PROGRAMSKI JEZIK JAVA

Uvod

Programski jezik Java

1. Java: platforma za izvršavanje programa
2. Java: programski jezik

Java kao platforma

- dizajniran da što manje zavisi od specifičnih karakteristika konkretnog računarskog sistema
- jednom napisan i *preveden* program se izvršava na bilo kojoj platformi koja podržava Javu

Java kao platforma

- interpretirani jezik
  - just in time compiler
- bajt-kod
  - specifikacija je dostupna – više implementacija kompajlera
- Java virtuelna mašina (JVM)
  - specifikacija je dostupna – više implementacija JVM

Java kao programski jezik

- jezik opšte namene
- konkurentno, objektno-orijentisano programiranje
- literatura
  - Referentna dokumentacija: JavaSoft homepage http://java.sun.com
  - Knjige:
    Milosavljević, Vidaković: *Java i Internet programiranje*
    Bruce Eckel: *Thinking in Java*,
    http://www.bruceeckel.com

Java program

- Sastoji se iz .java fajlova
- Jedan .java fajl predstavlja jednu klasu
  - Klasa je osnovni entitet nad kojim se radi u objektnom programiranju
  - Klasa sadrži podatke i kod koji definiše operacije nad tim podacima
- Svaki .java fajl sadrži
  1. deklaraciju paketa u kojem se klasa nalazi
     - klase se organizuju u pakete zbog jedinstvene identifikacije i zbog grupisanja po srodnosti
  2. import sekciju
     - spisak drugih klasa čijim funkcionalnostima se pristupa iz klase
  3. Programski kod klase
- Ovakav redosled je obavezan!

Izvršavanje programa

- Izvršavanje Java programa kreće od bloka koda definisanog u okviru metode

```
public static void main(String args[])
```

- U ovom kursu metodu možemo posmatrati kao funkciju koja
  - sadrži blok koda koji obavlja određene operacije
  - prima parametre na osnovu koje koristi pri obavljanju operacija
  - vraća rezultat operacije
    - ako nije tipa void

Izvršavanje programa

- metoda **`main()`**

`Hello.java`

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

Prevođenje i pokretanje

- prevođenje:

```
javac Hello.java
```

- pokretanje:

```
java Hello
```

[ovo važi sa standardni razvojni paket JDK (Java Development Kit)]

Izrazi

- Naredba u programskom kodu
- Obično se završavaju znakom ;
- Programski blok čini više izraza ograničenih vitičastim zagradama { }
- Ako se u bloku nalazi samo jedan izraz, ne moraju zagrade

Promenljive

- Promenljive koristimo za čuvanje vrednosti kojima program operiše u toku rada
- Zavisno od vrednosti, promenljiva ima tip
- Definišu se kao

  tip identifier

- Npr int broj;

Dodela vrednosti promenljivoj

- U promenljivu se postavlja naznačena vrednost
- Sintaksa

  naziv_promenljive = vrednost_promenljive;
- Primer

  broj = 10;
- Može i istovremeno deklaracija i inicijalizacija

  int broj = 10;
- Promenljiva se može deklarisati (i inicijalizovati) bilo gde u kodu
- Promenljiva postoji od deklaracije do kraja bloka koda u kojem je definisana!!!

Identifikatori

- Nazivi dodeljeni entitetima u programu
  - promenljivima, klasama, metodama, ...
- Identifikator može da sadrži
  - Velika i mala slova, cifre, znak _ (underscore), znak $
- Na početku ne sme biti cifra
- Ne sme biti razmak niti bilo koji *whitespace* karakter
- Ne smeju se koristiti Java rezervisane reči (*main, int, for, class, ...*)
- *Case sensitive*
  - različito se tretiraju velika i mala slova
  - identifikatori racunar i Racunar su različiti identifikatori

Tip promenljive

- Primitivni tipovi
  - koriste se za skladištenje jedne vrednosti koja je broj, slovo ili ima logičku vrednost (tačno/netačno)
- Složeni tipovi (objekti)
  - objedinjuju više podataka i operacije nad tim podacima
  - za sada najvažniji ovakav tip je String
    - predstavlja niz karaktera

Osnovni koncepti

- primitivni tipovi podataka

Primitivni tip	Veličina	Minimum	Maksimum
boolean	1-bit	-	-
char	16-bit	Unicode 0	Unicode 2^{16}- 1
byte	8-bit	-128	+127
short	16-bit	-2^{15}	$+2^{15} - 1$
int	32-bit	-2^{31}	$+2^{31} - 1$
long	64-bit	-2^{63}	$+2^{63} - 1$
float	32-bit	IEEE754	IEEE754
double	64-bit	IEEE754	IEEE754
void	-	-	-

Komentari

- Jednolinijski

  //

- Višelinijski

  /*

  */

- JavaDoc komentari
  - iz njihovog sadržaja se programski može generisati dokumentacija o programskom kodu

  /**

  */

Operatori

- aritmetički
- relacioni
- logički
- operator dodele

Aritmetički operatori

- Osnovne operacije:

+, -, *, /, %

- Umesto x = x + 5

x += 5

- Automatski inkrement:
  - ++x isto kao x+=1
  - x++ isto kao x+=1
  - Razlika:
    - rezultat izraza ++x je nova vrednost x i posledica je uvećan x za 1
    - rezultat izraza x++ je stara vrednost x, a posledica je uvećan x za 1
  - Slično, postoji i
    - x–
    - --x
- % - ostatak pri celobrojnom deljenju
  - 7%2 = 1 (jer je rezultat 3 i ostatak 1)

Aritmetički operatori

y = 5;

Operator	Rezultat
x=y+2	X← 7
x=y-2	X← 3
x=y%2	X← 1
x=++y	X←6, y←6
x=y++	X←5, y←6
x=--y	X←4, y←4

Aritmetički operatori

x = 10;

y = 5;

Operator	Isto kao	Rezultat
x=y		x← 5
x+=y	x=x+y	x← 15
x-=y	x=x-y	x← 5
x*=y	x=x*y	x← 50
x/=y	x=x/y	x← 2
x%=y	x=x%y	x← 0

Relacioni i logički operatori

- Relacioni: < > <= >= == !=
- Logički: && (I), || (ILI), ! (NE)

Relacioni operatori

x = 5;

Operator	Rezultat
==	x == 8 je netačno (false)
!=	x != 8 je tačno (true)
>	x > 8 je netačno (false)
<	x < 8 je tačno (true)
>=	x >= 8 je netačno (false)
<=	x <= 8 je tačno (true)

Logički operatori

- Logički: **&& || !**
- Rezultat logičkih operatora je tačno (true) ili netačno (false)
- Operandi logičkih operatora su logički izrazi

&&	false	true
false	false	false
true	false	true

||	false	true
false	false	true
true	true	true

!	
false	true
true	false

Logički operatori

x = 6;

y = 3;

Operator	Objašnjenje	Primer
&&	konjukcija (and, i)	(x < 10 && y > 1) tačno (true)
||	disjunkcija (or, ili)	(x==5 || y==5) netačno (false)
!	negacija (not, ne)	!(x==y) tačno (true)

Operator dodele

- Ako su operandi primitivni tipovi, kopira se sadržaj:

```
int i = 3, j = 6;
i = j;  // u i ubačeno 6
```

- Ako su operandi reference, kopira se sadržaj reference, a ne kompletni objekti na koje ukazuju!

Implicitna konverzija tipova

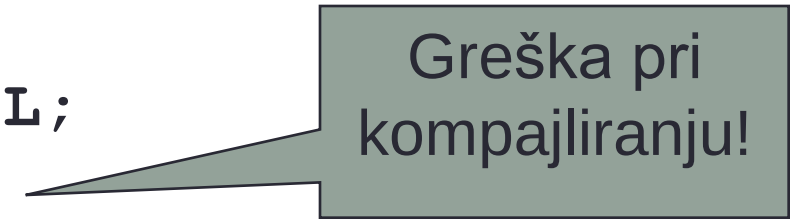
- Sa “užeg” ili “manjeg” tipa na “širi” ili “veći” tip.
- Nema gubitka informacije jer “uži” tip podatka staje u “širi” tip podatka.
- Primer:

```
long a;
int i = 5;
a = i;
```

Eksplicitna konverzija tipova

- Sa “šireg” na “uži” tip podatka – posledica je gubljenje informacije.
- Primer:

```
long a = 5L;
int b = a;
```



Eksplicitna konverzija tipova

- Pravilna eksplicitna konverzija – upotreba **cast** operatora:
- Primer:

```
long a = 5L;
int b = (int)a;
```

Kontrola toka

- `if else`
- `switch`
- `for`
- `while`
- `do while`
- `break`
- `continue`

if

```
if (uslov)
  akcija
```

```
if (uslov)
  akcija
else
  druga_akcija
```

if else

```
int result = 0;
if(testval > target)
   result = -1;
else if(testval < target)
   result = +1;
else
   result = 0; // match
```

```
if (bodovi >= 95)
    ocena = 10;
else if (bodovi >= 85)
   ocena = 9;
else if (bodovi >= 75)
   ocena = 8;
else if (bodovi >= 65)
   ocena = 7;
else if (bodovi >= 55)
   ocena = 6;
else
   ocena = 5;
```

Uslovni operator

```
a = i < 10 ? i * 100 : i * 10;
```

- isto kao:

```
if (i < 10)
      a = i * 100;
else
     a = i * 10;
```

switch

- Izraz u switch() izrazu mora da proizvede celobrojnu vrednost.
- Ako ne proizvodi celobrojnu vrednost, ne može da se koristi switch,() već if()!
- Ako se izostavi break; propašće u sledeći case:
- Kod default: izraza ne mora break; - to se podrazumeva.

switch

```
switch(c) {
  case 'a':
  case 'e':
  case 'i':
  case 'o':
  case 'u':
    System.out.println("samoglasnik");
    break;
default:
    System.out.println("suglasnik");
}
```

varijanta 1, bez switch

```
if (ocena == 5)
  System.out.println(“odlican”);
else if (ocena == 4)
  System.out.println(“vrlo dobar”);
else if (ocena == 3)
  System.out.println(“dobar”);
else if (ocena == 2)
  System.out.println(“dovoljan”);
else if (ocena == 1)
  System.out.println(“nedovoljan”);
else
  System.out.println(“nepostojeca ocena”);
```

Varijanta 2, switch

```
switch(ocena) {
        case 5: System.out.println(“odlican”);
                break;
        case 4: System.out.println(“vrlo dobar”);
                break;
        case 3: System.out.println(“dobar”);
                break;
        case 2: System.out.println(“dovoljan”);
                break;
        case 1: System.out.println(“nedovoljan”);
                break;
        default: System.out.println(“nepostojeca ocena”);
}
```

for

- Za organizaciju petlji kod kojih se unapred zna koliko puta će se izvršiti telo ciklusa.
- Petlja sa početnom vrednošću, uslovom za kraj i blokom za korekciju.
- Opšta sintaksa:

```
for (inicijalizacija; uslov; korekcija)
   telo
```

for

```
for (int i = 0; i < 10; i++)
  System.out.println(i);
for (int i = 10; i >= 1; i--)
  System.out.println(i);
```

while

- Za cikličnu strukturu kod koje se samo zna uslov za prekid.
- Telo ciklusa ne mora ni jednom da se izvrši
- Opšta sintaksa:

```
while (uslov)
   telo
```

- Važno: izlaz iz petlje na false!

while

```
int i = 0;
while (i <= 10) {
   System.out.println("Trenutno je " + i);
   i=i+1;
}
```

Primer02-03

do while

- Za cikličnu strukturu kod koje se samo zna uslov za prekid
- Razlika u odnosu na while petlju je u tome što se telo ciklusa izvršava makar jednom.
- Opšta sintaksa:

```
do
  telo
while (uslov);
```

- Važno: izlaz iz petlje na false!

do while

```
int i = 0;
do {
   System.out.println(i++);
} while (i < 10);
```

break i continue

- break – prekida telo tekuće ciklične strukture (ili case: dela) i izlazi iz nje.
- continue – prekida telo tekuce ciklične strukture i otpočinje sledeću iteraciju petlje.

break i continue

```
for(int i = 0; i < 10; i++) {
   if (i==7) {
    break;
   }
   if (i == 2)
     continue;

   System.out.println("Broj je:" + i);
}
```

Primer02-04

break i continue

```
int i = 0;
System.out.println("Broj je:" + i);
while (i++ <= 9) {
   if (i == 7) {
     break;
   }
   if (i == 2)
     continue;
   System.out.println("Broj je:" + i);
}
```

Izlaz iz ugnježdene petlje

```
for (...)
{
   for (...)
   {
      ...
      if (uslov)
        break;
   }
}
```