

Uvod u programiranje dr Ninoslava Savić

Potprogrami Metodi

Potprogram

- Potprogram predstavlja imenovani segment koda koji se može pokrenuti u drugom delu koda.
- Potprogram sadrži grupu naredbi koje obično čine jednu logičku (funkcionalnu celinu) u programu.
- Potreba za kreiranjem potprograma koji se mogu pozivati na raznim mestima u glavnom programu nastaje:
 - primenom metodologije modularnog i top-down projektovanja programa,
 - kada se grupa naredbi ponavlja više puta u programu, ima smisla napraviti od nje potprogram.

2

Metod

- Metod predstavlja implementaciju koncepta potprograma iz klasičnih programskih jezika u Javi.
- Metod preuzima neke parametre, izvršava neke akcije ili izračunavanja i vraća neku povratnu vrednost (opciono).
- Definicija metoda piše se u deklarativnom delu programa.
- Definicija metoda pojavljuje se samo jednom u programu, dok se
 - poziv na izvršavanje metoda može pojaviti više puta u programu, kada god je potrebno izvršavanje grupe naredbi tj. funkcionalnosti koja je definisana metodom.

3

Deklaracija metoda

- Deklaracija metoda:

```
<specifikator pristupa> <static/final> <povratni tip>
<naziv metoda> <lista argumenata>
{
    telo metoda;
}
```

- Primer - main metod

```
public static void main(String args[]){
    //telo metoda
}
```

4

Struktura metoda - zaglavlje

1. Zaglavlje metoda sadrži:

- obavezno:
 - **povratni tip** – tip vrednosti koju metod vraća kao svoj rezultat ili **void** ako ne vraća nikakvu vrednost
 - **naziv metoda** – identifikator (ime koje treba da ukazuje na funkciju metoda)
 - **listu parametara** - u obliku zagrada, razdvojeni zarezom, predstavljeni tipom i imenom.
 - ako metod nema parametara zagrade su prazne.
- opciono:
 - **specifikatori pristupa** (*public/private/protected*)
 - **specifikator *static***
 - **specifikator *final***
 - **listu izuzetaka** (izuzeci su Java runtime greške)

5

Struktura metoda - telo

2. Telo metoda

- U vitičastim zgradama
- Sadrži:
 - deklaracije lokalnih promenljivih (opciono),
 - izvršni kod metoda,
 - **return** instrukciju (opciono, tj. samo ako povratni tip nije *void*)

6

Zaglavlje metoda

Zaglavlje metoda sadrži:

- specifikatore pristupa (vidljivosti metoda),
- tip promenjive (ili objekta) koju metod vraća kao povratnu vrednost ili **void**,
- naziv metoda,
- listu fiktivnih argumenata u obliku zagrada
 - lista sadrži **tip i naziv svakog argumenta**,
 - lista može biti i prazna ako nema argumenata,
- izuzetke koje metod obrađuje

Primeri zaglavlja metoda:

```
public float površinaKvadrata(float stranica)
public boolean postavi(int i, int j, String ime)
    throws IndexOutOfBoundsException
```

Zaglavlje metoda

Primer:

```
void prikazi() { // zaglavlje metoda bez
                argumenata
    .....      // telo metoda
}
```

Obavezni delovi u zaglavlju metoda su:

- povratni tip (**svi tipovi podataka ili void**)
- naziv metoda
- lista fiktivnih argumenata ili () ako nema argumenata

8

Deklaracija metoda - primeri

```

/* Metod koji računa obim kvadrata na osnovu
   zadate stranice */

// prvi način
public static float obimKvadrata1(float stranica){
    float obim;           // lokalna promenljiva
    obim = 4 * stranica;
    return (obim) ;
}

// drugi način
public static float obimKvadrata2(float stranica){
    return ( 4 * stranica );
}

```

9

Poziv metoda na izvršavanje

- Instrukcija za poziv metoda na izvršavanje:
naziv metoda (lista **stvarnih** argumenata);
- Lista sadrži imena (ili vrednosti) stvarnih argumenata, odvojene zarezima
- Lista stvarnih argumenata mora odgovarati listi fiktivnih argumenata iz zaglavlja deklaracije metoda po:
 - broju argumenata,
 - tipu argumenata i
 - redosledu njihovog navođenja.
- Pozivi metoda se mogu pojaviti:
 - isključivo kao samostalne instrukcije u Java programu (ako je povratni tip *void*) ili
 - u okviru izraza ili naredbe (za sve povratne tipove osim *void*)

Povratak iz metoda – return instrukcija

- Return se MORA pojaviti u telu svakog metoda čiji je povratni tip različit od void
- Sintaksa:
- ```
return izraz; ili return (izraz);
```
- Vrednost uz return je u opštem slučaju izraz (konstanta ili promenljiva kao specijalni slučaj izraza)
  - Ta vrednost MORA biti istog tipa kao povratni tip iz definicije metoda.

### Semantika:

#### Return izaziva:

- dodelu vrednosti izraza uz return imenu metoda
- bezuslovni prekid metoda i prelazak na izvršavanje sledeće instrukcije u glavnom programu posle poziva metoda.

11

## Metodi - primer

- Definisati metod za množenje 2 cela broja.
- Pozvati metod da bi se izračunao proizvod dva zadata broja.

```

// definicija metoda pomnozi
public static int pomnozi (int x, int y)
{
 return x * y;
}

// deo koda u kome se poziva metod
int a = 99;
int b = 57;
....
int proizvod = pomnozi (a, b);
/* poziv metoda u okviru izraza */
....

```

12

## Primer metoda – a na stepen b

- a) Definirati metod za izračunavanje a na stepen b.  
b) Pozvati metod da izračuna 2 na stepen 4 i prikazati rezultat.

```
// definisanje metoda stepen
public static int stepen (int a, int b) {
 int proizvod = 1;
 for (int i = 0; i < b; i++)
 proizvod = proizvod * a;
 return proizvod;
}

// negde u main metodu
int broj = 2;
int eksponent = 4;
int rezultat;
rezultat = stepen (broj, eksponent); // poziv metoda
System.out.println
(broj + " na stepen " + eksponent + " = " + rezultat);
```

## Greške pri upotrebi metoda

- Greska 1 – pri definiciji metoda  
neusaglašenost povratnog tipa i izraza (promenljive) uz return  
static **boolean** vratiDuzinu() {  
 int len = 10;  
 return (len);  
}
- Greska 2 – pri pozivu metoda na izvršenje  
neusaglašenost tipa metoda u naredbi dodele  
static **int** vratiDuz() {  
 return duzina;  
}  
.....  
float r = vratiDuz();

14

## Primer - Metod za sabiranje

### Primer:

- Deklarisati metod koji sabira dva cela broja.
- U glavnom programu pozvati metod da izvrši sabiranje broja 5 i 3, zatim ispisati rezultat.

```
// deklaracija metoda saberi

public static int saberi(int broj1, int broj2) {
 int suma;
 suma = broj1 + broj2;
 return suma;
}
```

15

## Primer - Metod za sabiranje

- Kompletno rešenje primera

```
public class ZbirMetod{
 // deklaracija metoda saberi
 public static int saberi(int broj1, int broj2){
 int suma;
 suma = broj1 + broj2;
 return suma;
 }
 // glavni program
 public static void main (String args[]){
 int x = 5;
 int y = 3;
 int z = saberi(x,y); // poziv metoda
 System.out.println(x + " + " + y + " = " + z);
 }
} // na izlazu će pisati 5 + 3 = 8
```

16

## Zadatak

Dat je metod `proba(int x, int y)`:

```
public static int proba(int x, int y){
 x = x + 10;
 y = y + 10;
 return x;
}
```

Koje su vrednosti promenljivih `p`, `q` i `r` nakon izvođenja narednog koda?

```
int p = 1;
int q = 2;
int r = 5;
r = proba(p, q);
```

17

## Rešenje

Dat je metod `proba(int x, int y)`:

```
public static int proba(int x, int y){
 x = x + 10;
 y = y + 10;
 return x;
}
```

Koje su vrednosti promenljivih `p`, `q` i `r` nakon izvođenja narednog koda?

```
int p = 1;
int q = 2;
int r = 5;
r = proba(p, q);
```

1, 2, 11

18

## Primer - Definicija metoda `maximum`

Zadatak:

- Definisati metod koji vraća veći od dva zadata realna broja

```
public static double maximum(double br1, double br2) {
 double rezultat;
 if (br1 > br2)
 rezultat = br1;
 else rezultat = br2;
 return rezultat;
}
```

19

## Primer – Poziv metoda `maximum`

```
public class Test {
 public static double maximum(double br1, double br2) {
 double rezultat;
 if (br1 > br2)
 rezultat = br1;
 else rezultat = br2;
 return rezultat;
 }
 public static void main(String [] args) {
 double a = 4.00001;
 double b = 2.00002;
 double c = maximum(a, b); // poziv metoda
 System.out.println
 ("Veći od brojeva " + a + " i " + b + " je " + c);
 }
}
```

20

## Primer metoda – Površina pravougaonika

```
/* Definicija i poziv metoda za računanje
površine pravougaonika */
public class Povrsina {
 public static void main(String[] args) {
 int duz = 10;
 int sirina = 5;
 int povrsina = izracunajPovrsinu(duz, sirina);
 // poziv metoda
 System.out.println(povrsina);
 }
 // definicija metoda
 public static int izracunajPovrsinu(int a, int b) {
 int p = a * b;
 return p;
 }
} // kraj class
```

21