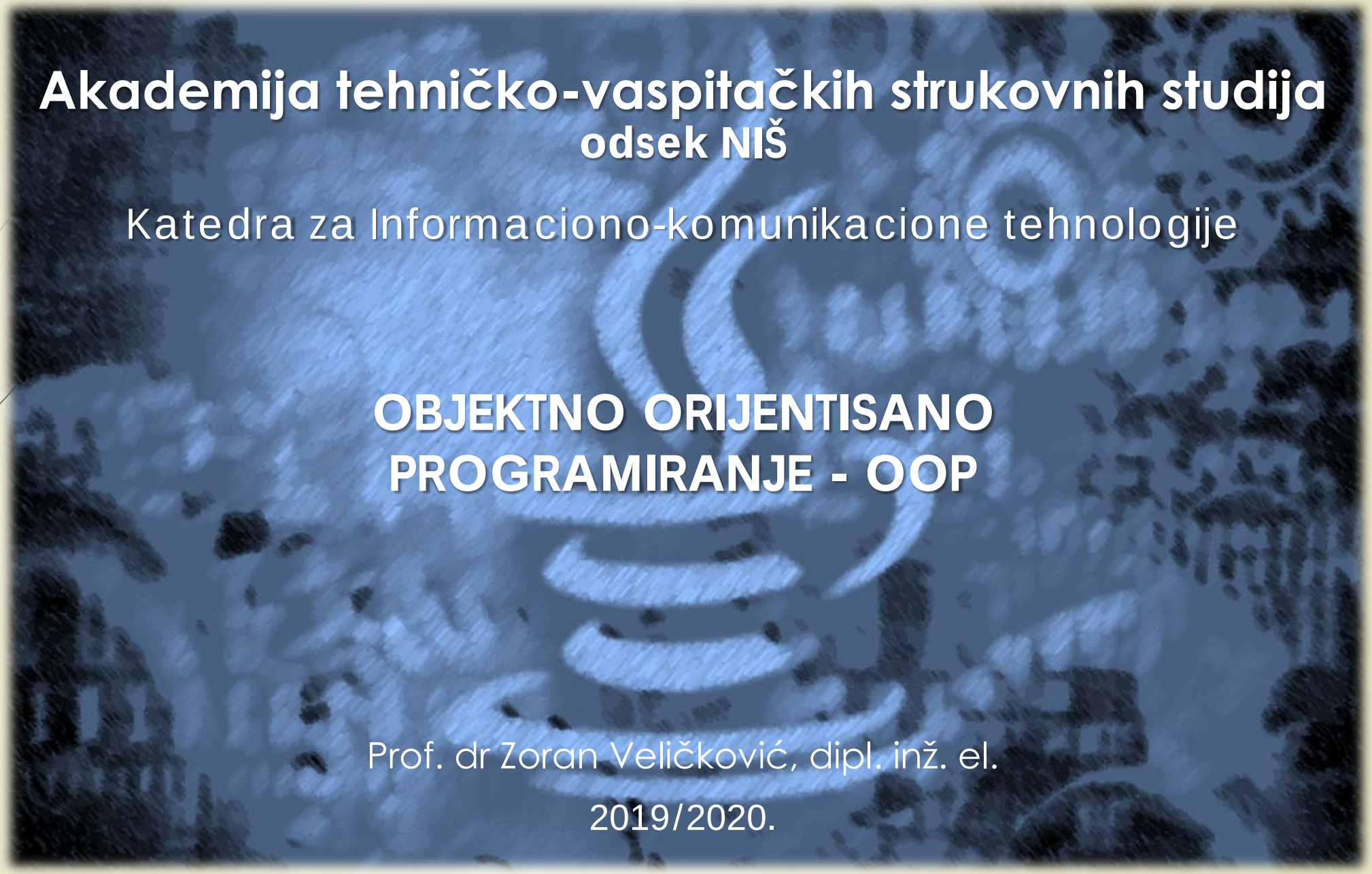


Akademija tehničko-vaspitačkih strukovnih studija
odsek NIŠ

Katedra za Informaciono-komunikacione tehnologije

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Prof. dr Zoran Veličković, dipl. inž. el.

2019/2020.

Prof. dr Zoran Veličković, dipl. inž. el.

OBJEKTNO ORIJENTISANO PROGRAMIRANJE - OOP

Osnove klasa u Javi

(6)

Sadržaj

- **UVOD U KLASE**
  - Analogije sa klasama i objektima
  - Opšta definicije klase u Javi
- **KLASE BEZ METODA**
  - Operator **new**
  - Formiranje objekta
  - Reference na objekte
- **DODAVANJE METODA KLASI**
  - Vraćanje vrednosti
  - Konstruktori klasa
  - Parametarski konstruktori
  - Preklapanje konstruktora
  - Preklapanje metoda
  - Primer klase **`Faktorijel`**

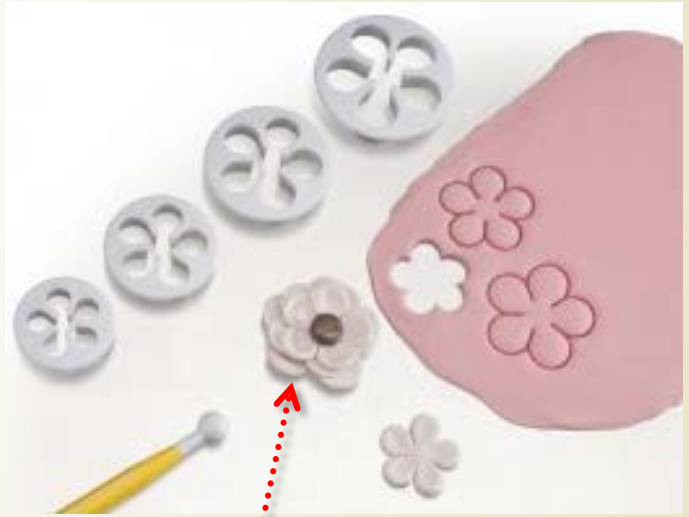
Uvod u klase

- **KLASA** (engl. *class*) predstavlja **LOGIČKU KONSTRUKCIJU** (celinu), na kojoj se zasniva programski jezik Java, kojom se definiše **FORMU** i **NAČIN PONAŠANJA** objekata.
- Već znamo iz osnova OOP-a, da se **KLASOM** definišu **NOVI TIPOVI PODATKA** koji se nazivaju **OBJEKTI** (engl. *objects*).
- Jednom istom klasom se može formirati **VIŠE OBJEKATA**.
- Dakle, klasa predstavlja "**ŠABLON**" za formiranje objekta, a objekat je samo **PRIMERAK** (često se kaže **INSTANCA**) klase.
- Kao i u .NET-u ili JavaScript-u klasa se **DEKLARIŠE** pomoću rezervisane (ključne) reči **`class`**.
- **KLASA** se sastoji od **PODATAKA** i **PROGRAMSKOG KODA** (koji se u kontekstu **OO** programiranja naziva **METODA**) koji koristi **TE PODATKE**.

Analogije sa klasama i objektima

- Klasa **MORA** da sadrži metodu pod nazivom **`main()`** samo ako u njoj **POČINJE** Java program.
- Setite se, **IZUZETAK** od ovog pravila su **APLETI** (posebna vrsta Java programa - izučavani u Internet tehnologijama) koji **NE POSEDUJU** metodu **`main()`**.

Analogija MODLICE za kolače sa **KLASOM**



Analogija utisnutog kolačića sa **OBJEKTOM**

Analogija formiranja **SLOŽENOG OBJEKTA** od nekoliko manje složenih kolačića (takođe objekata)

Opšta definicije klase u Javi

```
class ime_klase {
```

Članovi klase

```
    tip promenljiva_1;
     ...
    tip promenljiva_N;
```

1 → PODACI i PROMENLJIVE **INSTANCI**

```
    tip metoda_1 (lista parametara)
    {
        // telo metode_1
    }

    ...

    tip metoda_K (lista parametara)
    {
        // telo metode_K
    }
```

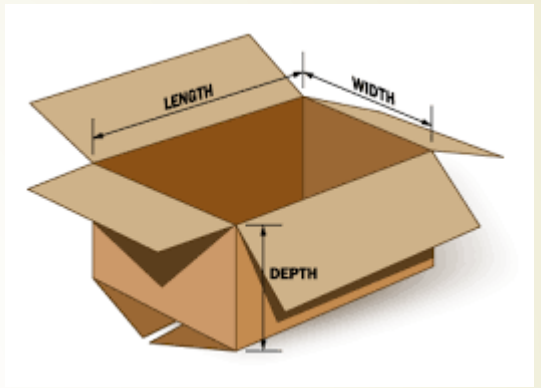
2 → PROGRAMSKI KOD **METODE**

```
}
```

KLASA

Klasa bez metoda - Kutija

```
class Kutija {
        double širina;
        double visina;
        double dubina;
       }
```



- Na ovaj način se formira **ŠABLON** za **NOVI TIP PODATAKA** (generičko – opšte ime ovog tipa je **OBJEKT**) pod nazivom **Kutija** (trenutno je bez metoda).
- Na osnovu formirane klase, **INICIJALIZACIJA OBJEKTA** se odvija u **DVA KORAKA**:
- Korak 1: **FORMIRANJE** objekta tipa **Kutija**:

```
Kutija mojaKutija = new Kutija();
```

- Korak 2: **DODELJIVANJE VREDNOSTI** promenljivama u objektu:

```
mojaKutija.širina = 100;
```

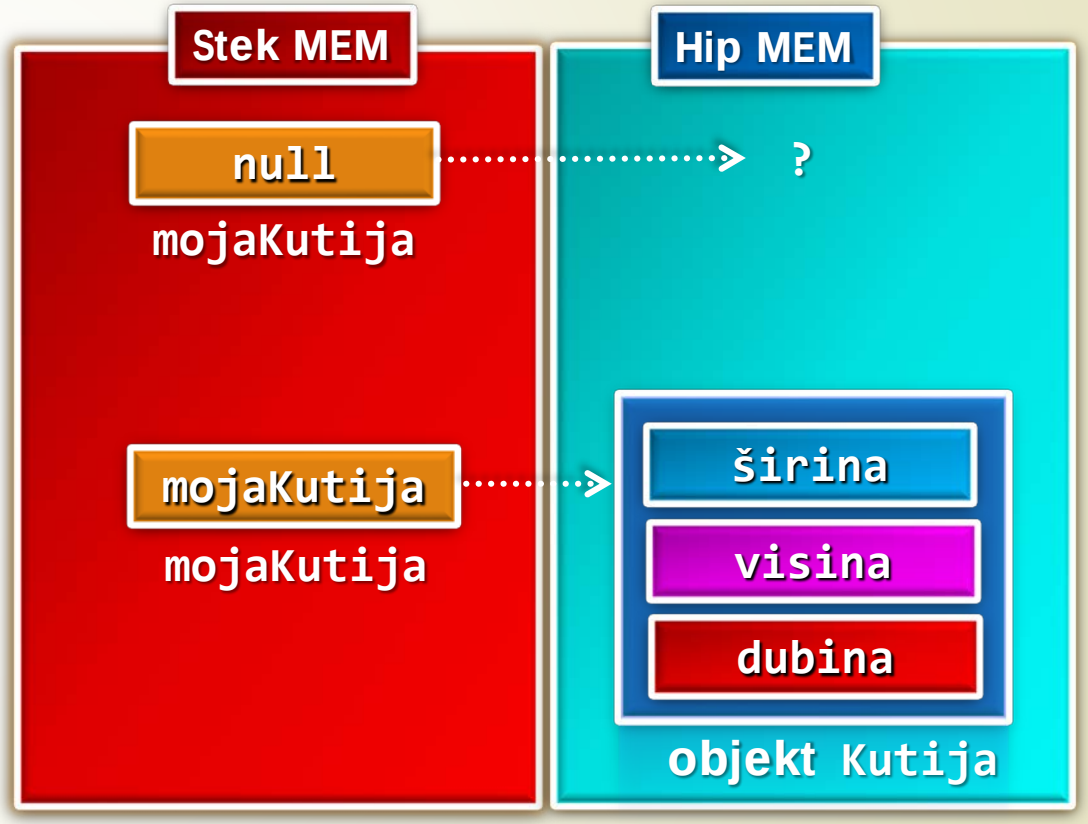
Formiranje objekta Kutija

PROGRAMSKI ISKAZ

1) `Kutija mojaKutija;`

2) `mojaKutija = new Kutija();`

EFEKAT



- Korak 1 - **FORMIRANJE** objekta tipa **Kutija**
- Korak 2 - **DODELJIVANJE** vrednosti deklarisanoj promenljivoj - inicijalizacija.

Operator new

- Operator **new** dinamički dodeljuje memoriju objektu na sledeći način:

```
Promenljiva = new imeklase();
```

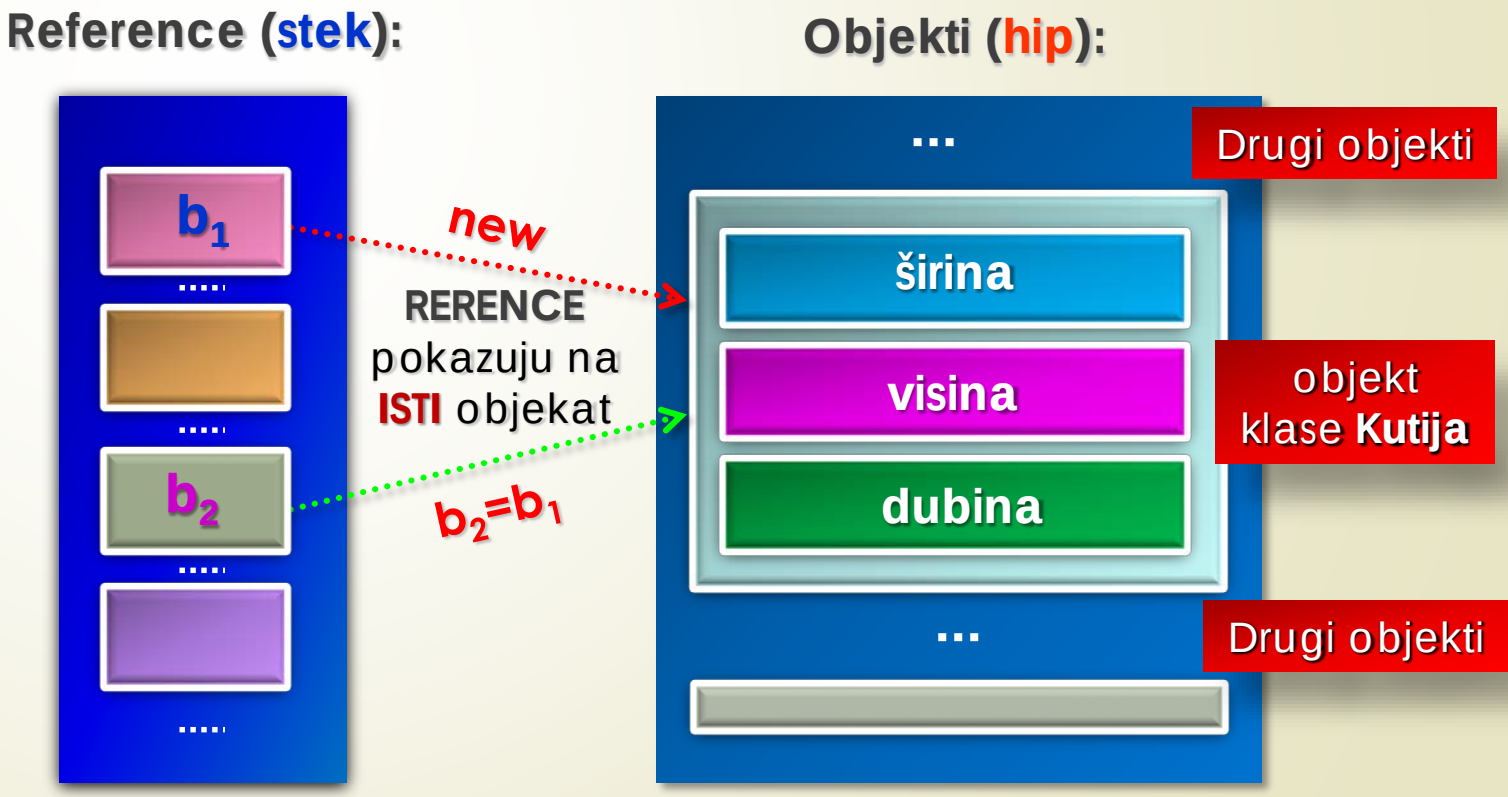
- Imenom klase iza koga sledi „**()**“ poziva se **SPECIJALNA METODA** - **KONSTRUKTOR**.
- **KONSTRUKTOR** je programski kod kojim se definiše **ŠTA** se tačno dešava kada se formira **JEDAN OD OBJEKATA KLASE**.
- Ako ne kreiramo konstruktor u izvornom kodu, Java obezbeđuje **PODRAZUMEVANI KONSTRUKTOR** (kao u prethodno prikazanom primeru).
- Već znamo, u **OO JEZICIMA** formiranim objektima može pristupiti samo preko **REFERENCI** koje su napravljene prilikom kreiranja objekata – tako je i u Javi.
- Kreiranje referenci:

```
Kutija b1 = new Kutija();
Kutija b2 = b1;
```

Reference na objekte

- Promenljivoj b_2 **NIJE DODELJENA KOPIJA** objekta b_1 već mu se dodeljuje **REFERENCA** na **ISTI OBJEKAT**!
- Ovo je suštinska razlika između **VREDNOSNIH** i **REFERENCNIH** tipova podataka.

- Promenljivoj **b_2** (referenci) **NIJE DODELJENA KOPIJA** objekta **b_1** već mu se dodeljuje **REFERENCA** na **ISTI OBJEKAT**!
- Ovo je suštinska **RAZLIKA** između **VREDNOSNIH** i **REFERENCNIH** tipova podataka.



Dodavanje metoda klasi Kutija

```
class Kutija {
    double širina;
    double visina;
    double dubina;
      // Dodavanje metode: zapremina()
    void zapremina()
    {
       System.out.print("Zapremina je: ");
       System.out.print(širina*visina*dubina);
    }
}
```

Kasa **Kutija**

Prethodna verzija klase **Kutija**

Dodato – metoda **zapremina()**

Vraćanje vrednosti - return

```
class Kutija
{
    double širina;
    double visina;
    double dubina;

    double zapremina ()
    {
        return širina*visina*dubina;
    }
}
```

Kasa **Kutija**

Prethodna verzija klase **Kutija**

Tip podatka koji se vraća pozivaocu

Dodato – metoda **zapremina()**

Rezervisana reč **return** kojom se obezbeđuje vraćanje vrednosti

Konstruktori klasa

```
class Kutija {
    double širina;
    double visina;
    double dubina;

    Kutija() {
        širina=10.0;
        visina=10.0;
        dubina=10.0;
    }

    double zapremina ()
        {
          return širina*visina*dubina;
        }
}
```

KLASA **Kutija**

KONSTRUKTOR klase **Kutija**, inicijalizuje vrednost pojedinih parametara. Ima isto ime kao i KLASA!

METODA klase Kutija **zapremina()**

Parametarski konstruktori

```
class Kutija {
    double širina;
    double visina;
    double dubina;
    Kutija(double s, double v, double d) {
        širina=s;
        visina=v;
        dubina=d;
    }
    double zapremina() {
        return širina*visina*dubina;
    }
}
```

KLASA **Kutija**

PARAMETARSKI KONSTRUKTOR **Kutija(s,v,d)**, pri inicijalizaciji definiše vrednosti parametara **s**, **v** i **d**

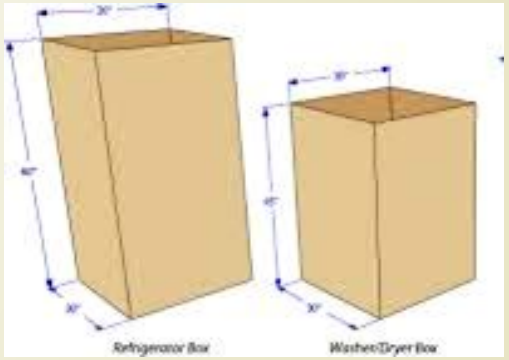
SCHILDT

Primer: parametarski konstruktori

```
class Parm_Konstr {
  public static void main (String args[]) {
  Kutija mybox1 = new Kutija (10, 20, 15);
  Kutija mybox2 = new Kutija (3, 6, 9);
  double vol;
  vol = mybox1.zapremina();
  System.out.println("Zapremina je " + vol);
  vol = mybox2.zapremina();
  System.out.println("Zapremina je " + vol);
  }
}
```

Deklarišu se DVE INSTANCE tipa **Kutija**: **mybox1** i **mybox2**. Prva ima parametre 10, 20 i 15, a druga 3, 6 i 9.

Pristup metodi **zapremina()** klase **Kutija**



Preklapanje konstruktora (1)

```
class Kutija {
  double širina; double visina; double dubina;
    Kutija(double w, double h, double d) {
       širina = w;
       visina = h;
       dubina = d;
    }
    Kutija() {
           širina = -1;
           visina = -1;                  // koristi -1 da se ukaže na
           dubina = -1;                  // neautorizovanu kutiju
    }
    Kutija(double len) {
         širina = visina = dubina = len;
    }
     double zapremina() {
         return širina * visina * dubina;
    }
}
```

I. 3-parametra

Mogu se definisati dva ili više konstruktora sa **ISTIM IMENOM (Kutija)** sve dok se njihove dekleracije **RAZLIKUJU** (sa ili bez parametra).

II. Bez parametara

III. 1-parametar

Ove metode **(Kutija)** nazivaju se **PREKLOPLJENE (overloaded)** metode.

Preklapanje konstruktora (2)

```
class PreklapanjeKonstruktora {
 public static void main (String args[]) {
   Kutija mybox1 = new Kutija(10, 20, 15);
   Kutija mybox2 = new Kutija();
   Kutija mycube = new Kutija(7);
   double vol;
 zapremima = mybox1.zapremina();
   System.out.println("Zapremina mybox1 je " + zapremima);
zapremima = mybox2.zapremina();
   System.out.println("Zapremina mybox2 je " + zapremima);
zapremima = mycube.zapremina();
   System.out.println("Zapremina mycube je " + zapremima);
 }
}
```

Tri poziva PREKLOPLJENOG konstruktora **Kutija**.

Koja će se metoda **Kutija** pozvati?

Šta je izlaz?

Preklapanje metoda

- Mogu se definisati **VIŠE METODA** sa **ISTIM IMENOM** sve dok im se **DEKLERACIJA PARAMETARA RAZLIKUJU**!
- U tom slučaju metode su **PREKLOPLJENE** (engl. *overload*).
- Na osnovu **TIPA** I/ILI **BROJA ARGUMENATA** Java određuje koja će se metoda zapravo pozvati!

Preklapanje konstruktora (3)

```
class Preklopljene_metode {
  void test() {
    System.out.println("Nema parametara");
   }
   void test(int a) {
    System.out.println("a: " + a);
   }
  void test(int a, int b) {
    System.out.println("a i b: " + a + " " + b);
   }
  double test(double a) {
    System.out.println("double a: " + a);
    return a*a;
   }
}
```

Metode sa **ISTIM IMENOM**, a različitim **POTPISOM METODE** – brojem ili tipom parametara!

Preklapanje metoda test (2)

```
class Preklapanje {
  public static void main(String args[]) {
  Preklopljene_metode obj = new Preklopljene_metode();
    double result;
    // poziv svih verzija test()
    obj.test();
    obj.test(10);
    obj.test(10, 20);
    rezultat = obj.test(123.25);
    System.out.println("Rezultat obj.test(123.25): " + rezultat);
  }
 }
```

Metode **test()** sa različitim "POTPISOM" - OVERLOAD metode.

Primer klase Faktorijel

```
class Faktorijel {
    int fakt (int n) {
        int rezultat;
        if (n == 1)
            return 1;
        rezultat = fakt (n-1) * n;
            return rezultat;
    }
}
```

Uporedite rešenje sa prethodnih predavanja.

```
class Rekurzija {
    public static void main (String args[]) {
        Faktorijel f = new Faktorijel ();
        System.out.println("Faktorijel broja 3 je " + f.fakt(3));
        System.out.println("Faktorijel broja 5 je " + f.fakt(5));
        System.out.println("Faktorijel broja 10 je " + f.fakt(10));
    }
}
```

Primer klase Faktorijel, Eclipse