7.Rad sa metodima

7.1.. Kreiranje metoda sa istim imenom

(preopterećenost metoda)

Primer:

```
class Pravougaonik {
            int  x1=0;
            int x2 = 0
            int y1 = 0;
            int y2 = 0;
      }
 Pravougaonik gradi(int x1, int y1, int x2, int y2) {
          this.x1 = x1;
          this.y1 = y1;
          this.x2 = x2;
          this.y2 = y2;
          return this;
 }
```

```
Pravougaonik gradi (Point goreLevo, Point doleDesno) {
      x1 = goreLevo.x;
      y1= goreLevo.y;
       x2 = doleDesno.x;
       y2 = doleDesno.y;
       return  this;
}
Pravougaonik gradi (Point goreLevo, int  l, int h){
          x1 = goreLevo.x;
          y1 = goreLevo.y;
          x2 = goreLevo.x + l;
          y2 = goreLevo.y + h;
          return this;
  }
///
void printPrav() {
   System.out.println("Pravougaonik :["(+x1+","+y1+
                                        ") , ("+x2+","+y2+")]");
}
```

```
public static void main (String args[]){
    Pravougaonik  p = new Pravougaonik();
    p.gradi(10,20, 30,40);
    p.printPrav();

    p.gradi(new Point(10,10), new Point(30,30));
    p.printPrav();

    p.gradi (new Point (10,10), 20, 30);
    p.printPrav();
  }
}
```

7.2. Konstruktor-metodi -Primer

```
class Knjiga {
        String  autor;
        String  naslov;
        int  brojStrana;
        Knjiga(String a, String n, int bs) {
              autor = a;
              naslov = n;
              brojStrana = bs;
        }
        void stampaPod(){
               System.out.println("Autor: "+autor);
               System.out.println("Naslov: "+naslov);
               System.out.println("Broj strana: "+brojStrana);
         }
```

```
public static void main(String args[]){
Knjiga jedina;
jedina = new Knjiga("Gospodjica","Ivo Andric",
257);
jedina.stampaPod();
    }
}
```

7.3. Preopterećenost konstruktor-metoda

Mogu biti preopterećeni kao i ostali metodi.

Ako postoji dodatni konstruktor (koji ima neke nove osobine),u njemu se mo`e pozvati ve} postoje}i sa:

```
this(arg1, arg2, …);
```

7.4. Predefinisanost metoda (overriding)

Proces definisanja metoda u potklasi koji ima isto ime kao i metod u natklasi.

Primeri:

1.4.1. Pozivanje originalnog metoda

Koristi se ključna reč ***super:***

Primer:

```
void mojMetod( int x, int y){
    //   neka naredbe
  super.mojMetod(x,y);
    //  jos naredbi
 }
```

7.5. Predefinisanje konstruktor–metoda

U principu ne može biti izvršeno jer ima isto ime kao klasa

u kojoj se nalazi. Konstruktor iz natklase može se pozvati samo preko ključne reči super.

super(arg1, arg2, …)

Primer:

```
import  java.awt.Point;
class MojaTacka extends Point {
    String ime;
    MojaTacka (int x, int y,  String ime){
        super(x,y);
        this.ime = ime;
     }
}
```

7.6. Finalizer-metod

Služi za oslobađanje memorije. Nešto suprotno konstruktor metodu. Pomaže skupljaču otpadaka.

Poziva se sa:

finalize().

Može se predefinisati u sopstvenoj klasi sa:

```
protected void finalize() {
   ……
}
```

Nije neophodno njegovo korišćenje.

7.7. Rekurzija

Java omogućava rekurzivne pozive. Realizuju se tako što metod poziva samoga sebe.

Fakt.java

7.8. Prevođenje objekta u string

Ponekad je pogodno napraviti stringovnu reprezentaciju objekta zbog jednostavnog štampanja. Za te svrhe treba predefinisati metod

```
public String toString(){ ….}
```

Knjiga2.java

7.9. Korišćenje argumenata aplikacije

Argumeti aplikacije su tipa String. Ako se žele koristiti kao druga vrsta argumenta, moraju biti prevedeni iz stringa u tu vrstu

SlucNiz.java

Kontrolna pitanja

42. Šta su konstruktor metode? Navedite primer konstruktora neke klase.
43. Šta je predefinisanost metoda?
44. Šta je finalizer metod?
45. Šta je rekurzija?