# Objektna metodologija

Operator *instanceof*
Konverzija među klasama
Ispitni zadatak
Ključna reč *final*

## Operator instanceof

- Logički operator, služi za proveru da li je neki objekat instanca neke klase (podklase, interfejsa)
- Ako je **a** objekat (instanca) neke klase, a **A** određena klasa, tada logički izraz

  **a instanceof A**

  vraća vrednost **true** ako je objekat **a** instanca klase **A,** inače vraća vrednost **false.**

**Primer:**

```
class Primer1{
    public static void main(String args[ ]) {
        Primer1 s = new Primer1( );
        System.out.println(s instanceof Primer1);   // true
    }
}
```

2

## Tipovi konverzije u Javi

- Postoje 3 vrste konverzije:
  1. konverzija među tipovima podataka
  2. konverzija među klasama (objektima klasa)
  3. konverzija iz klase u tip podatka

1. konverzija među tipovima podataka
   1. Implicitna – iz užeg u širi tip, bez gubitka
   2. Eksplicitna - iz šireg u uži tip
      (ciljni tip) ime_promenljive – 'casting'

3

## Konverzija među klasama

2. konverzija među klasama (objektima klasa)

- **Moguća je samo ako su klase u relaciji nasleđivanja**
- **Konverzija je :**
  - implicitna - iz podklase u klasu
  - eksplicitna - iz klase u podklasu (konverzija se mora navesti)
- Konverzija se realizuje uz pomoć casting operator-a

  ( *imeCiljneKlase* ) imeKlase

## Konverzija među klasama

**Primer:**

```
class Student extends Osoba {
 ..............
s1: Student;
o1: Osoba;
 ..............
s1 = (Student) o1;

/* ova konverzija dodaje objektu o1 dodatna svojstva i
funkcije iz podklase Student  */
```

## Primer - Konverzija među klasama

**Primer:**

```
class Osoba {
  ......
void predstavljanje( ) {
      ......     }
}
class Student extends Osoba {
  .........
 void promeniSmer(String noviSmer) {
      ......     }
}
```

## Primer - Konverzija među klasama

```
class Zaposleni extends Osoba {
  ......
void promeniKabinet(int noviKab){
..... }
class PrimerKonverzije {
      Osoba o;   Student s;    Zaposleni z;
      .........
// može ovako kad je iz podklase u klasu
      o = s; o = z;
// ali mora eksplicitno kad je iz klase u podklasu
      s = (Student) o;      z = (Zaposleni) o;
```

## Primer - Konverzija među klasama

```
// može ovako, kad je metod nadklase
      o.predstavljanje( );
      s.predstavljanje( );
      z.predstavljanje( );
// ali može samo
      s.promeniSmer("turizam");
      z.promeniKab(noviKab);
// ne može
      o.promeniSmer ("turizam");
// čak i kad je
      o instanceof Student   - true
```

## Konverzija iz klase u tip podatka

3.1. konverzija iz klase (objekta) u tip podatka

```
int bod = Integer.parseInt(ulaz.nextLine());
```

parseInt(String s) je metod 'omotačke' klase Integer

3.2. konverzija iz tipa podatka u objekat klase

```
double d = 874.45 / 22.2;
String s = String.valueOf(d);
System.out.println(s) ;
                        // "39.38963963963964"
```

valueOf(num d) je metod klase String

9

## Primer – zadatak sa kolokvijuma

Napisati JAVA aplikaciju za potrebe jedne knjižare.

- Vode se sledeci podaci o knjigama:
  *naziv, glavni autor, izdavač, osnovna cena, broj primeraka.*
- Na cenu svih knjiga odobrava se promenljivi popust (rabat) – unos sa tastature.
- Ako je knjiga stručna, prati se još i:
  *tematska oblast* i *dodatni procenat popusta.*
- U aplikaciji treba omogućiti sledeće funkcionalnosti:
  - unos podataka o knjigama,
  - listanje kompletnog fonda knjiga - stanja na lageru knjižare,
  - listanje svih knjiga iz oblasti RAČUNARSTVO, čiji je broj primeraka 1,
  - listanje svih stručnih knjiga sa dodatnim popustom preko 10%

(2. kolokvijum)

10

## Primer

```
import java.util.*;
class Knjiga {
  String naziv;
  String autor;
  int cena;
  int brp;
  Knjiga(String naziv, String autor, int cena, int brp) {
     this.naziv=naziv;
     this.autor=autor;
     this.cena=cena;
     this.brp=brp;   }
  void prikaz( ) {
     System.out.println("\nNaziv knjige: " + naziv);
     System.out.println("Autor knjige: " + autor);
     System.out.println("Cena knjige: " + cena);
     System.out.println("Broj primeraka knjige: " + brp);   }
```

11

## Primer

```
float prodCena(int r) {
   return (cena – cena * (float) r/100);
                        // cena knjige umanjena za % popusta
  }
} // end Knjiga
class StrucK extends Knjiga {
  String oblast;
  int popust;
  StrucK(String naziv,String autor,int cena,int brp,String oblast,int
    popust) {
   super(naziv, autor, cena, brp);
   this.oblast = oblast;
   this.popust = popust;
  }
```

12

## Primer

```
void prikaz( ) {
   super.prikaz( );
   System.out.println("Oblast knjige: " + oblast);
   System.out.println("Dodatni popust: " + popust);
 }
 void rac1( ) {
  if ((oblast.equalsIgnoreCase("RACUNARSTVO")) && (brp==1))
     prikaz( );
 }
 void popust10( ) {
  if (popust > 10)
     prikaz( );
  }
```

13

## Primer

```
float prodCena(int rabat) {
  return
   (super.prodCena(rabat) – super.prodCena(rabat) *
    popust/100);
         // cena knjige umanjena za rabat i % dodatnog popusta
         // može i na 2. način
  // int pomCena = super.prodCena(vr);
  // pomCena -= pomCena * popust/100;
  // return pomCena;
 }
} // end StrucK
```

14

## Primer

```
class Knjizara {
  public static void main (String [ ] args) {
    Scanner ulaz = new Scanner (System.in);
    System.out.print("Unesite vrednost rabata u %: ");
    int r = ulaz.nextInt( );
    System.out.print ("\nUnesite broj razlicitih knjiga u knjizari: ");
    int n = ulaz.nextInt( );
    Knjiga [ ] knjige = new Knjiga[n];
    for (int i=0; i<n; i++) {
       System.out.print("\n\nUnesite naziv knjige: ");
       String naz = ulaz.nextLine( );
       System.out.print("\nUnesite autora knjige: ");
       String aut = ulaz.nextLine( );
       System.out.print("\nUnesite cenu knjige: ");
       int cena = ulaz.nextInt( );
```

15

## Primer

```
    System.out.print("\nUnesite broj primeraka knjige: ");
    int bp = ulaz.nextInt( );
    System.out.print("\nDa li je knjiga strucna? (DA/NE)");
    String odg = ulaz.nextLine();
    if (odg.equalsIgnoreCase("DA")) {
      System.out.print("\nUnesite oblast knjige: ");
      String obl = ulaz.nextLine();
      System.out.print("\nUnesite dodatni popust: ");
      int pop = ulaz.nextInt( );
      knjige[i] = new StrucK(naz, aut, cena, bp, obl, pop);
      }
    else
      knjige[i] = new Knjiga(naz, aut, cena, bp);
   } // for
```

16

## Primer

```
System.out.println("\n\nSPISAK SVIH KNJIGA U KNJIZARI");
for (int i=0; i<n; i++)
     knjige[i].prikaz( );
System.out.println
 ("\n\nKNJIGE IZ OBLASTI RACUNARSTVO SA BR. PRIMERAKA 1");
for (int i=0; i<n; i++)
    if (knjige[i] instanceof  StrucK)
       (StrucK) knjige[i].rac1();              // konverzija u podklasu
System.out.println
       ("\n\n STRUČNE KNJIGE SA POPUSTOM VECIM OD 10%");
for (int i=0; i<n; i++)
    if (knjige[i] instanceof StrucK)
       (StrucK) knjige[i].popust10( );       // konverzija u podklasu
 } // main
}   // class
```



## Ključna reč final

- Koristi se za **definisanje konstantnih vrednosti**
  - promenljivih, metoda ili klasa
- Programske konstante
  - deklarišu se sa final,
  - najčešće su static,
  - proizvoljnog su tipa

**Primer:**

```
final static int maxSlogova = 1000;
final static String poruka = "greska!";
```

- Pri dodeli vrednosti konstante obavezna je saglasnost tipova
  (kao i uvek kod naredbe dodele)

## Ključna reč final

- Final metodi
  - takvi metodi ne mogu biti redefinisani (preklopljeni) u podklasama njegove klase
  - koriste se za optimizaciju izvršnog koda

**Primer:**

```
final int vratiMaxBrzinu( ) {
      return maxBrzina; }
```

- Final klase
  - ne mogu da se nasledjuju, što znači da je zabranjena promena ili dopuna njihovih svojstava i ponašanja (atributa i metoda) u podklasama

## Ključna reč final

- **Primer:**

```
public final class Odeljenje {……..}
```

- Klase String, Math, InetAddress iz Javine biblioteke klasa
  - java.lang.String,
  - java.lang.Math,
  - java.net.InetAddress

  su definisane kao final klase

- U final klasama i sve metode moraju biti final
  - to ne moramo eksplicitno naglašavati

## Ključna reč `final`

- Ključna reč final može se koristiti:
  - u zaglavlju deklaracije klase,
  - u zaglavlju deklaracije metoda i
  - pri deklaraciji promenjivih
- U zavisnosti gde se koristi ima različitu ulogu, ali uvek je to zabrana neke promene:
  - final u deklaraciji promenjivih označava da se one neće moći menjati (tj. deklariše konstantu)
  - final se koristi kod metoda ako je dozvoljeno nasleđivanje neke klase, ali želimo da sprečimo redefinisanje nekog njenog metoda u podklasama
  - ako hoćemo da zabranimo nasleđivanje neke klase, tada je definišemo sa final

21

## Primer – final metod

```
Class Autoput{

  private int brojTraka = 3;

  final void postaviBrojTraka(int noviBrojTraka){
      brojTraka = noviBrojTraka;
  }

  final int citajBrojTraka(){
      return brojTraka;
  }
// ovi metodi se ne mogu preklopiti – redefinisati
  u podklasi zbog modifikatora final
}
```

22