Osnove Jave

Sadržaj

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

- Jedan .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)
```

- Primer programa

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

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

Komentari

- Jednolinijski
  //
- Višelinijski
  /*
  */
- JavaDoc komentari
  - iz njihovog sadržaja se programski može generisati dokumentacija o programskom kodu

  /**
  */

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	–	–	–

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

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

Enumeracije

- Nabrojivi tipovi podataka (celobrojni)
- Primer:

```
enum Size {SMALL, MEDIUM, LARGE, EXTRA_LARGE};
Size s = Size.MEDIUM;
enum Days {MON, TUE, WEN, THU, FRI, SAT, SUN};
Days d = Days.MON;
```

Operatori

- aritmetički operatori
- relacioni i logički
- bit-operatori
- 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)

Relacioni i logički operatori

- Relacioni: < > <= >= == !=
- Logički: && (I), ||(ILI), ! (NE)
- Rezultat logičkih operatora je tačno (true) ili netačno (false)
- Operandi logičkih operatora su logički izrazi

A	B	A&&B
false	false	false
false	true	false
true	false	false
true	true	true

A	B	A||B
false	false	false
false	true	true
true	false	true
true	true	true

A	!A
false	true
true	false

Bit operatori

- Logičko I nad bitovima: &
- Logičko ILI nad bitovima: |
- Ekskluzivno ILI (XOR) nad bitovima: ^
- Logička negacija nad bitovima –unarni operator: ~
- Kombinacija sa = :

  `&= |= ^=`

A	B	A^B
false	false	false
false	true	true
true	false	true
true	true	false

Bit operatori

- Shift-ovanje (pomeranje):
  - a>>b – pomera bitove u a za b mesta
    - ako je a pozitivan, ubacuje 0
    - ako je a negativan, ubacuje 1
  - a<<b – pomera bitove u levo i ubacuje 0
  - a>>>b – pomera bitove u a u desno za b mesta i ubacuje 0 bez obzira na znak a.
- Rezultat pomeranja je 32-bitan, osim ako promenljiva koja prihvata rezultat nije long (tada je 64-bitan)!

Operator dodele

- Ako su operandi primitivni tipovi, kopira se sadržaj:

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

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

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; // kada su vrednosti iste
```

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 (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");
}
```

Ciklusi

- Izvršavanje istog koda ciklično
- Kod se izvršava dok god je ispunjen određeni uslov
- Tri tipa ciklusa u Javi
  - for
  - while
  - do-while

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);
```

- može i višestruka inicijalizacija i step-statement:

```
for(int i = 0, j = 1; i < 10 && j != 11;i++,
    j++)
```

- oprez (može da se ne završi):

```
for (double x = 0; x != 10; x+=0.1) ...
```

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;
}
```

- Važno: izlaz iz petlje na false!

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);
```

- Važno: izlaz iz petlje na false!

break i continue

- break – prekida telo tekuće ciklične strukture (ili case: dela) i izlazi iz nje.
- continue – prekida telo tekuće 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);
}
```

Izlaz iz ugnježdene petlje

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

Nizovi

```
int a[]; // jos uvek nije napravljen niz!
a = new int[5]; // niz od 5 nula ili

int a[] = new int[5]; // niz od 5 nula ili

int a[] = { 1, 2, 3, 4, 5 };
```

Iteriranje kroz nizove

- Niz ima definisan atribut **length** koji čuva broj elemenata u nizu
- Klasična for petlja:

```
int niz[] = {1, 2, 3, 4};
for (int i = 0; i < niz.length; i++)
    System.out.println(niz[i]);
```

- for each petlja omogućuje prolazak kroz niz ili kolekciju.
- Opšta sintaksa:

```
for (promenljiva : niz) {
   ... // telo
}
```

- Primer:

```
for (int el : niz) {
    System.out.println(el);
}
```

Višedimenzionalni nizovi (matrice)

```
int a[][] = { {1, 2, 3 }, {4, 5, 6 } };

int a[][] = new int[2][3];

int a[][] = new int[2][];
for(int i = 0; i < a.length; i++)
{
    a[i] = new int[3];
}
```

- Moguće napraviti dvodimenzionalni niz koji ima različit broj kolona u svakoj vrsti:

```
int[][] aa = {
{0},
{10, 11},
{20,21, 22}
};
```

Iteriranje kroz višedimenzionalni niz

```
int[][] a = { {1, 2, 3}, {4, 5, 6} };
for (int i = 0; i < a.length; i++) {
  for (int j = 0; j < a[i].length; j++) {
    System.out.println(a[i][j]);
  }
  System.out.println();
}
```