               i parë paraqet emrin e skedarit, kurse argumenti i dytë ka vlerën true. Në qoftë
               se për vlerën e argumentit të dytë vendoset false, përmbajtja ekzistuese e skedarit
               do të fshihet, dhe do të regjistrohet përmbajtja e re që i dërgohet metodës write.
          */

         FileOutputStream rrjedhadalese  = new FileOutputStream(skedari, true);

         String fjalia  = "Kjo fjali do të shtohet në skedarin e përmendur!";

         /* String fjalia zbërthehet në një varg bajtësh dhe dërgohet
            në rrjedhën dalëse (skedar) */
         rrjedhadalese.write(fjalia.getBytes());

         // Rrjedha dalëse e të dhënave mbyllet.
         rrjedhadalese.close();
      }
```

```
        // Përpunimi i gabimit në qoftë se nuk ekziston skedari
        catch (FileNotFoundException ex) {
            System.out.println("Skedari i përmendur nuk u gjet : " + ex);
        }

        // Përpunimet e gabimeve të shfaqura gjatë regjistrimit të të dhënave
        catch (IOException ioe) {
            System.out.println("Është paraqitur gabimi gjatë regjistrimit të të dhënave: " + ioe);
        }
    }
}
```

Për nisjen e programit, në direktoriumin *Shembulli* në *C:* krijoni skedarin *ShtimiStringut.txt* me përmbajtjen e mëposhtme:

```
Mësojmë programimin në Javë: puna me rrjedhat hyrëse dhe dalëse!
```

Mbas nisjes së programit, përmbajtja e skedarit *ShtimiStringut.txt* do të jetë:

```
Mësojmë programimin në Javë: puna me rrjedhat hyrëse dhe dalëse!
Kjo fjali do të shtohet në skedarin e përmendur!
```

Projekti i tërë ndodhet në dosjen *Teksti/src/RrjedhatSkedaret/FileOutputStream* në CD.

8.2.3. Klasa *Scanner*

Klasa Scanner

Klasa ***Scanner*** mundëson leximin e të dhënave nga një rrjedhë hyrëse e të dhënave ashtu që lexon toke-nin (varg karakteresh deri te dilimetri, d.m.th. deri te shenja speciale) dhe e zbërthen në tipin përkatës të të dhënave (*string*, `byte`, `int`, `double`, `boolean` ose një tip tjetër). Delimiteri standard është " bosh " ose "space", dhe në varësi nga nevoja mund të përkufizohet një delimetër tjetër, për shembull (`;` ose `:` ose një string tjetër). Në këtë mënyrë është lehtësuar dukshëm leximi i të dhënave nga rrjedha hyrëse, sepse automatikisht kryhet edhe interpretimi i të dhënave të lexuara dhe konvertimi i tyre në njërin prej tipave standardë të të dhënave, duke bërë thirrjen e metodës përkatëse të klasës *Scanner*.

Klasa *Scanner* mund të përdoret për leximin e të dhënave nga tastiera (rrjedha standarde hyrëse *System.in*) ose për leximin e të dhënave nga ndonjë skedar. Varësisht se nga cila rrjedhë hyrëse dëshirohet të lexohen të dhënat, përdoren konstruktorët përkatës të klasës *Scanner* që janë përmendur në tabelën 8.7.

Tabela 8.7. Konstruktorët e klasës *Scanner* që përdoren më shpesh

Konstruktori	Domethënia
`public Scanner(InputStream rrjedhaHyrëse)`	Krijon skanerin e ri që lexon të dhënat nga një tip hyrës i të dhënave.
`public Scanner(File skedari)`	Krijon skanerin e ri që i lexon të dhënat nga skedari.
`public Scanner(String burimiTëDhënave)`	Krijon skanerin e ri që i lexon të dhënat nga një string.

Më shumë informacione mbi klasën ndodhen në sajtin: http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html

Klasa *Scanner* ka metoda të shumta për leximin e tipave të të dhënave të caktuara nga rrjedha hyrëse. Krahas metodës për leximin e të dhënave, kjo klasë ka edhe metodat me të cilat verifikohet nëse në rrjedhën hyrëse ndodhen të dhënat e tipit të caktuar. Kjo metodë e lehtëson dukshëm kontrollin e procesit të leximit të të dhënave nga një rrjedhë hyrëse.

Në tabelën 8.8. janë paraqitur metodat e klasës *Scanner* që përdoren më shpesh.

Tabela 8.8. Metodat e klasës Scanner që përdoren më *shpesh*

Metodat	Deklarimi	Domethënia
nextLine	`public String nextLine()`	Nga rrjedha hyrës e të dhënave lexon vargun e karaktereve deri te rreshti i ri dhe i zbërthen në string.
hasNextLine	`public boolean hasNextLine()`	Verifikon nëse në rrjedhën hyrëse ndodhen të dhënat apo jo. Kthen vlerën true në qoftë se në rrjedhën hyrëse ndodhen të dhënat ose false, nëse rrjedha hyrëse është e zbrazët.
nextBoolean	`public boolean nextBoolean()`	E lexon tokenin vijues dhe e zbërthen në vlerë boolean.
hasNextBoolean	`public boolean hasNextBoolean()`	Kontrollon nëse tokeni vijues në rrjedhën hyrëse ka vlerën boolean apo jo.
nextDouble	`public double nextDouble()`	E lexon tokenin vijues dhe e zbërthen në vlerë dooble.
hasNextDouble	`public boolean hasNextDouble()`	Kontrollon nëse tokeni vijues në rrjedhën hyrëse ka vlerën dooble.
nextInt	`public int nextInt()`	E lexon tokenin vijues dhe e zbërthen në vlerë int.
hasNextInt	`public boolean hasNextInt()`	Kontrollon nëse tokeni vijues në rrjedhën hyrëse ka vlerën int.
nextByte	`public byte nextByte()`	E lexon tokenin vijues dhe e zbërthen në vlerë byte.
hasNextByte	`public boolean hasNextByte()`	Kontrollon nëse tokeni vijues në rrjedhën hyrëse ka vlerën byte.
useDelimiter	`public Scanner useDelimiter(String pattern)`	Krijon dilimetrin e ri që përdoret gjatë skanimit të tokenit nga rrjedha hyrëse e të dhënave.
close	`public void close()`	E mbyll skanerin.

Në shembullin e ardhshëm është paraqitur mënyra e përdorimit të klasës *Scanner* për leximin e të dhënave nga hyrja standarde (tastiera). Për krijimin e skanerit që do të lexojë të dhënat nga tastiera, përdoret konstruktori, të cilit si parametër i dërgohet hyrja standarde (*System.in*). Për leximin e numrit të plotë përdoret metoda *nextInt*().

Shembulli 3. Të shkruhet programi për llogaritjen e shumës së dy numrave të futur nëpërmjet tastierës.

```
import java.util.Scanner;

public class Skaner {
   public static void main(String[] args) {

      Scanner skaner = new Scanner(System.in);
      System.out.println("Shkruani dy numra të plotë");
      int a = skaner.nextInt();
      int b = skaner.nextInt();
      int shuma = a + b;
      System.out.println("Shuma e numrave " + a + " dhe " + b + " është " + shuma);
   }
}
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Shkruani dy numra të plotë
5
15
Shuma e numrave 5 dhe 15 është 20
```

Programi i tërë ndodhet në dosjen *Teksti/src/RrjedhatSkedaret/Scanner* në CD.

Në shembullin vijues është paraqitur mënyra e përdorimit të klasës Scanner për leximin e të dhënave nga një skedar.

Shembulli 4. Të shkruhet programi për leximin gradual të tekstit rresht nga një rresht dhe paraqitjen e përmbajtjes së lexuar në daljen standarde (ekran).

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class ShembulliScannerReadFile {
   public static void main(String[] args) {
      try {
         File skedari = new File("C://Shembulli/test.txt");
         Scanner skaneri1 = new Scanner(skedari);
         System.out.println("Përmbajtja e skedarit:");
         while (skaneri1.hasNextLine()) {
            String linjaTekstit = skaneri1.nextLine();
            System.out.println(linjaTekstit);
         }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      }
   }
}
```

Programi i tërë ndodhet në dosjen *Teksti/src/RrjedhatSkedaret/Scanner* në CD.

Për nisjen e programit në dosjen e krijuar *Shembulli* në *C*: krijoni skedarin (fajllin) *Test.txt* me përmbajtjen e mëposhtme:

```
Mësojmë programimin në Javë: punë me rrjedhat hyrëse-dalëse!
Mësojmë si të futen të dhënat nga tastiera dhe nga skedari...
```

Kur programi niset, në daljen standarde do të paraqitet:

```
Përmbajtja e skedarit:
Mësojmë programimin në Javë: punë me rrjedhat hyrëse-dalëse!
Mësojmë si të futen të dhënat nga tastiera dhe nga skedari...
```

8.2.4. Klasa *PrintStream*

Klasa PrintStream

Deri tani kemi përdorur shumë herë rrjedhën ***PrintStream***, dhe një gjë të tillë nuk e kemi përmendur në mënyrë eksplicite. Çdoherë kur kemi përdorur metodën *print* ose *println* nga klasa *System.out*, kemi përdorur rrjedhën *PrintStream*. Ndryshoret *in* dhe *out* janë atribute të klasës *PrintStream*. Klasa *PrintStream* paraqet rrjedhën dalëse të të dhënave që përdoren më shpesh, sepse mund të printojë (shënojë) një varg tërësisht të ndryshëm të tipave të të dhënave. Metodat e saj *print* dhe *println*, mund të pranojnë parametrat e ndryshëm dhe në mënyrë të suksesshme t'i shënojnë në monitor apo në një pajisje tjetër dalëse.

Tabela 8.9. Konstruktorët e klasës *PrintStream* që përdoren më shpesh

Konstruktori	Domethënia
`public PrintStream(File skedari) throws FileNotFoundException`	Krijon një instancë të re të klasës *PrintStream* me përdorimin e datotekës.
`public PrintStream(String emërtimi) throws FileNotFoundException`	Krijon një instancë të re të klasës *PrintStream* me përdorimin e emërtimit të datotekës që është paraqitur me string.
`public PrintStream(OutputStream rrjedhaDalëse)`	Krijon një instancë të re të klasës *PrintStream* me përdorimin e rrjedhës dalëse.

Në tabelën 8.10. janë treguar metodat e klasës *PrintStream* që përdoren më shpesh.

Tabela 8.10. Metodat e klasës *PrintStream* që përdoren më shpesh

Metodat	Deklarimi	Domethënia
write	`public void write(int eDhëna)`	Rregjistron *të dhënën* (bajtin) e dhënë në rrjedhën dalëse.
write	`public void write(byte[] bafer, int pozicioni, int gjatesia)`	I merr të dhënat nga baferi, me *gjatësi* të përkufizuar nga *pozicioni* i përcaktuar dhe i regjistron në rrjedhën dalëse.
print ili **println**	`public void print(boolean b)`	Në rrjedhën dalëse regjistron vlerën `boolean` *b*.
print ili **println**	`public void print(char c)`	Në rrjedhën dalëse regjistron vlerën `char` c.
print ili **println**	`public void print(int i)`	Në rrjedhën dalëse regjistron vlerën `int` i.
print ili **println**	`public void print(long l)`	Në rrjedhën dalëse regjistron vlerën `long` l.