**Hyrje në SHK1**

**Cili është qëllimi i kësaj lënde ?**

Do të jetë fillimi i një edukimi tuaj si Programer (Inxhinier të softuerit)

**Cila është rëndësia e kësaj lënde ?**

Nga kjo lëndë do të fitoni dijeninë si të përdorni gjuhën programuese Java për të ndërtuar softuer

**Llojet softuerëve (Programeve)**

* Kompjuterët janë pajisje të fuqishme harduerike që nuk mund të bëjnë asgjë pa pjesën softuerike
* Në përgjithësi ekzistojnë tri lloje të softuerëve:
  + Sistemet operative
  + Programet aplikative
  + Veglat për zhvillim të softuerit (SDK)
* Si zhvillues të softuerit mund të ndodhë të shkruani njërën apo që të tri llojet e softuerëve të lartpërmendur

**Programet e Sistemeve Operative (O/S)**

* O/S programet e kontrollojnë harduerin dhe i mundësojnë programeve aplikative të ekzekutohen
* O/S kryesisht janë të ndërtuara që të ekzekutohen në platforma të veçanta harduerike, p.sh. në x86 apo MAC
* Kryesisht këto janë programet më komplekse dhe më të vështira për t’u ndërtuar dhe testuar
* P.sh. M/S Dos, Windows, Unix, Linux, etj

**Programet Aplikative**

* Këto programe kryejnë punë të rëndësishme për shfrytëzuesit
* Aplikacionet kryesisht janë të ndërtuara që të ekzekutohen në sisteme operative të veçanta dhe platforma të veçanta harduerike
* Shfrytëzuesit kryesisht duhet të ofrojnë informata të hollësishme për funksionalitetin që programi duhet ta ketë, në mënyrë që programerët të ndërtojnë aplikacionin specifik që përmbushë kërkesat e shfrytëzuesit
* P.sh. Word, PowerPoint, Netscape, etj..

**Veglat për zhvillim të softuerëve**

Veglat për zhvillim të softuerëve (ang. Software Development Kit - SDK) janë programe specifike që i mundësojnë zhvilluesve të programeve të shkruajnë programe kompjuterike dhe t’i testojnë ato

Zhvilluesit me përvojë kryesisht përdorin “Integrated Development Environment – IDE”, por mund të përdoren edhe editorë të thjeshtë për të kompajluar/ekzekutuar (interpretuar) programin

P.sh.

* + IDE: Dr Java IDE, Netbeans IDE, Eclipse, etj
  + Editorë: PFE, Notepad++, etj

**Llojet e “User Interface”-ëve**

* Janë dy lloje dominuese të “user interface”-ëve:
  + Command Line Interface (CLI)
  + Graphical User Interface (GUI)
* Si zhvillues të softuerit ju duhet të jeni në gjendje të shkruani programe për dy llojet e “user interface”-ëve

**Llojet e “User Interface”-ëve**

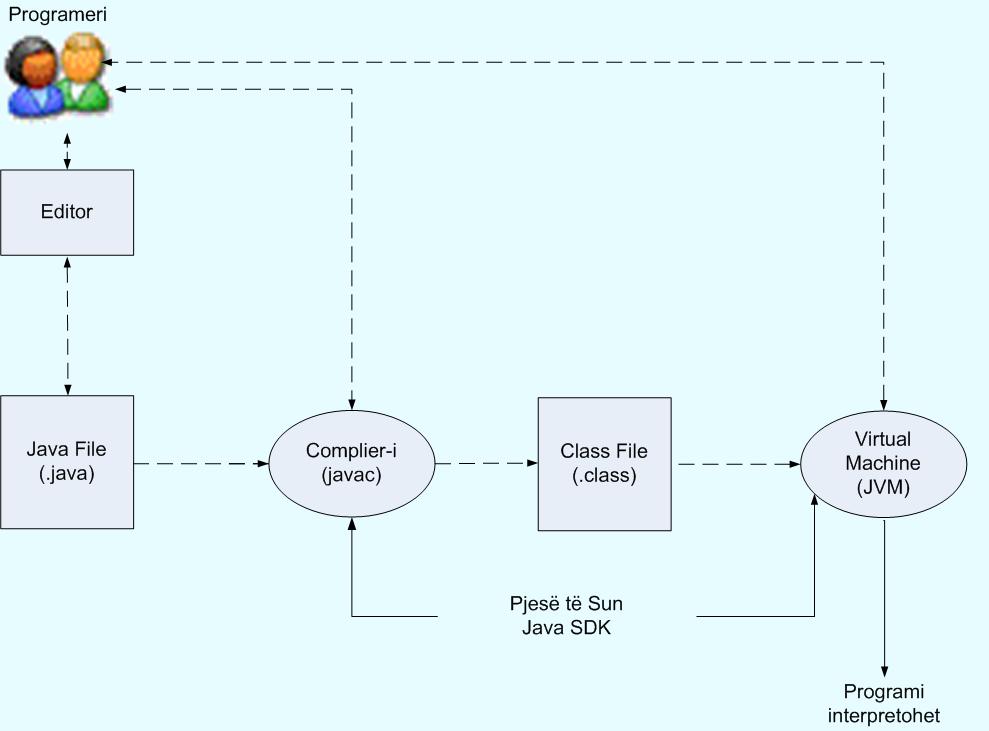
* **Command Line Interface (CLI)**
  + Në monitor paraqitet kërkesa që shfrytëzuesi të fut ndonjë të dhënë.
  + Shfrytëzuesi i fut të dhënat e kërkuara, ndërsa programi e proceson inputin e shfrytëzuesit.
  + Ky është një lloj i vjetër i “user interface”-ëve ende i përdorur në disa aplikacione dhe sisteme operative
  + Nuk është i përshtatshëm për shfrytëzim (ang. not User Friendly)
  + P.sh. Në DOS C:\> type test.txt (shfaq përmbajtjen e dokumentit)

**Llojet e “User Interface”-ëve**

* **Graphical User Interface (GUI)**
  + Në monitor paraqitet dritarja me tekst dhe komponente grafike duke i ofruar shfrytëzuesit opsione të ndryshme
  + Shfrytëzuesi përdor “mouse”-in dhe tastierën, të bënë zgjidhje nga opsionet e ofruara si dhe për të shkruar tekst
  + Tani përdoret më shumë për shkak se komponentët për procesim kompjuterik kushtojnë tani më lirë
  + Është më i përshtatshëm për shfrytëzim (ang. User Friendly)
  + P.sh. Microsoft Windows, Microsoft Office, etj.

**Veglat për zhvillim të softuerëve**

* Përdorimi i editorit të thjesht dhe Sun JDK (Java Development Kit)



**Përdorimi i editorit të thjeshtë dhe Sun JDK**

* Shkruhet Java kodi në ndonjë editor. File-i ruhet me të njëjtin emër si emri i klasës dhe me extension .*java*. P.sh nëse e kemi klasën HelloWorld atëherë ruhet në file-in me emër HelloWorld.java
* Në DOS:

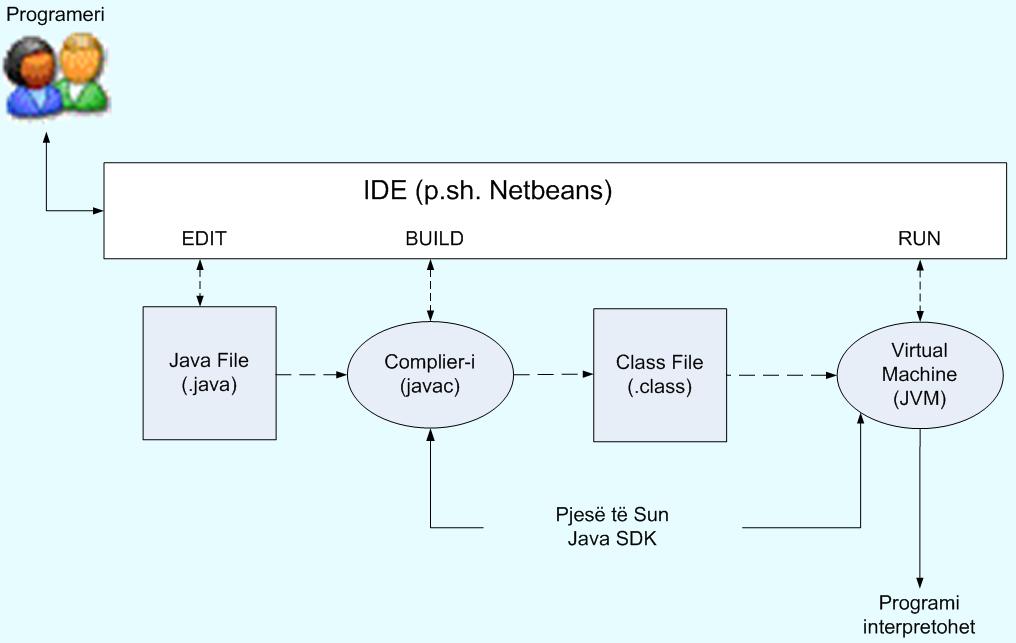
- C:\>javac HelloWorld.java (krijohet një file-i i ri me prapashtesë *.class, në këtë rast HelloWorld.class*)

- C:\>java HelloWorld (interpretohet programi) dhe dërgohet output-i në konsolë

* + C:\>exit (del prej DOS-it)

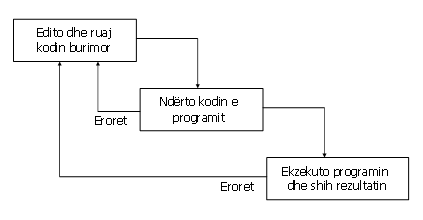
**Veglat për zhvillim të softuerëve**

* Përdorimi i IDE (Integrated Development Environment)



**Hapat e zhvillimit të softuerit**

* Modeli klasik “Waterfall”



**Eroret**

* Gjatë ndërtimit të një programi mund të paraqiten 3 lloje të eroreve:
  + Compile-time error – që paraqet gabim në sintaksën e programit dhe që vërehet gjatë kohës së kompilimit dhe nuk krijohet file-i me extension *.class*
  + Run-time error – që paraqitet gjatë ekzekutimit/interpretimit të programit, p.sh. Pjesëtimi me zero e detyron programin të ndërpritet në mënyrë jonormale
  + Logical error – që do të thotë që programi ekzekutohet normalisht mirëpo rezultati nuk është ai i prituri

**Gjuha programuese Java dhe zhvillimi i softuerit**

* + Gjuhët programuese
  + Struktura e programit në Java
  + Zgjidhja e problemeve
  + Programimi i orientuar/bazuar në objekte

**Gjuhët programuese**

* + Gjuhët programuese përdoren nga programerët për të shkruar programe kompjuterike
  + Ndonjë gjuhë programuese është më e mirë për ndonjë lloj të programit dhe për ndonjë lloj të “user interface”-it, se tjetra
  + Me siguri se keni dëgjuar për gjuhë të ndryshme programues si: Cobol, Basic, C, C++, Java, C#, etj

**Gjuhët programuese**

* + Një gjuhë programuese specifikon fjalët dhe simbolet që mund të përdoren për të shkruar programe kompjuterike
  + Një gjuhë programuese përdorë rregulla të veçanta që diktojnë se si fjalët dhe simbolet mund të kombinohen që të formohen shprehje të vlefshme programuese
  + Një gjuhë programuese posedon edhe sintaksën edhe semantikën e vet

**Sintaksa dhe semantika**

* + Rregullat sintaksore tregojnë si mund të kombinohen fjalët e rezervuara, simbolet dhe identifikuesit që të krijohet një program i vlefshëm.
  + Semantika tregon kuptimin e një shprehjeje të vlefshme (çka është qëllimi dhe roli i asaj shprehjeje në atë program)
  + Një program që është korrekt sa i përket sintaksës nuk do të thotë se është korrekt edhe për nga semantika

Një program gjithmonë bënë atë që i themi të bënë, e jo çka mendojmë t’i themi të bënë

**Nivelet e gjuhëve programuese**

Ekzistojnë disa nivele të gjuhëve programuese:

Machine Language

Assembly Language

Fourth-generation language

High-level Language

Çdo lloj i CPU (Central Processing Unit) e ka gjuhën e vet specifike të makinës (ang. Machine Language). Përbënë instruksionet dhe të dhënat të cilat ekzekutohen direkt nga CPU.

Gjuhët e tjera kanë rrjedhë nga nevoja që programerët më lehtë të shkruajnë dhe lexojnë programet kompjuterike.

**Kompajlimi i kodit në Java**

* + Kompajleri i Java-së bënë përkthimin e Java kodit në një reprezentim specifik të njohur si Java bytecode të ruajtur në një file me prapashtesën ***.class***
  + Java bytecode nuk është gjuhë e makinës për asnjë CPU specifike
  + Një softuer tjetër i njohur si interpreter-i (në rastin tonë JVM - Java Virtual Machine) e interpreton bytecode-in
  + Java njihet si gjuhë neutrale sa i përket arkitekturës së kompjuterëve
  + Java nuk lidhet me ndonjë makinë specifike
  + JVM mund të implementohet në çdo lloj të makinës

**Struktura e programit në Java**

* + Në gjuhën programuese Java:
    - Programi ndërtohet nga një ose më shumë klasë
    - Një klasë ka një apo më shumë atribute
    - Një klasë ka një apo më shumë metoda
    - Metoda përmban shprehje të programit (ang. program statements)
  + Këto terme do të hulumtohen dhe do të shpjegohen më në detal gjatë ligjëratave të ardhshme
  + Çdo klasë e veçantë ruhet në një file me emrin e njëjtë si emri i klasës por me prapashtesën ***.java***
  + Një Java file mund të ketë më tepër se një klasë në te, por vetëm njëra mund të ketë qasje publike dhe emërtimi i file-it bazohet në emrin e kësaj klase
  + Një Java aplikacion fillon nga klasa e cila ka metodën *main(…)* (shih shembullin *MyProgram.java*)

**Struktura e programit në Java**

//komentet për klasën

**public class MyProgram {**

//komentet për atributet

private String emri = “Fatos”;

……

……

//komentet për metodat

public static void main(String [] argv){ //njihet edhe si “the signature”

}

**}**

**Komentet**

* + Komentet brenda kodit njihen si “inline documentation”
  + Ato futen në kod për të dhënë shpjegime rreth kodit atyre që e lexojnë kodin
  + Nuk e afektojnë si punon programi, pasi që ato injorohen nga kompajleri
  + Ekzistojnë 3 lloje të komenteve:

// Ky koment vlen vetëm për këtë rresht

/\* Ky koment vlen për krejt bllokun dhe vlen deri sa të gjendet simboli terminues dhe mund të shkoj në disa rreshta \*/

/\*\* Ky është *javadoc* koment\*/

**Identifikuesit (ang. Identifiers)**

* + Identifikuesit janë fjalët/simbolet që i përdor programeri në një program.
  + Një identifikues mund të ndërtohet nga shkronjat, numrat, shenja (\_) (ang. underscore) si dhe nga shenja e dollarit $
  + Identifikuesi nuk mund të filloj me numër
  + Java është “casesensitive” që d.m.th – fjalët Total, total dhe TOTAL trajtohen si identifikues të ndryshëm
  + Ligj i pashkruar por i përvetësuar nga programerët që shkruajnë programe në Java, përdorin këtë formë të shkrimit së kodit:
    - Për emra të klasave përdoret Titlecase: p.sh HelloWorld
    - Emrat e atributeve dhe metodave fillojnë me shkronja të vogla: p.sh. printName
    - Emrat e konstanteve me të madhe: p.sh. MAXIMUM

**Identifikuesit**

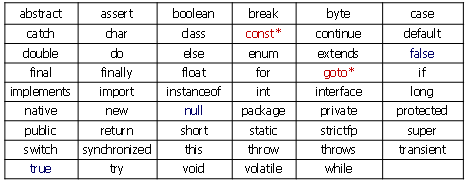
Ndonjëherë ne i zgjedhim identifikuesit tanë, si në shembullin më lart *MyProgram*

Ndonjëherë i përdorim identifikuesit të cilët i kanë zgjedh programerët e tjerë, p.sh. *println*

Shpesh përdorim identifikues të cilët njihen si të rezervuar dhe që kanë kuptim të veçantë në Java, p.sh. *int*

Një identifikues i rezervuar nuk mund të përdoret për asgjë tjetër pos për çka është i definuar

**Fjalët e rezervuara**

****

Në ueb faqen më poshtë mund të gjeni shpjegim për secilin identifikues të rezervuara në Java:

[*http://en.wikibooks.org/wiki/Java\_Programming/Keywords*](http://en.wikibooks.org/wiki/Java_Programming/Keywords)

**Hapësirat (ang. white spaces)**

* + Hapësirat, reshtat e zbrazët,TAB njihen si “white space”
  + Këto hapësira përdoren për t’i ndarë fjalët dhe simbolet në program si dhe për ta strukturuar programin që të jetë më i lexueshëm
  + Sa i përket kompajlerit këto hapësira injorohen
  + Sa i përket lexueshmërisë së kodit, ekzistojnë disa mënyra se si të formatohet programi në Java
  + Programet formatohen duke përdorë hapësira dhe “indentation”

**Zgjidhja e problemeve**

* + Qëllimi i një programi kompjuterik është zgjidhja e ndonjë problemi të veçantë
  + Zgjidhja e një problemi të veçantë konsiston nga një numër i aktiviteteve:
    - Duhet të kuptohet mirë problemi
    - Konsiderohen mirë të gjitha alternativat
    - Të dizajnohet zgjidhja
    - Të implementohet zgjidhja
    - Testohet zgjidhjen
  + Këto aktivitet nuk janë lineare por ato mbulohen nga njëra tjetra dhe bashkëveprojnë

**Zgjidhja e problemeve**

* + Çelësi i zgjidhjes së problemit është zbërthimi i problemit në pjesë më të menaxhueshme
  + Ndërtimi i një softueri nënkupton zbërthimin dhe dizajnimin e pjesëve të veçanta
  + Programimi i bazuar në objekte (ang. Object-oriented programming) është shumë i përshtatshëm për këtë lloj të zbërthimit
  + Ne gjithmonë do ta zbërthejmë zgjidhjen tonë në pjesë, të njohura si **klasë** dhe **objekte**

**Programimi i bazuar në objekte**

**(ang. Object Oriented Programming)**

* + Java është gjuhë programuese e bazuar në objekte
  + Siç nënkuptohet nga ky term objekti është entiteti bazë i një Java programi
  + Java objektet mund të përdoren në mënyrë efektive që të reprezentojnë objektet në botën reale
  + P.sh. Një objekt i Java-së mund të reprezentoj një llogari bankare
  + Çdo llogari bankare menaxhon me të dhënat që kanë të bëjnë me atë llogari bankare si dhe me pjesën për procesimin e këtyre të dhënave

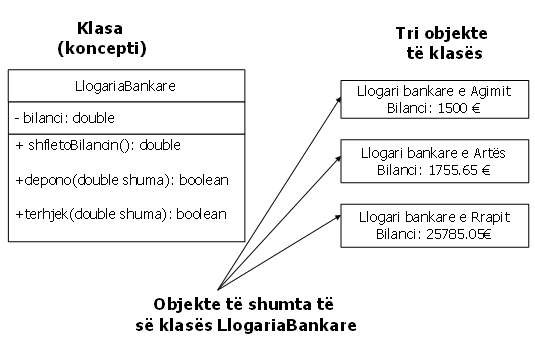
**Objektet**

* + Një objekt ka:
    - Gjendjen e tij (ang. *state*) – karakteristikat përshkruese
    - Sjelljet e tij (ang. *behaviour*) – çka mund të bëjë apo lejon t’i bëhet
  + *Gjendja* e llogarisë bankare përfshinë p.sh. Emrin, Bilancin, etj
  + *Sjelljet* që kanë të bëjnë me llogarinë bankare janë, shfletimi i gjendjes, depozitimi i mjeteve si dhe tërheqja e mjeteve
  + Siç shihet nga lart disa *sjellje* të këtij objekti ndryshojnë gjendjen e tij, p.sh. deponimi apo tërheqja e mjeteve ndryshojnë bilancin e llogarisë bankare

**Klasat**

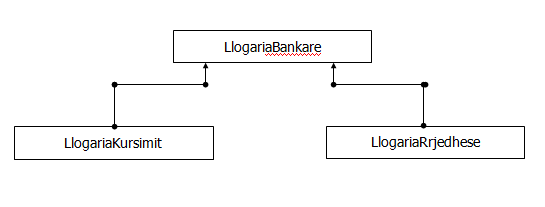
* + Një objekt definohet nga një *klasë* që reprezenton një *koncept* të veçantë
  + Klasa definon modelin e çdo instance të objektit
  + Shumë objekte mund të krijohen nga një klasë
  + Klasa ka atributet/fushat që definojnë ***gjendjen***e një objekti
  + Klasa ka metodat që definojnë ***sjelljen*** e një objekti
  + Klasa që ka metodën *main()* paraqet pikën fillestare të një programi në Java
  + Java programet kryesisht përbëhen nga më shumë klasë se vetëm klasa që ka metodën *main()*

**Objektet dhe klasat**



**Trashëgimia (ang. Inheritance)**

* + Klasat mund të organizohen në hierarki
  + Një klasë mund të derivohet/trashëgohet nga një tjetër duke përdor “inheritance”



***Variablat, Konstantet dhe Tipet e të Dhënave***

***Stringjet (ang. Strings)***

***String literal***

**Stringjet**

* + Një varg i simboleve mund të paraqitet si *string literal* duke i vënë thonjëza përreth tekstit:
  + Shembull:
    - *“Ky është string literal.”*
    - *“Rruga Nena Tereze, Nr. 123”*
    - *“X”*
  + Në Java çdo String është Objekt, i definuar përmes klasës *String* që gjendet në paketën *java.lang*.
  + Çdo String literal është një Objekti tipit *java.lang.String*

**Metoda *println***

* + Në orën e kaluar, në shembullin HelloWorld.java, kemi thirrë metodën *println* që të shtypim vargun e simboleve (String-un) “Hello World”
  + Objekti *out* i klasës *System*, pra *System.out* simbolizon një destinacion (p.sh. monitorin) të cilit ne mund t’i dërgojmë output-in, pra ku të shtypet Stringu.

*System.out.println (“Ckado qe beni, bejeni mire.”);*

*Objekti Emri i Informata që i jepet metodës*

*metodës (parametrat)*

* + Objekti *System.out* ndër shumë metoda të tjera ofron edhe metodën: *print*.
  + Metoda *print* është e ngjashme me metodën *println*, me përjashtim se kursori nuk bartet në rreshtin e ri
  + Kështu që çdo parametër që i dërgohet metodës *print* do të shtypet në të njëjtin rresht
  + P.sh.

*System.out.print (“Nje…”);*

*System.out.print (“Dy…”);*

* + Shtypet si:

*Nje… Dy…*

**Bashkëngjitja e stringjeve (ang. concatenation)**

Operatori për bashkëngjitje të stringjeve (+) përdoret që të bashkëngjitet një string në fund të stringut tjetër:

*“Pershendetje juve “+ “ nga stafi i UBT-se”*

Mund të përdoret edhe që të bashkëngjitet një numër me një string

Një string literal nuk mund të ndahet nëpër dy rreshta të programit

*System.out.println(“Pershendetje te gjitheve*

*nga une”);*

Por mundet nëse e bashkëngjisim, si më poshtë:

*System.out.println(“Pershendetje te gjitheve “+*

*“nga une”);*

**Bashkëngjitja** **String**

* + Operatori + gjithashtu përdoret si operator aritmetik i mbledhjes
  + Funksionin të cilin ky operator e kryen varet nga tipi i informatave me të cilat operon
  + Nëse të dy operandët janë të tipit String, ose një është String e tjetri numër, atëherë kryen bashkëngjitjen
  + Nëse të dy operandët janë të tipit numerik, atëherë bënë mbledhjen e tyre
  + Operatori + vlerësohet nga e majta në të djathtën, mirëpo kllapat mund të përdoren në mënyrë që të ndërrohet renditja
  + *System.out.println*(“24 dhe 45 te bashkengjitur: “ + 24 + 45);
  + Rezultati paraqitet si: *24 dhe 45 te bashkengjitur: 2445*

**Escape Sequences**

* + Çka bëhet nëse dëshirojmë të shtypim thonjëzat?
  + Shprehja më poshtë do të habiste kompajlerin sepse do të interpretonte qiftin e thonjëzave si dy stringje dhe tekstin në mes të stringjeve si gabim sintaksor:

*System.out.println (“Tung juve* “*nga” stafi i UBT-se.”);*



*String Gabim String*

*Sintaksor*

* + “Escape Sequence” është një seri e simboleve që reprezenton një simbol të veçantë
  + Escape Sequence fillon me simbolin (\) (ang. Backslash)

*System.out.println (*“*Tung juve* \”*nga*\*”*  *stafi i UBT-se*

**Escape Sequences**

* + Disa “Escape Sequences” në Java:

Escape Sequence: Domethënja:

\b Backspace

\t Tab

\n Newline/Line Feed

\r Carriage return

\f Form feed

\” Double quote

\’ Single quote

\\ Backslash

\u Unicode

* + Shembull

*System.out.println* (“Pershendetje juve,\n\tnga stafi i UBT-se” );

* + Rezultati paraqitet si:

*Pershendetje juve,*

*nga stafi i UBT-se*

***Variablat, Konstantet dhe Tipet e të Dhënave***

***Variablat, Inicializimi dhe Përcaktimi (ang. Assignment)***

***Konstantet***

***Llojet primitive të të dhënave (ang. primitive data types)***

**Variablat**

* + Variabla është emri i një lokacioni në memorie
  + Variabla duhet të deklarohet duke specifikuar emrin e variablës dhe tipin e informatës të cilën ajo do të përmbaj

tipi emri i variablës

*int total;*

* + Më shumë se një variabël mund të deklarohet në një shprehje të vetme, si më poshtë:

*int count, temp, result;*

**Inicializimi i variablave**

* + Variablës mund t’i jepet një vlerë fillestare përmes deklarimit:

*int sum = 0;*

*int base = 32, max = 149, total=0;*

* + Kur variabla referencohet në program, atëherë përdoret vlera e saj aktuale

*int keys = 88;*

*System.out.println(“Pianoja ka ” + keys + “ taste.”);*

* + Shtypet si:

*Pianoja ka 88 taste.*

**Përcaktimi i vlerës (ang. Assignment)**

* + Shprehja e përcaktimit të vlerës ndërron vlerën e një variable
  + Operatori i përcaktimit është shenja =

total = 55;



* + Shprehja në të djathtë kalkulohet/vlerësohet dhe rezultati ruhet si vlerë e variablës në të majtë
  + Vlera e mëparshme e variablës *total* mbishkruhet me vlerën e re
  + Një variable mund t’i jepet vetëm një vlerë që përputhet me tipin e deklaruar të variablës.

**Konstantet**

* + Konstantet janë identifikues që janë të ngjashëm me variablat por që mban të njëjtën vlerë përgjatë gjithë ekzistencës së saj
  + Ashtu siç nënkupton vet emri, është konstante, jo variabël
  + Nëse tentohet të ndryshohet vlera e konstantës kompajleri do të jap *compile-time* error
  + Në Java, përdoret fjala e rezervuar *final* për të deklaruar një konstantë, p.sh.

*final int MIN\_HEIGHT = 50;*

**Konstantet**

* + Konstantet janë të rëndësishme për tri arsye:
  + Së pari, ato u japin kuptim vlerave të mirëfillta:
    - *P.sh. NUM\_ALFABET do të thotë më shumë se vlera 36*
  + Së dyti, ato lehtësojnë mirëmbajtjen e programit

*- Nëse konstanta përdoret në shumë vende dhe duhet t’i ndërrohet vlera më vonë, vlera duhet të përmirësohet vetëm në një vend*

* 1. Së treti, ato formalisht tregojnë se vlerat nuk duhet të ndërrohen, duke iu shmangur gabimeve të shkaktuara nga pakujdesia e programerëve

**Të dhënat (ang. Data) primitive**

* + Ekzistojnë disa lloje të të dhënave primitive
  + Katër prej tyre simbolizojnë numrat e plotë (ang. integer):
    - * *byte, short, int, long*
  + Dy prej tyre simbolizojnë numrat me presje dhjetore (ang. floating point):
    - * *float, double*
  + Njëra simbolizon simbolet (ang. character):
    - * *char*
  + Dhe një prej tyre simbolizon vlerat boolean:

*- boolean*

**Të dhënat Numerike Primitive**

* + Diferenca në mes të llojeve të ndryshme të të dhënave numerike primitive është madhësia e tyre, që do të thotë cilat janë vlerat minimale dhe maksimale që ato mund të përmbajnë:

**Lloji Memoria Vlera Min Vlera Max**

byte 8 bits -128 127

short 16 bits -32,768 32,767

int 32 bits -2,147,483,648 2,147,483,647

long 64 bits <-9 x 1018 > 9 x 1018

float 32 bits +/- 3.4 x 1038

double 64 bits +/- 1.7 x 10308

char 16 bits

boolean 8 bits (1 për reprezentim). Vlerat e mundshme janë **true** dhe **false**

**Bashkësia e simboleve**

**(Ang. Character Set)**

* + Bashkësia e simboleve është një listë e simboleve, ku çdo simbol korrespondon në një numër unik
  + Një variabël e tipi *char* në Java mund të ruan një simbol nga bashkësia *Unicode* e simboleve
  + Bashkësia *Unicode* e simboleve përdorë 16 bit për çdo simbol, që mundëson 65,536 simbole unike

Është bashkësi e simboleve e njohur ndërkombtarisht dhe përmban simbole nga shumë gjuhë botërore

**Simbolet**

* + Bashkësia *ASCII* e simboleve është më e vjetër dhe mund të paraqes më pak simbole se *Unicode,* 8 bit*,* por ende është mjaft e popullarizuar
  + *ASCII* simbolet janë nënbashkësi e simboleve *Unicode* ku përfshihen:
    - Shkronjat e mëdha A,B,C,…
    - Shkronjat e vogla a,b,c,…
    - Shenjat e pikësimit pikë, presje, dy pika, …
    - Numrat 1,2,3,…
    - Simbolet e ve**ç**anta &,\,”,…
    - Simbolet e kontrollit “carriage return”, “tab”, ...

**Boolean**

* + Një vlerë *boolean* paraqet një kusht që mund të jetë i vërtetë apo i pavërtetë
  + Vlerat ***true*** dhe ***false*** janë fjalë të rezervuara dhe vlerat e vetme të mundshme për variablat e tipit *boolean*.

*boolean perfundoi = false;*

* + Një variabël e tipit *boolean* mund të paraqes vetëm dy gjendje, p.sh. Nëse marrim llambën ajo mund të jetë e ndezur ose e fikur

*boolean eNdezur= true;*

***Shprehjet (Statements), Konvertimi (Shndërrimi) i të dhënave dhe “Input”-i***

***Shprehjet aritmetike***

***Operatorët dhe precedenca***

***Operatorët e përcaktimit (ang. Assignment)***

***Konvertimi i të dhënave***

**Shprehjet aritmetike**

* + Një shprehje aritmetike është kombinim i dy apo më shumë operandëve dhe operatorëve
  + Shprehjet aritmetike llogarisin rezultate numerike dhe përdorin operandë numerik:

Mbledhja +

Zbritja -

Shumëzimi \*

Pjesëtimi /

Mbetja %

* + Nëse njëri apo që të dy operandët e përdorur në një shprehje, janë numra me presje dhjetore, atëherë rezultati do të jetë një vlerë me presje dhjetore.

**Pjesëtimi dhe mbetja**

* + Nëse të dy operandët e operatorit të pjesëtimit (/) janë integjer (numra të plotë), rezultati do të jetë integjer (numrat pas presjet dhjetore nuk merren parasysh):

14 /3 e barabartë me 4

8 / 12 e barabartë me 0

* + Operatori i mbetjes (%) kthen mbetjen pasi që të pjesëtoi operandin e parë me të dytin:

14 % 3 e barabartë me 2

8 % 12 e barabartë me 8

**Precedenca e operatorit**

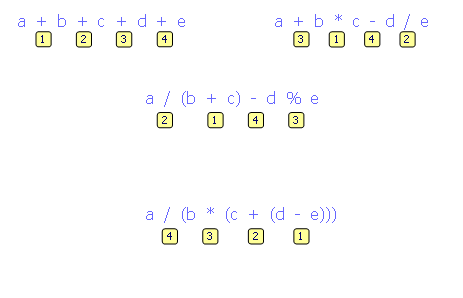
* + Operatorët mund të kombinohen në shprehje komplekse

result = total + count / max – offset;

* + Operatorët e kanë të definuar precedencën, gjë që determinon rendin sipas të cilit kryhet kalkulimi
  + Shumëzimi, pjesëtimi dhe mbetja kryhen para mbledhjes dhe zbritjes
  + Te operatorët aritmetik me të njëjtën precedencë, shprehja kalkulohet nga e majta në të djathtë, mirëpo kllapat mund të përdoren në mënyrë që renditja e kalkulimit të ndërrohet

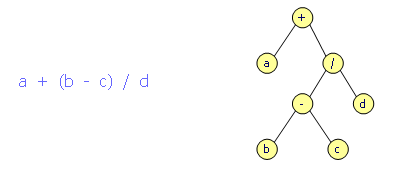
**Precedenca e Operatorëve**

* + Cila është renditja e kalkulimit në shprehjet e mëposhtme?

**

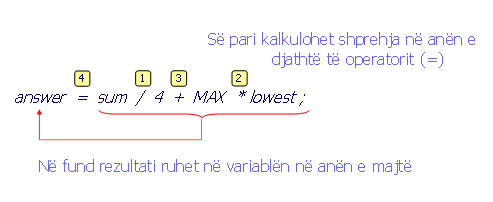
**Trungu i Shprehjes**

* + Kalkulimi i një shprehjeje të veçantë mund të paraqitet me ndihmën e trungut të shprehjeve
  + Operatorët që qëndrojnë më ulët në trung kanë precedence më të lartë në atë shprehje:



**Rishikim i operatorit të përcaktimit**

* + Operatori i përcaktimit ka përparësinë më të ulët se të gjithë operatorë aritmetik



**Rishikim i operatorit të përcaktimi**

* + Ana e majtë edhe e djathtë e operatorit të përcaktimit mund të përmbajë të njëjtën variabël

Së pari, vlera një i shtohet vlerës origjinale të

variablës count

count = count + 1;

Pastaj rezultati ruhet përsëri në count

(pra zëvendëson rezultatin origjinal)

**Shtimi dhe Pakësimi**

**(Increment dhe Decrement)**

* + Operatorët e rritjes dhe zvogëlimit përdorin vetëm një operand
  + Operatori i shtimit (++) e rritë vlerën e operandit për 1
  + Operandi i zbritjes (--) e zbret vlerën e operandit për 1
  + Shprehja

count++;

Sipas funksionit është ekuivalent me shprehjen:

count = count + 1;

**Shtimi dhe Pakësimi**

**(Increment dhe Decrement)**

* + Operatorët e shtimit dhe pakësimit mund të aplikohen në formën

*postfix:*

count++

ose, *prefix:*

++count

* + Të përdorura si pjesë të një shprehjeje më të gjatë, këto dy forma mund të kenë efekte të ndryshme
  + Për shkak të ndjeshmërisë së tyre, operatorët e shtimit dhe pakësimit duhet të përdoren me shumë kujdes

**Operatorët e Përcaktimit**

* + Shpesh ne kryejmë një veprim mbi variabël, dhe më tutje ruajmë rezultatin në të njëjtën variabël
  + Java ofron operatorë të tillë të përcaktimit që të lehtësoj këtë proces
  + Për shembull, deklarimi:

num += count;

është ekuivalent me:

num = num + count;

**Operatorët e Përcaktimit**

* + Ekzistojnë më shumë operatorë të tillë në Java:

**Operatori** **Shembulli** **Me vlerë të njëjtë si**

+= x += y x = x + y

-= x - = y x = x – y

\*= x \* = y x = x \* y

/= x / = y x = x / y

%= x % = y x = x % y

**Operatorët e Përcaktimit**

* + Ana e djathtë e operatorit të përcaktimit mund të jetë ndonjë shprehje komplekse
  + Shprehja në anën e djathtë, së pari, kalkulohet në tërësi, pastaj rezultati kombinohet me variablën origjinale
  + Kështu që

result /= (total – MIN) % num ;

është ekuivalent me:

result = result / ((total – MIN) % num);

**Operatorët e Përcaktimit**

* + Sjellja e disa operatorëve të përcaktimit varet nga lloji i operandit
  + Nëse kemi të bëjmë më operandë të tipit String, operatori i përcaktimit kryen veprimin e bashkëngjitjes së stringjeve
  + Sjellja e një operatori të përcaktimit (+=) është gjithmonë i njëjtë me sjelljen e operatorit korrespondues (+) dhe (=)

**Konvertimi i të dhënave**

**(Data Conversion)**

* + Ndonjëherë është e nevojshme të konvertohen të dhënat nga një lloj në tjetrin
  + Për shembull, në një situatë të veçantë ndoshta na duhet të trajtomë integjerin si vlerë me presje dhjetore
  + Këto konvertime nuk ndryshojnë llojin e variablës ose vlerën që është e ruajtur në te – ato vetëm konvertojnë vlerën si pjesë të llogaritjes

**Konvertimi i të dhënave**

* + Konvertimi duhet të përdoret me shumë kujdes që t’i shmangemi humbjes së informatës
  + *Zgjerimi i konvertimit* është më i sigurt pasi që konvertohet tipi më i vogël në tipi më të madh të variablës (p.sh. *short* në *int*)
  + *Ngushtimi i konvertimit* është ekundërtadhe gjatë këtij lloji të konvertimit mund të humbet informata (p.sh. *int* në *short*)
  + Në Java konvertimi mund të paraqitet në tri forma:

- konvertimi gjatë përcaktimit të vlerës (Assignment Convertion)

- promovimet (Promotions)

- përshtatja e llojit (ang. Typecasting)

**Konvertimi gjatë përcaktimit të vlerës**

* + Konvertimi gjatë përcaktimit ndodh kur një vlerë e një lloji të variablës i përcaktohet/jepet variablës së llojit tjetër
  + P.sh. nëse *money* është variabël e tipit *float* ndërsa *dollars* është një variabël e tipit *int*, përcaktimi në vijim e konverton vlerën e variablës *dollars* në *float*

money = dollars

* + Vetëm *Konvertimet Zgjeruese* mund të ndodhin në konvertimin gjatë përcaktimit të vlerës
  + Vëreni: vlera dhe lloji i variablës *dollars* nuk ka ndryshuar

**Konvertimi i të dhënave**

**Promovimi**

* + *Promovimi* ndodh automatikisht kur operatorët në shprehje konvertojnë operandët e tyre
  + Për shembull, nëse *sum* ështëvariabël e llojit *float* dhe *count* është variabël e llojit *int*, vlera e *count* konvertohet në vlerë me presje dhjetore në mënyrë që të kryhet kalkulimi i më poshtëm:

result = sum / count ;

* + Normal që edhe variabla *result* do të jetë e llojit float.

**Përshtatja e llojit të variablës (Typecasting)**

* + *Typecasting* është teknika (metoda) më e fuqishme dhe më e “rrezikshme” e konvertimit.
  + Edhe zgjerimi edhe ngushtimi i konvertimi mund të realizohen përmes “typecasting” eksplicit të vlerës.
  + Që të realizohet ky lloj i konvertimit, lloji në të cilën konvertohet variabla futet në kllapa dhe shënohet para variablës që konvertohet
  + Për shembull, nëse variabla *total* dhe *count* janë integjerë, por ne dëshirojmë rezultat me presje dhjetore kur ato të pjesëtohen, ne mund ta arrijmë këtë duke e “typecast” variablën *total*:

result = (float) total / count ;

***Shprehjet (Statements), Konvertimi (Shndërrimi) i të dhënave dhe “Input”-i***

***“Input-”i dhe klasa Scanner***

**Leximi i “input”-it**

* + Pasi që klasa *Scanner* gjendet në paketën *java.util* ajo duhet të importohet në program para se të përdoret.
  + Çdo klasë e cila nuk gjendet në paketën *java.lang* duhet të importohet para se të përdoret në program.
  + Importimi i klasave që gjenden në librari arrihet përmes komandës *import* dhe shkruhet në fillim të programit. P.sh.

import java.util.Scanner;

ose

import java.util.\*;

* + Metoda *nextLine* e klasës *Scanner* lexon krejt një rresht, pra lexon deri sa të gjendet fundi i rreshtit

**Talonet (Token) e “input”-it**

* + Vetëm në rastet kur specifikohet ndryshe, zbrazëtirat (White space) përdoren për ndarjen e elementeve/taloneve të një “input”-i (të njohura si “token”)
  + Zbrazëtirat përfshijnë simbolet si hapësira(space), tab, rreshti i ri (new line), etj
  + Metoda *next* e klasës *Scanner* lexon shenjën e ardhshme të “input”-it dhe e kthen atë si një *String*
  + Metodat siç janë *nextInt* dhe *nextDouble* janë metoda specifike që mundësojnë leximin e të dhënave të veçanta, si *int* dhe *double*
  + Para se të përdoren metodat e klasës Scanner, duhet të krijohet një instance e re e këtij objekti.
  + Krijimi i një instance të klasës Scanner që lexon nga default “input”-i

Scanner scan = new Scanner(System.in);

* + Metodat për lexim të “input”-it aplikohen në instance:

String fjala = scan.next(); //lexon një fjalë nga fjalia e shtypur

String fjalia = scan.nextLine(); //lexon tërë fjalinë e shtypur

int intNr= scan.nextInt(); //lexon numrin e shtypur si int

double doubleNr = scan.nextDouble(); //lexon numrin e shtypur si double

boolean boolValue = scan.nextBoolean(); //lexon një vlerë të tipit boolean

* + Pos metodës për leximin e char, ekzistojnë metoda për leximin e të gjitha të dhënave të thjeshta

***Shprehjet boolean dhe if***

***Rrjedhshmëria e kontrollit/Deklaratat e kushtëzuara***

***Deklarata if***

***Operatorët logjik***

***Klauza else***

***Bllok Deklaratat***

***Deklaratat e ndërthurura if***

**Rrjedhshmëria e Kontrollit**

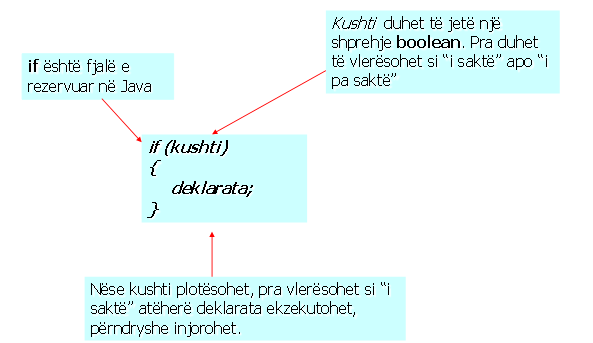
* + Përveç nëse është e specifikuar ndryshe, renditja e ekzekutimit të deklaratave në metodë janë lineare:
    - *Një deklaratë pas tjetrës në mënyrë sekuenciale*
  + Disa deklarata në programim lejojnë që:
    - *të vendosim nëse do të ekzekutojmë një deklaratë të veçantë apo jo*
    - *të përsërisim ekzekutimin e një deklarate më shumë se një herë*
  + Këto vendime janë të bazuara në shprehjet e llojit *boolean* (në kushte) që vlerësohen ‘të sakta’ ose ‘të pasakta’
  + Renditja e ekzekutimit të këtyre deklaratave njihet si *kontrolli i rrjedhshmërisë (ang. Control Flow)*

**Deklaratat e Kushtëzuara**

* + Deklaratat e kushtëzuara na lejojnë që të zgjedhim se cila është deklarata e ardhshme që do të ekzekutohet
  + Prandaj ato ndonjëherë njihen edhe si *deklarata të zgjidhjes (ang. Selection Statement)*
  + Deklaratat e kushtëzuara na japin mundësinë që të sjellim vendime bazike, se cila deklaratë/shprehje e ardhshme do të ekzekutohet
  + Në Java deklaratat e kushtëzuara janë:
    - *Deklarata* ***if***
    - *Deklarata* ***if-else***
    - *Deklarata* ***switch***

**Deklarata if**

* + Deklarata ***if*** ka këtë sintaksë :



**Kushtet/Shprehjet e llojit boolean**

* + Një kusht shpesh arrihet duke përdorur operatorë të barazimit dhe/ose operatorë *relacional* që krijojnë një shprehje të llojit ***boolean*** dhe që kthejnë një rezultat të tipit***boolean****:*

== e barabartë me

!= jo e barabartë me

< më e vogël se

> më e madhe se

<= më e vogël ose e barabartë me

>= më e madhe ose e barabartë me

* + Vëreni diferencën në mes të operatorit të barazimit (**==**) dhe të operatorit të përcaktimit **(=)**
  + Një shembull i deklaratës **if**:

if (sum > MAX) {

delta = sum - MAX;

}

System.out.println (“Ndryshimi është ” + delta);

* + Së pari kushti vlerësohet nëse është plotësuar, (në shembullin tonë shikohet nëse vlera e variablës ***sum*** është më e madhe se vlera e konstantës ***MAX***)
  + Nëse kushti plotësohet, deklarata e përcaktimit ekzekutohet (në shembullin tonë *delta=sum-MAX*), nëse jo atëherë injorohet.

Sidoqoftë, thirrja e ***println*** është e deklarata e ardhshme që ekzekutohet

**Indentimi (ang. Indentation)**

* + Deklarata e kontrolluar nga deklarata ***if*** ,është e indentuar në mënyrë që të paraqitet relacioni në fjalë
  + Përdorimi konsekuent i këtij stili të “indentimit” e bënë programin më të lexueshëm dhe më të lehtë për ta kuptuar
  + Megjithëse nuk ka ndonjë rëndësi sa i përket kompajllerit, “indentimi” i saktë është shumë i rëndësishëm për personat që e lexojnë kodin e programit

**Operatorët Logjik**

* + Operatorët e mëposhtëm logjik mund të përdoren gjithashtu në shprehjet e tipit boolean:

! Operatori logjik NOT

&& Operatori logjik AND

|| Operatori logjik OR

* + Ato operojnë në operand të tipit ***boolean*** dhe rezultatet që fitohen gjithmonë janë të tipit ***boolean***
    - Operatori logjik NOT është operator ***unar*** (ang. unary)(vepron vetëm në një operand)
    - Operatorët logjik AND dhe OR janë operatorë ***binar (ang. binary)*** (veprojnë në dy operandë)

**Operatori logjik NOT**

* + Veprimi i operatorit logjik NOT njihet edhe si *negacioni logjik* ose *komplimenti logjik*
  + Nëse një shprehje apo një variabël **a** e tipit boolean është e saktë, atëherë **!a** është e pasaktë;
  + Nëse **a** është e pasaktë, atëherë  **!a** është e saktë
  + Veprimet logjike mund të tregohen edhe përmes tabelës së saktësisë:

|  |  |
| --- | --- |
| **a** | **!a** |
| e saktë | e pasaktë |
| e pasaktë | e saktë |

**Operatorët logjik AND dhe OR**

* + Shprehja logjike AND

a && b (në Java)

është e saktë nëse edhe **a** dhe **b** janë të sakta, përndryshe e gjithë shprehja është e pasaktë

* + Shprehja logjike OR

a || b (në Java)

është e saktë nëse së paku njëra nga **a** apo **b**, apo edhe të dyja, janë të sakta, përndryshe e gjithë shprehja është e pasaktë

**Operatorët Logjik**

* + Tabela e vërtetësisë tregon të gjitha kombinimet e mundshme (e saktë/e pasaktë) të shprehjeve
  + Pasi që edhe operatori && edhe operatori || veprojnë në dy operandë, atëherë ekzistojnë katër kombinime të mundshme për kushtet **a** dhe **b**

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **a && b** | **a || b** |
| e saktë | e saktë | e saktë | e saktë |
| e saktë | e pasaktë | e pasaktë | e saktë |
| e pasaktë | e saktë | e pasaktë | e saktë |
| e pasaktë | e pasaktë | e pasaktë | e pasaktë |

**Operatorët “në qark të shkurtër”**

**(Ang. Short-Circuited)**

* + Procesimi i operatorëve logjik **AND** dhe **OR** është i “*qarkut të shkurtër”*
  + Nëse operandi i majtë, apo kushti i parë është i mjaftueshëm që të përcaktohet i gjithë rezultati, operandi i djathtë nuk testohet fare

*if (count != 0 && total/count > MAX)*

*System.out.println (“Testing…”) ;*

Kjo teknikë e kodimit duhet të përdoret me shumë kujdes

**Deklarata if-else**

* + Klauza ***else*** mund t’i shtohet një deklarate ***if*** që të krijohet një deklaratë ***if-else***

*if (kushti)*

*deklarata1;*

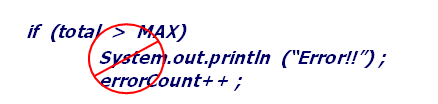
*else*

*deklarata2;*

* + Nëse *kushti* është i saktë, ekzekutohet **deklarata1** përndryshe ekzekutohet **deklarata2**
  + Njëra ose tjetra deklaratë do të ekzekutohet, por jo që të dyja

**Indentimi (vazhdim)**

* + Mos harroni se “indentimi” është për njerëzit dhe injorohet nga kompajlleri i Java-së

******

*Pa marrë parasysh se çka nënkuptohet me “indentim”,*

*rritja e vlerës së variablës errorCount për një do të ndodhë*

*pa marrë parasysh nëse kushti është i saktë apo i pasaktë*:

***if (total > MAX)***

***System.out.println (“Error!!”) ;***

***errorCount++ ;***

**Bllok deklaratat**

* + Disa deklarata mund të grupohen në bllok, të definuar me kllapat gjarpërore, si më poshtë:

***if (total > MAX)***

***{***

***System.out.println (“Error!!”) ;***

***errorCount++;***

***}***

tani rritja do të ndodhë vetëm kur

kushti ***if*** është i saktë

Bllok deklarimi mund të përdoret edhe kur kemi një deklarim të vetëm apo edhe kur numri i deklaratave brenda bllokut është i pa limituar

**Bllok deklaratat**

* + Në një deklaratë ***if-else,*** edhe pjesa ***if*** edhe pjesa ***else*** mund të jenë bllok deklarata

***if (total > MAX)***

***{***

***System.out.println (“Error!!”) ;***

***errorCount++;***

***}***

***else***

***{***

***System.out.println (“Total: ” + total) ;***

***current = total\*2;***

***}***

**Testimi për më shumë kushte**

* + Një deklaratë ***if-else,*** mund të testohet për më shumë kushte

***if (total > MAX)***

***{***

***System.out.println (“Error!!”) ;***

***errorCount++;***

***}***

***else if (total == MAX)***

***{***

***System.out.println (“Vlerat janë të barbarta!”) ;***

***}***

***else***

***{***

***System.out.println (“Total: ” + total) ;***

***current = total\*2;***

***}***

**Operatori kushtëzues**

* + Java ofron edhe një operator kushtëzues, që përdor një kusht ***boolean*** të determinojë se cila prej dy shprehjeve duhet të ekzekutohet
  + Sintaksa e këtij operatori është:

kushti ? shprehja1 : shprehja2

* + Nëse ***kushti*** është i saktë, ekzekutohet ***shprehja1***; përndryshe ekzekutohet ***shprehja2;***
  + Vlera e gjithë operatorit kushtëzues është vlera e shprehjes së zgjedhur nga kushti, pra ose ***shprehja1*** ose ***shprehja2***
  + ***Operatori kushtëzues është i ngjashëm me deklaratën if-else, përveç se kjo është shprehje e cila kthen një vlerë të vetme***
  + ***Për shembull:***

***larger = (num1 > num2) ? num1 : num2;***

* + ***Nëse num1 është më i madh se num2, atëherë vlera e num1 i përcaktohet variablës larger; përndryshe num2 i përcaktohet variablës larger***
  + ***Operatori kushtëzues është i ternar (ang. ternary) sepse nevojiten tre operandë: një kusht dhe dy vlera alternative***

**Deklaratat e ndërthurura if**

* + Deklarata e ekzekutuar si rezultat i një deklarate ***if*** ose një klauze ***else*** mund të jetë një deklaratë tjetër ***if***
  + Këto njihen si ***deklarata të ndërthurura if***
  + Një klauzë ***else*** i takon ***if*** –it të fundit (pa marrë parasysh se çka tregohet/shihet nga indentimi i kodit)
  + Kllapat mund të përdoren që të specifikohet deklarata ***if*** të cilës i takon klauza ***else***

***Krahasimi i të dhënave***

***dhe klasa Math***

*Krahasimi i të dhënave*

*Klasa Math*

**Krahasimi i të dhënave**

* + Kur krahasojmë të dhënat duke përdorur shprehje “boolean”, është e rëndësishme të kuptohen nuancat e disa llojeve të veçanta të të dhënave
  + T’i shqyrtojmë disa situata të rëndësishme:
    - Krahasimi i vlerave me presje dhjetore për barazi (operatorët relacional)
    - Krahasimi i simboleve (ang. Character)
    - Krahasimi i stringjeve (sipas radhitjes alfabetike)
    - Krahasimi i objekteve kundruall krahasimit të referencave të objekteve

**Krahasimi i vlerave me presje dhjetore**

* + Kur kemi të bëjmë me krahasimin e dy vlerave me presje dhjetore, operatori i barazimit (==) duhet të përdoret me mjaft kujdes dhe nëse mund të evitohet atëherë duhet të evitohet
  + Dy vlera me presje dhjetore janë të barabarta vetëm nëse reprezentimi binar i tyre plotësisht përshtatet
  + Llogaritjet shpesh rezultojnë në diferenca të vogla që mund të jenë të parëndësishme
  + Në shumë raste, dy numra me presje dhjetore mund të konsiderohen se janë “mjaft të përafërta” edhe pse nuk janë plotësisht të barabarta
  + Në mënyrë që të vërtetohet nëse dy vlera me presje dhjetore janë të “përafërsisht të barabartë”, mund të përdorim teknikën si më poshtë:

if (Math.abs (f1 - f2) < TOLERANCA)

System.out.println (“Perafersisht te barabarte”);

* + Nëse vlera absolute e diferencës në mes të dy vlerave me presje dhjetore është më e vogël se TOLERANCA (vlera e lejuar), ato konsiderohen të jenë të barabarta
  + Toleranca mund të vihet në një nivel të duhur, si p.sh. 0.000001
  + Në mënyrë që të vërtetohet nëse dy vlera me presje dhjetore janë të “përafërsisht të barabartë”, mund të përdorim teknikën si më poshtë:

if (Math.abs (f1 - f2) < TOLERANCA)

System.out.println (“Perafersisht te barabarte”);

* + Nëse vlera absolute e diferencës në mes të dy vlerave me presje dhjetore është më e vogël se TOLERANCA (vlera e lejuar), ato konsiderohen të jenë të barabarta
  + Toleranca mund të vihet në një nivel të duhur, si p.sh. 0.000001

**Krahasimi i simboleve (ang. Characters)**

* + Sikur që kemi diskutuar edhe më parë, simbolet në Java janë të bazuar në bashkësinë Unicode të simboleve (16 bit)
  + Sipas bashkësisë Unicode të simboleve, çdo simboli i përcaktohet një numër i veçantë dhe kështu krijohet një radhitje e simboleve
  + Ne mund të përdorim operatorët relacional për krahasimin e simboleve duke u bazuar në këtë radhitje
  + Për shembull, simboli ‘A’ ka vlerë më të vogël se simboli ‘J’ sepse radhitet para saj në bashkësinë Unicode të simboleve.

Më shumë informata për Unicode mund të gjeni në: [**http://unicode.org/**](http://unicode.org/)

**Krahasimi i simboleve**

* + Në bashkësinë Unicode të simboleve, simbolet numerike (‘0’- ‘9’) janë fqinjësore dhe radhiten sipas vlerës së tyre numerike
  + Në të njëjtën mënyrë, shkronjat e mëdha (ang. uppercase) (A-Z) dhe shkronjat e vogla (ang. lowercase) (a-z) janë fqinjësore dhe radhiten sipas radhitjes alfabetike

|  |  |
| --- | --- |
| Simbolet | Vlerat në Unicode |
| 0 - 9 | 48 deri 57 |
| A - Z | 65 deri 90 |
| a - z | 97 deri 122 |

**Krahasimi i simboleve**

* + Prandaj, nëse dëshirojmë të mbështesim një vendim në programin tonë të bazuar në faktin se a është një simbol numerik apo jo, mund të përdorim kodin siç vijon:

if (character >= ‘0’ && character <= ‘9’ )

System.out.println (“Eshte numer!”) ;

* + Gjithashtu mund të vërtetojmë nëse një simbol është shkronjë e madhe apo jo:

if (character >= ‘A’ && character <= ‘Z’ )

System.out.println (“Eshte shkronje e madhe!”) ;

**Krahasimi i Stringjeve**

* + Mos harroni se në Java një String është ***objekt*** e jo e dhënë e thjeshtë (ang. Simple/primitive datatype)
  + Operatori (==) nuk mund të përdoret që të krahasohen vlerat e dy stringjeve nëse janë të barabartë apo jo për nga përmbajtja.
  + Në vend të operatorit == duhet të përdoret metoda ***equals*** që të determinohet nëse dy stringje përmbajnë saktësisht të njëjtat simbole me radhitje të njëjtë
  + Dhe rezultati i metodës ***equals*** është një ‘boolean’

if (emri1.equals (emri2))

System.out.println (“Emri i njejte”) ;

**Krahasimi i Stringjeve**

* + Operatorët relacional nuk mund të përdoren për krahasimin e stringjeve
  + Për këtë qëllim klasa ***String*** përmban metodën ***compareTo*** që përdoret të determinohet nëse një String vjen para një tjetri
  + Nëse thirret emri1.compareTo (emri2)

- kthen zero nëse ***emri1*** dhe ***emri2*** janë të barabartë (përmbajnë simbolet e njëjta)

- kthen vlerë negative nëse ***emri1*** është më i vogël se ***emri2***

- kthen vlerë pozitive nëse ***emri1*** është më i madh se ***emri2***

if (emri1.compareTo (emri2) < 0)

System.out.println (emri1 + “eshte i pari”) ;

else if (emri1.compareTo (emri2) == 0)

System.out.println (“Emrat jane te barabarte”) ;

else

System.out.println (emri2 + “eshte i pari”) ;

* + Për shkak se krahasimi i simboleve dhe stringjeve është i bazuar në bashkësinë e simboleve, njihet si ***renditja alfabetike***

**Renditja Leksikografike**

* + Renditja leksikografike nuk është absolutisht alfabetike kur kemi përzierje të simboleve (shkronjave) të mëdha dhe të vogla
  + Për shembull, stringu “*I madh*” vjen para stringut “*fantastik*” sepse shkronjat e mëdha në bashkësinë Unicode të simboleve kanë vlerë më të vogël se shkronjat e vogla. Kështu që, ‘I’ është më i vogël se ‘f’
  + Gjithashtu, stringjet më të shkurtra (me më pak simbole) vijnë para stringjeve më të gjata edhe pse e kanë të njëjtën parashtesë (bazuar në alfabet)
  + Kështu që “*madh*” vjen para “*madhështor*”

**Krahasimi i objekteve**

* + Operatori == mund të aplikohet mbi objekte dhe kthen prapa “e saktë” (true) nëse dy referencat janë pseudonime të njëra tjetrës, pra i referohen të të njëjtit objekt
  + Metoda e ***equals*** është e definuar për të gjitha objektet, dhe mund të mbishkruhet (ri-definohet) në klasën e re dhe ka semantikë (kuptim) të njëjtë si operatori ==
  + Kur të krijoni një klasë të re, metoda ***equals*** mund të mbishkruhet (ri-definohet) që të kthej rezultatin “e saktë” (true) për të gjitha kushtet e përshtatshme
  + Metoda ***equals*** *ë*shtë ri-definuar edhe në klasën String, që të mundësohet krahasimi i simboleve të dy stringjeve

**Aplikacionet Interaktive (CLI)**

Një program interaktiv në “Command Line Interface” përmban një seri veprimesh me qëllim që:

* + - * Të udhëzoj shfrytëzuesin të shkruaj ndonjë të dhënë
      * Të lexoj dhe t’i ruaj të dhënat e shtypura nga shfrytëzuesit
      * Të procesoj të dhënat pasi që të gjitha të dhënat e kërkuara të

jenë pranuar

* + Shfrytëzuesin mund ta informojmë:

*System.out.println (“Shkruaje tekstin: ”) ;*

* + Ndërsa lexojmë dhe formatizojmë të dhënat e shfrytëzuesit, si vijon:

*lloji variabla = skaneri.nextLloji() ; p.sh.*

*int numri1 = skaneri.nexInt();*

**Aplikacionet Interaktive**

*int a, b, c; // koeficientet e llojit integjer*

*Scanner scan = new Scanner (System.in) ; //krijo një instancë të Scanner*

*System.out.print(“Shkruaje koeficientin A: ”); //udhëzo shfrytëzuesin*

*a = scan.nextInt () ; //lexo input-in nga shfrytëzuesi si* ***int*** *dhe ruaji në*

*//variablën* ***a***

*System.out.print(“Shkruaje koeficientin B: ”);*

*b = scan.nextInt () ;*

*System.out.print(“Shkruaje koeficientin C: ”);*

*c = scan.nextInt () ;*

*// Të dhëna për të zgjedhur ekuacionin kuadratik*

*// ax2+ bx + c=0*

**Ne kemi vlerat e inputit, e pastaj?**

* + Për të zgjedhur ekuacionin kuadratik, ne duhet t’i programojmë në Java formulat e mësuara në algjebrën e shkollës së mesme:

diskriminanti = b2 – 4ac

rrënja katrore1 = (-b + rrënja katrore e diskriminantit)/2a

rrënja katrore2 = (-b – rrënja katrore e diskriminantit)/2a

* + Si i programojmë këto ekuacione?
  + Mënyra më e lehtë është të përdorim klasën *Math* gjatë kalkulimit të ekuacionit

**Klasa Math**

Klasa *Math* është pjesë e pakos j*ava.lang*

Klasa *Math* përmban metoda që kryejnë funksione të ndryshme matematikore

Ndër të tjera përfshihen edhe funksionet (metodat) e më poshtme:

vlerën absolute

rrënjën katrore

ngritjen në fuqi

funksionet trigonometrike etj..

* + Metodat e klasës matematikore janë *metoda statike* (gjithashtu të njohura si *metoda të klasës*)
  + Metodat statike mund të thirren përmes emrit të klasës – nuk nevojitet të ekzistoj ndonjë objekt i *klasës Math*

vlera = Math.cos (90) + Math.sqrt (delta) ;

* + Ngjashëm

diskriminanti = Math.pow (b, 2) – 4.0 \* a \* c ;

root1 = (-1.0 \* b + Math.sqrt (diskriminanti)) / (2.0 \* a) ;

root2 = (-1.0 \* b – Math.sqrt (diskriminanti)) / (2.0 \* a) ;

* + Veni re: Ne nuk mund të programojmë formulën me + - në Java, por duhet ta kalkulojmë secilën ndaras.

**Zgjidhja e Ekuacioneve Kuadratike**

* + Megjithatë, programi i cili bazohet vetëm në aplikimin e formulave për zgjedhjen e rrënjëve të ekuacioneve kuadratike është i pamjaftueshëm!
  + Ekuacionet për kalkulimin e rrënjëve janë në rregull por në program nuk japin gjithmonë rezultatet e pritura
  + Pasi që shfrytëzuesi, si vlera të koeficienteve, mund të jap çfarëdo vlera të tipit integjer, ne duhet të analizojmë rastet speciale të mundshme ku nuk mund të llogaritet formula apo rezultati nuk është korrekt, duke u bazuar vetëm në vlerat e input-it të **a**, **b** dhe **c**
  + Le të jetë kjo hyrje në Projektin tuaj të parë
  + Shfrytëzuesi mund të shkruajnë çfarëdo vlere për **a**, **b**, dhe **c**
  + Nëse shfrytëzuesi shkruan vlera të cilat nuk mund të llogariten si duhet duke përdorur formulat, atëherë programi nuk do të mund të funksionoi në mënyrë korrekte
  + Të provojmë a = 1, b = 0, dhe c = 1
  + Programi për të dy rrënjët nxjerr përgjigjen NaN.
    - NaN që qëndron për (Nuk është Numër)
  + Arsyeja?
  + Me këto vlera të koeficientit, formula për llogaritjen e rezultatit të diskriminantit jep numër negativ:

diskriminanti = Math.pow(b, 2) – 4.0 \* a \* c;

diskriminanti = Math.pow(0, 2) – 4.0 \* 1 \* 1;

diskriminanti = -4.0;

* + Më vonë në kalkulimin e rrënjëve, formula përdorë rrënjën katrore të diskriminantit
  + Matematikisht, një numër negativ nuk ka rrënjë katrore të “vërtetë”
  + Metoda *Math.sqrt()* nuk mund të ofrojë një numër të “vërtetë” që është rrënjë katrore e -4.0
  + Në këtë rast, kthen rezultatin “NaN”
  + Megjithatë, në algjebër ekziston mundësia që të “nxjerrim” rrënjën katrore të një numri negativ, duke përdorur numrin “imagjinar” *i* (rrënja katrore e -1)

Math.sqrt(-4.0) mund të paraqitet si

Math.sqrt(-1.0 \* 4.0) që mund të paraqitet si

Math.sqrt(-1.0) \* Math.sqrt(4.0) që jep rezultatin

i \* 2.0 ku **i** është rrënja katrore e **-1**

* + Si mund ta bëjmë programin tonë të printoj këtë përgjigje?
  + Ne na nevojitet të shkruajmë programin në mënyrë që kodi ynë të kontrollojë vlerën e diskriminantit para se të provoj të nxjerrë rrënjën katrore të saj
  + Nëse vlera e diskriminantit është negative, ne na nevojitet të konstruktojmë përgjigjen korrekte

= “i \* “ + Math.sqrt (Math.abs (-4.0)) ;

* + Ky kod do të ofroj rezultatin si String

= “i \* 2.0”

* + Ka edhe vlera të tjera të mundshme të **a**, **b**, dhe **c** që mund të rezultojnë në NaN ose në një rezultat jo të vlefshëm
  + Po supozojmë se shfrytëzuesi shkruan vlerën 0 për variablën **a**
  + Formula për rrënjë pjesëtohet me (2.0 \* a)
  + Nëse vlera e **a** është 0, pjesëtimi nuk do të jetë i mundshëm
  + Vlerësimi i shprehjes do të ofroj rezultatin *NaN* ose një *Exception*
  + Përsëri, ne na duhet të shkruajmë programin në mënyrë që kodi ynë të testoi vlerën e **a** para se të provoi ta kryej pjesën e pjesëtimit në ekuacion
  + Pra nëse vlera e **a** është 0, a ka zgjedhje?
  + Përgjigja është **PO**. Ta analizojmë një ekuacion ku **a** = 0

në (ax2+ bx + c=0)

0 \* Math.pow(x, 2) + b \* x + c = 0 është e njëjtë si:

b \* x + c =0 që mund të zgjidhet si ekuacion linear:

x = -c/b përderisa b nuk është e barabartë me 0

vërej se: Tani ekziston vetëm një rrënjë –

* + Duke u bazuar në pjesën më lartë, ne mund të shohim edhe një rast tjetër të veçantë, pra nëse **b** është e barabartë me 0 (por vetëm nëse edhe **a** është gjithashtu e barabartë me 0)
  + Është në rregull nëse **b** është e barabartë me 0, ndërsa **a** nuk është 0
  + Po supozojmë se edhe **a** = 0 dhe **b** = 0?
  + Ekuacioni i mbetur duket kështu:

0 + 0 + c = 0

* + Nëse shfrytëzuesi shkruan vlerën 0 për c, atëherë çdo vlerë e **x** është zgjidhje, pra 0 + 0 + 0 gjithmonë është = 0
  + Mirëpo, supozojmë se shfrytëzuesi ka shkruar një vlerë për **c** që nuk është e barabartë me 0?
  + Tani, nuk ka zgjidhje të mundshme për **x**
  + Asnjë vlerë e **x** nuk mund ta bëjë vlerën jo-zero të **c** të bëhet e barabartë me
  + Tani pasi që kemi marrë parasysh të gjitha këto raste, çka do të thotë kjo për programimin e zgjidhjes së ekuacioneve kuadratike të bazuara në vlerat e **a**, **b** dhe **c**?
  + Ne na duhet të shkruajmë një program ashtu që kodi ynë të marrë vendim për secilin prej këtyre rasteve speciale të mundshme, para se të vendos ta kalkulojë rezultatin bazuar në formulë

***Krahasimi i të dhënave***

***dhe Deklarata “Switch”***

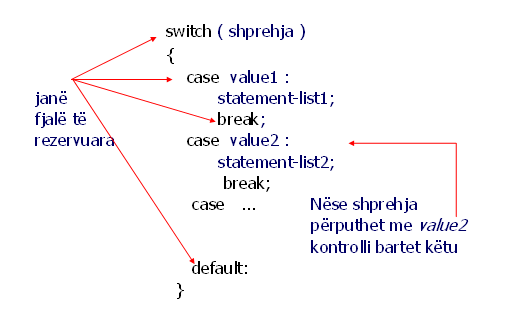
*Deklarata “Switch”*

**Deklarata “Switch”**

* + Deklarata ***switch*** ofron një mënyrë tjetër se si të vendoset cila deklaratë e ardhshme duhet të ekzekutohet dhe është e përafërt me deklaratën ***if***
  + Deklarata ***switch*** vlerëson/teston një shprehje dhe pastaj provon ta përputh rezultatin me një nga rastet e mundshme
  + Shprehja e cilat testohet mund të jetë çfarëdo që mund të interpretohet si ***int*** (byte, short, int, char).
    - Që nga versioni 1.6 edhe tipet **enum** dhe instancat e klasave (Byte, Short, Integer dhe Character).
    - Që nga versioni 1.7 edhe String
  + Çdo rast (ang. case) përmban një vlerë dhe një listë të deklaratave
  + Rrjedhshmëria e kontrollit transferohet te blloku i deklaratave të cilat përshtaten me vlerën e parë me të cilin përputhet

**Deklarata “Switch”**

* + Sintaksa e përgjithshme e deklaratës *switch* është:



**Deklarata “Switch”**

* + Shpesh deklarata “***break***” përdoret si deklarata e fundit në një bllok të deklaratave
  + Deklarata ***break*** transferon kontrollin në fund të deklaratës ***switch***
  + Në rast të mospërdorimit të deklaratës ***break***, rrjedhshmëria e kontrollit do të vazhdoj në rastin (ang. case) tjetër
  + Ndonjëherë kjo mund të jetë e përshtatshme, mirëpo shumë më shpesh ne vetëm dëshirojmë të ekzekutojmë deklaratat e shoqëruara me një rast (ang. case) të vetëm

**Deklarata “Switch”**

* + Një shembull i deklaratës switch:

switch (opsioni)

{

case ‘A’ :

aCount++ ;

break ;

case ‘B’ :

bCount++ ;

break ;

}

**Deklarata “Switch”**

Deklarata **switch** mund të ketë një rast (ang. case) opsional si vlerë automatike “***default***”, nëse asnjë rast tjetër nuk e plotëson kushtin

Rasti i nënkuptuar (ang. default case) nuk ka vlerë krahasuese por thjesht vetëm përdor fjalën e rezervuar ***default***

Nëse kemi rast të nënkuptuar (ang. default case) dhe asnjë vlerë tjetër nuk e plotëson kushtin, kontrolli i kodit transferohet në listën e deklaratave të përfshira në rastin e nënkuptuar (ang. default case)

Në rast se nuk e kemi rastin e nënkuptuar (ang. defaul case) dhe asnjë vlerë tjetër nuk e plotëson kushtin, kontrolli bartet në deklaratën pas switch-it pa e ekzekutuar asnjë deklaratë brenda switch-it

**Deklarata “Switch”**

* + Një shembull i deklaratës **switch** që përdor edhe deklaratën e nënkuptuar (ang. default case):

switch (opsioni)

{

case ‘A’ :

aCount++ ;

break ;

case ‘B’ :

bCount++ ;

break ;

default :

errorCount++ ;

break ;

}

***Ciklet ose Unazat/Blloqet përsëritëse “Loops”while, do, for***

*Deklaratat përsëritëse*

***while*** *loop*

***do while*** *loop*

***for*** *loop*

***For each*** *loop*

*Iterator – (Iteratorët)*

**Deklaratat përsëritëse**

* + Deklaratat përsëritëse mundësojnë që një deklaratë ose bllok i deklaratave të ekzekutohet disa herë
  + Shpesh kësaj dukurie i referohemi si cikël/unazë apo “loop” (shqiptimi. Llupë)
  + Sikurse deklaratat e kushtëzuara edhe loop kontrollohen nga shprehjet e tipit “boolean”
  + Java ofron disa lloje të loop:

**- while** loop

**- do while** loop

**- for** loop

Programeri duhet të zgjedhë llojin e duhur të ciklit në bazë të situatës

* + **while** loop- ka sintaksën si më poshtë:

**while ( kushti )**

**{**

***blloku i deklaratave;***

**}**

* + Nëse **kushti**është i saktë (true) blloku i **deklaratave** do të ekzekutohet
  + Pastaj kushti vlerësohet/testohet përsëri, dhe nëse është ende i saktë, blloku i deklaratave ekzekutohet përsëri
  + Përderisa kushti plotësohet deklaratat do të ekzekutohen. Momentin që kushti nuk plotësohet, loop përfundon

**while** loop

* + Një shembull i **while** loop:

**int count = 1;**

**while (count <= 10)**

**{**

**System.out.println (count) ;**

**count++;**

**}**

* + Nëse kushti i **while** loop është i pasaktë që në fillim, deklaratat nuk do të ekzekutohen fare
  + Kështu që, blloku i deklaratave brenda një **while loop** do të ekzekutohen zero ose më shumë herë
  + Disa raste të përdorimit të **while** loop:
    - Loop mund të përdoret me qëllim që të kalkuloj ndonjë shumë të caktuar
    - Loop gjithashtu mund të përdoret për vërtetimin/validimin e inputit, gjë që e bënë programin më të qëndrueshëm (robust)
  + Vlera “*sentinel”* është një vlerë e veçantë e inputit që përfaqëson përfundimin e loop

**Loop pafund (infinite)**

* + Ekzekutimi i deklaratave brenda **while** loop eventualisht do ta ndryshoj rezultatin e kushtit, pra kushti do të rezultoi në “false”
  + Nëse kjo nuk ndodh atëherë kjo njihet si loop pafund (infinit), që do të thotë se ekzekutohet deri në momentin kur shfrytëzuesi e ndërprenë programin në mënyrë të parregullt
  + Kryesisht ky është një gabim logjik i zakonshëm
  + Logjika e programit duhet të verifikohet në mënyrë që të sigurohemi se loop, eventualisht, në ndonjë kusht të veçantë do të përfundojnë

Një shembull i një loop pafund:

**int count = 1;**

**while (count <= 25)**

**{**

**System.out.println (count);**

**count = count – 1;**

**}**

* + Ky loop do të vazhdoi të ekzekutohet deri në momentin kur ndërpritet nga shfrytëzuesi në mënyrë të parregullt .

**Loop të ndërthurura**

* + Po sikur deklaratat e ndërthurura **if**, po ashtu edhe loop mund të jenë të ndërthurura
  + Që do të thotë se, deklaratat brenda loop (loop body) mund të përmbaj një loop tjetër
  + Për çdo përsëritje të loop të jashtëm, loop i brendshëm përsëritet në tërësi
  + Sa herë do të shtypet Stringu “Ketu”?

**int count1 = 1;**

**while (count1 <= 10)**

**{**

**int count2 = 1;**

**while (count2 <= 20)**

**{**

**System.out.println (“Ketu”);**

**count2++;**

**}**

**count1++;**

**} // 10 \* 20 = 200**

***Ciklet ose Unazat/Blloqet përsëritëse “Loops”***

***do while, for***

*do while loop*

*for loop*

**do while** loop

* + **do while** loop ka sintaksën si më poshtë:

**do**

**{**

**blloku i deklaratave;**

**} while ( kushti );**

* + Më parë ekzekutohet përmbajtja e **do while** loop, e pastaj testohet **kushti**
  + Blloku i deklaratave ekzekutohet përderisa kushti plotësohet, njëjtë si në **while** loop
  + Një shembull i **do while** loop:

**int count = 1;**

**do**

**{**

**count++;**

**System.out.println (count);**

**} while (count <= 10);**

* + Përmbajtja e **do while** loop ekzekutohet një ose më shumë herë (Vërej që përmbajtja ekzekutohet së paku një herë)
  + Shiko shembullin ReverseNumber.java (më poshtë)

**Shembulli** ReverseNumber.java

import java.util.Scanner;

public class ReverseNumber{

public static void main (String[] args) {

int number, cpNumber,lastDigit, reverse = 0;

Scanner scan = new Scanner (System.in);

System.out.print ("Shtyp nje numer te plote pozitiv: ");

number = scan.nextInt();

cpNumber = number;

do

{

lastDigit = number % 10;

reverse = (reverse \* 10) + lastDigit;

number = number / 10;

} while (number > 0);

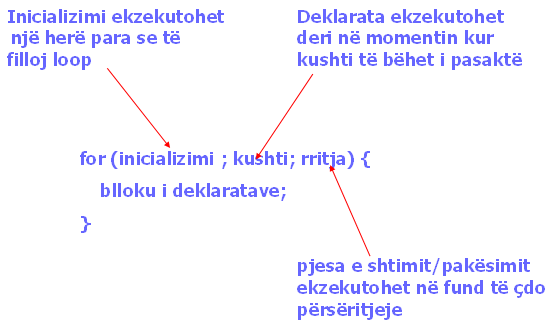
System.out.println (cpNumber + " i kthyer mbrapsht eshte " + reverse);

}

}

**for** loop

**for** loop ka këtë sintaksë:



**for** loop

* + **for** loop është me funksionalitet i barabartë me strukturën e **while** loop, si në vijim:

**inicializimi;**

**while ( kushti )**

**{**

**blloku i deklaratave;**

**shtimi; // (increment)**

**}**

* + Një shembull i **for loop**:

**for (int count=1; count <= 5; count++){**

**System.out.println (count);**

**}**

* + Pjesa e inicalizimit mund të përdoret edhe për deklarimin e variablës
  + Sikur se tek **while** loop, kushti **for** loop testohet para se të ekzekutohet loop
  + Kështu që, përmbajtja e **for** loop do të ekzekutohet zero ose më shumë herëNë pjesën e shtimit (increment) mund të kryhet çfarëdo kalkulimi:

**for (int num=100; num > 0; num -= 5)**

**{**

**System.out.println (num) ;**

**}**

* + Një **for** loop është i përshtatshëm të përdoret kur e di paraprakisht sa herë do të ekzekutohet përmbajtja e loop
  + Shiko Multiples.java (më poshtë)

Shiko Stars.java (më poshtë)

**Shembulli** Mulitples.java

import java.util.Scanner;

public class Multiples {

public static void main (String[] args) {

final int PER\_LINE = 5;

int count = 0;

Scanner scan = new Scanner (System.in);

System.out.print ("Shtyp nje numer pozitiv: ");

int value = Math.abs(scan.nextInt());

System.out.print ("Shtyp limitin e eperm: ");

int limit = Math.abs(scan.nextInt());

System.out.println ("\nShumefishet e " + value + " mes " +value + " dhe " + limit + " (inkluzive) jane:");

for (int mult = value; mult <= limit; mult += value) {

System.out.print (mult + "\t");

count++;

if (count % PER\_LINE == 0)

System.out.println();

}

}

**}**

**Shembulli** Stars.java

public class Stars{

public static void main (String[] args){

final int MAX\_ROWS = 10;

for (int row = 1; row <= MAX\_ROWS; row++){

for (int star = 1; star <= row; star++)

System.out.print ("\*");

System.out.println();

}

}

}

**for** loop

* + Secila shprehje (pjesë) në **for** loop është opsionale
  + Nëse inicalizimi lihet jashtë, inicializimi nuk do të kryhet fare
  + Nëse kushti lihet jashtë, konsiderohet që gjithmonë kushti është i saktë, kështu që krijohet një loop pafund
  + Nëse shtimi (increment) lihet jashtë, asnjë veprimi i shtimit nuk do të kryhet
  + “Për gjithmonë” mund të shkruhet si:

**for (;;) {**

**blloku i deklaratave;**

**}**

***Vargjet (Arrays)***

***Vargjet janë objet që ndihmojnë në organizimin e informatave të shumta***

*Në këtë mësim do të fokusohemi në:*

*Deklarimin dhe përdorimin e vargjeve*

*Dërgimi i vargjeve dhe elementeve të vargjeve si parametër*

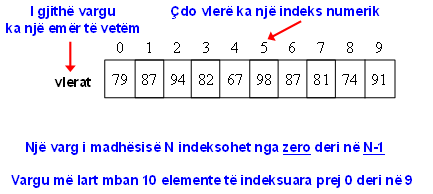
*Vargjet e objekteve*

*Vargjet e objekteve në një varg*

*Vargjet dy-dimensionale*

**Vargjet**

Një varg është një listë e elementeve



Një elementi t veçantë n vargë i qasemi duke përdourur emrin e

vargut së bashku me indeksin brenda kllapave të mesme

* + Përshemull, shprehja më poshtë

***vlerat[2]***

* + i referohet vlerës 94 (vlerës së tretë me radhë në varg)

Shprehja më lart, në rastin tonë përfaqson një vent ku mund të ruhet dhe t`i qasemi një vlere të llojit integjer

Për shembull, një element në varg mund ti jepet një vlerë të shtypet apo të përdoret në ndonjë kalkulim:

***vlerat [2] = 89;***

***vlerat [param] = vlerat [param] + 2;***

***mesatarja = (vlerat [0] + vlerat [1]/2;***

***System.out.println("max = " + vlerat[5]);***

Vlerat e ruajtura në **varg** njihen si elementet e vargut

Në një **varg** mund të ruhen vetëm vlera të të njëjtit lloj ***(të njohur si tipi i elementit - element type)***

Tipi i elementit mund të jetë tip i thjeshtv *(primitiv)* apo një referencë në objekt

Prandaj, mund të krijomë një varg të integjerëve, apo një varg të char, apo një varg të objekteve të llojit **String**, etj.

**Në Java, vet vargu është objekt**

Pra emri i vargut është variabël që referencon një objekt, prandaj edhe vet vargu duhet të krijohet si objekt

**Deklarimi i vargjeve**

Vargu vlerat mund të deklarahohet si :

**int[] vlerat = new int [10]**

Lloji i variablës vlerat është *int [] (varg i integjerëve)*

Veni re lloji i vargut nuk e specifikon madhësinë e tij, por çdp objekt e ka madhësinë e vet

Variabla vlerat në rastin tonë është një varg i ri i cili munt t`i ruaj (mbaj) 10 integjerë

See **VarguBazik.java**

Disa shembuj të deklarimit të vargjeve

*double[] cmimet = new double [500];*

*boolean[] opionet = null;*

*opsionet = new boolean [20];*

*char[] kodet = new char[65536];*

*System.out.print(kodet[100]);*

**Testimi i kufijve**

* Në momentin që krijohet vargu, i përcaktohet madhësia dhe ajo është fikse (epandryshueshme)
* Gjatë përdorimit të indeksit duhet të kihet shumë kujdes që të specifikohet pozita e vlefshme e elementit
* Vlera e indeksit duhet të jetë brenda kufijve **(0 dhe N-1)** ku N është madhësia e vargut
* Nëse specifikohet ndonjë element jashtë limitit të lartpërmendur, atëher java e lajmëron gabimin përmes
* ArrayIndezOutofBounds Excption
* kjo njihet si testim automatik i kufijve.
* P.sh. *Nëse vargu kodet, shih më poshtë mund të mbaj 100 vlera, atëherë vargu indeksohet duke përdorë vlerat 0 deri 99*
* Nëse variabla count ka vlerën 100, atëherë shprehja më poshtë do të shkaktoj problemin (Exception) e përmendur më parë:

**System.out.println(kodet[count]);**

* Është gabim i shpeshtë të tejkalohet indeksi i vargut për një . Në ang. ky gabim njihet si off-by-one.

***for (int indeks=0; indeks <=(gabim)100; indeks++)***

***kodet[indeks] = indeks\*5****;*

* Cdo varg, që në Java është, ka një konstante publike me emrin length ku ruhet madhësia e vargut

I qasemi përmes emrit të vargut, p.sh.:

**vlerat.length**

* Veni re që length mban numrin e elementeve, jo indeksin më të madh

Shih **ReverseOrder.java**

Shih **ShkronjatCount.java**

**Lista inicializuese**

* Lista inicializuese mund të përdoret për të krijuar dhe inicializuar vargun me një hap të vetëm
* Vlerat përfshihen në kllapa gjarpërore dhe ndahen me presje

Shembull :

***int [] numrat = { 147, 323, 89, 933, 540, 269. 97};***

***char [] notat = {'A', 'B', 'C', 'D', 'F'};***

Veni re që kur përdorim listat inicializuese :

**- Nuk përdoret operatori new**

**- Nuk specifikohet madhësia e vargut**

* Madhësia e vargut përcaktohet nga numri i elementeve në listën inicializuese, ndërsa indeksin e përcakton renditja e elementit
* Lista inicializuese mund të përdoret vetem gjatë deklarimit të vargut

Shih **NumratEThjesht.java**

**Vargjet si parametra**

* I gjthë vargu mund t`i dërgohet si parametër një metodës
* Si me çdo objekt tjetër dërgohet referenca e vargut, që do të thotë se parametrat aktual dhe ata formal janë "pseudonm" i njëra tjetrës.
* Ndërrimi i ndonjë elementi të vargut brenda metodës e ndërron edhe elemetin korrespounues në vargun origjinal
* Edhe elementi i vargut mund t`i dërgohet si parametër një metode, dhe ndiqen rregullat e llojit të aij elementi
* Vargjet e objekteve
* Elementet e vargjeve mund të jenë edhe referenca të objekteve
* Deklarata më poshtë rezervon hapësirë në memorie për të ruajtur 25 referenca të tipit String

**String[] fjalet = new String [25];**

Veni re, objektet e tipit String NUK krijohen me deklaratën më lartë

Cdo objekt që ruhet në varg duhet të krijohet ndarazi me deklaratën më lartë

Cdo objekt që ruhet në varg duhet të krijohet ndarazi

Shih VarguString.java

**Argumentet në CLI**

Nga sintaksa e metodës main shohim se si parametër pronon një varg të objekteve String

Vlerat vijnë si argumente nga CL (Command Line) dhe jepen kur të thirret interpreteri (java)

P.sh., gjatë thirrjes së interpreterit, metodës main, i dërgohet një varg me dy objekte të tipit String :

**> java Pershendetje Final Fisteku**

Këto objekte të String ruhen në indekset 0-1 të parametrit args

Shih Pershendetje.java

**Vargjet e objekteve**

Objekte të ndryshme mund të kenë vargje si variabla të instancës

Struktura të shumta dhe të dobishme mund të krijohen duke përdorur vargjet dhe objektet

Zhvilluesit e softuerëve duhet të vendosim me kujdes në zgjedhjen e të dhënave dhe objekteve që të zgjidhen me kujdes në zgjidhjen e të dhënave dhe objekteve që të zgjidhjet ndonjë problem i caktuar

Shih **Studenti.java**

Shih **Fakulteti.java**

Shih **Universiteti.java**

Kërkimi në varg

Një veprim i zakonëshem kur kemi të bëjmë me vargjet, është kërkimi i ndonjë elementi të veçantë në varg

Kërkimi linear teston çdo element në varg derisa të gjindet elementi i kërkuar

Kërkim binar është më efikas se ai linear por mund të zbatohet në lista të renditura

Kërkimi binar teston gjithmonë elementin e mesvm dhe vazhdon kërkimin majtas nëse elementi i kërkuar është më i madh

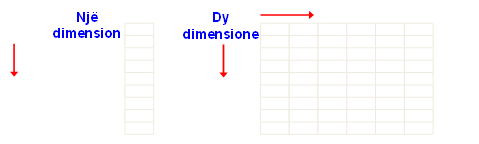
Ky proces përsëritet deri sa të gjendet elementi i kërkuar, normal nëse elementi i kërkuar gjendet në varg

Shih **Kerkimi.java**

**Vargjet Dy-Dimensionale**

Vargjet një-dimensionale ruajnë një listë të elementeve

Vargjet dy-dimensionale mund të shihen si tabela të elementeve, me rreshta dhe shtylla edhe pse në realitet janë listë e elementeve, ku çdo element i listës është përsëri listë e elementeve



Më saktësisht, një varg dy-dimensional në Java njihet si varg i vargjeve

Një varg dy-dimensional deklarohet duke specifikuar ndaras, madhësinë e seclit dimension (Specifikimi i dimensionit të dytë nuk është obligative)

**int[] [] vlerat = new int [12] [50];**

Një varg dy-dimensional referencohet duke përdorur dy vlera të indekseve

**vlerat = vlerat [3] [6]**

Vargu i ruajtur në një rresht specifikohet duke përdorur një indeks

