Objektna metodologija

Operator *instanceof*
Konverzija među klasama
Primer - ispitni zadatak
Ključna reč *final*
Obrada izuzetaka

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

Konverzija među klasama

- **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
```

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)

8

Primer – zadatak sa kolokvijuma

```
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);  }
```

9

Primer – zadatak sa kolokvijuma

```
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;
  }
```

10

Primer – zadatak sa kolokvijuma

```
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( );
   }
```

11

Primer – zadatak sa kolokvijuma

```
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
```

12

Primer – zadatak sa kolokvijuma

```
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( );
```

13

Primer – zadatak sa kolokvijuma

```
    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
```

14

Primer – zadatak sa kolokvijuma

```
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
```

15

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 nasleđuju, 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)
  - ako hoćemo da zabranimo nasleđivanje neke klase, tada je definišemo sa final
  - final se koristi kod metoda ako je dozvoljeno nasleđivanje neke klase, ali želimo da sprečimo redefinisanje nekog njenog metoda u podklasama

19

Ključna reč `final`

- Modifikator final se može koristiti:
  - u zaglavlju deklaracije klase
    - ako hoćemo da zabranimo njeno nasleđivanje
  - u zaglavlju deklaracije metoda
    - da sprečimo njegovo redefinisanje u podklasama
  - pri deklaraciji promenjive
    - za definisanje konstante
- U zavisnosti gde se koristi final ima različitu ulogu

20

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
}
```

21

Obrada izuzetaka

- Java jezik ima osobinu automatskog upravljanja izuzecima - greškama koje mogu nastati u fazi izvođenja programa
- Izuzeci u Java kodu su objekti klase **Exception** i njenih podklasa koje ih bliže određuju i klasifikuju, kao što su:
  - IllegalArgumentException,
  - ArrayIndexOutOfBoundsException,
  - ArithmeticException, ...
- Greške koje se ne mogu prepoznati prilikom prevođenja, a dešavaju se u fazi izvođenja programa obrađuju se u Java kodu `try-catch` i `throw` naredbama

22

Obrada izuzetaka

- Izuzeci u Javi - neželjene programske greške u fazi izvođenja programa, mogu biti npr:
  - deljenje nulom,
  - pristup nepostojećem elementu niza, (indeks niza izvan definisanih granica)
  - nemoguća eksplicitna konverzija,
  - prekid mrežne komunikacije,
  - ...

23

`try-catch` naredba

Sintaksa:

```
try {
    // deo koda u kome se može desiti izuzetak
} catch(NazivKlaseIzuzetka1 NazivPromenjive1) {
     // obrada prvog izuzetka

} catch(NazivKlaseIzuzetka2 NazivPromenjive2) {
     // obrada drugog izuzetka

} ....
```

24

Primer `try-catch` naredbe

```
int i = 30;
try {

   int niz[] = new int[10];
   niz[i] = 7;
}
catch ( ArrayIndexOutOfBoundsException e ) {

   System.out.println("Pogrešan index niza!");
   e.printStackTrace( );
}
```

25

Primer `try-catch` naredbe

```
import java.util.Scanner;
public class Deljenje {
   public static void main(String[] args) {
      int brojilac, imenilac, kolicnik;
      Scanner ulaz = new Scanner(System.in);
      System.out.println("Unesite brojilac: ");
      brojilac = ulaz.nextInt();
      System.out.println("Unesite imenilac: ");
      imenilac = ulaz.nextInt();
      try {
         kolicnik = brojilac/imenilac;
         System.out.print("kolicnik je: " + kolicnik);
      } catch (ArithmeticException e) {
         System.out.println("Deljenje sa nulom!!");
         e.printStackTrace()
      }

   }
}
```

26

Objektna metodologija – osnovni pojmovi sažetak

- Definicija klase sadrži:
  - Promenljive (atributi, obeležja)
  - Metode (funkcije)
- Promenljive instance i promenljive klase
- Metodi instance i metodi klase
- Promenljive klase - deklaracija

  static tipPromenljive imePromenljive;
- Metodi klase - deklaracija

  static .... imeMetoda( );

Kreiranje objekta - sažetak

- Objekat (instanca klase)
- Kreiranje objekata
  - Operator *new*
  - Metod konstruktor

  imeObj = new Konstruktor( );
- Metod konstruktor ima isto ime kao klasa
- Svaka klasa ima default konstruktor

  (prazan konstruktor, bez argumenata)

  imeKlase() { }

Referenciranje na promenljive i metode - sažetak

- Referenciranje na promenljivu instance
  imeObjekta.imePromenljiveInstance
- Referenciranje na metod instance
  imeObjekta.imeMetodaInstance()
- Referenciranje na promenljivu klase
  ImeKlase.imePromenljiveKlase ili samo
  imePromenljiveKlase – iz iste klase
- Referenciranje na metod klase
  ImeKlase.imeMetodaKlase() ili samo
  imeMetodaKlase() – iz iste klase