Objektna metodologija

Definicija klase – primeri
Pozivi statičkih metoda
Specifikatori pristupa
Nizovi objekata
Metodi preopterećenja (*overload* metodi)

Primer - definicija klase Bicikl

Definisati klasu Bicikl opisanu sa:

- tri celobrojne promenljive (godište, brzina, stepen prenosa),
- metodom za inicijalizaciju vrednosti godište konkretnog bicikla
- metodom za promenu stepena prenosa
- metodom za povećanje brzine za zadatu vrednost
- metodom za smanjenje brzine za zadatu vrednost
- metodom za prikaz svih promenljivih konkretnog bicikla

2

Primer - definicija klase Bicikl

```
// ideja OOM je da se vrednosti promenljivih mogu menjati samo
   pomoću metoda

class Bicikl {
    int godiste = 0;                    // definicija promenljivih klase
    int brzina = 0;
    int stepenPrenosa = 1;
                                        // definicija metoda klase
    void postaviGodiste(int novaVrednost) {
          godiste = novaVrednost;
    }

    void promeniStepenPrenosa(int novaVrednost) {
          stepenPrenosa = novaVrednost;
    }

    void povecajBrzinu(int inkrement) {
          brzina = brzina + inkrement;
    }
```

3

Primer - definicija klase Bicikl

```
    void smanjiBrzinu(int dekrement) {
          brzina = brzina - dekrement;
    }

    void stampajStanja() {
          System.out.println("godiste: " + godiste");
          System.out.println("brzina je: " + brzina
          System.out.println("stepen prenosa je: " + StepenPrenosa);
    }
}

class BiciklDemo {          /          / klasa koja koristi klasu Bicikl
    public static void main(String[] args) {

          // kreiramo dva nova objekta tipa Bicikl
          Bicikl bike1 = new Bicikl();
          Bicikl bike2 = new Bicikl();
```

4

Primer – klasa koja koristi klasu Bicikl

```
        // poziv metoda za te objekte

        bike1.postaviGodiste(2004);
        bike1.povecajBrzinu(10);
        bike1.promeniStepenPrenosa(2);
        bike1.stampajStanja();
        bike2.postaviGodiste(2006);
        bike2.povecajBrzinu(20);
        bike2.promeniStepenPrenosa(3);
        bike2.smanjiBrzinu(10);
        bike2.promeniStepenPrenosa(2);
        bike2.stampajStanja();
    }
}
```

5

Statičke promenljive i metodi

- Deklarišu se pomoću ključne reči static
  - static imePromenljive
  - static imeMetoda(lista argumenata)
- Koriste se bez prethodnog kreiranja instanci klase
- Pozivaju se na izvršenje:
  - izvan klase - preko imena klase ImeKlase.imeMetoda( ) ili
  - u okviru iste klase samo imenom metoda – imeMetoda( )
  - Statički metodi ne smeju pristupati promenljivama koje nisu statičke, niti smeju pozivati metode koji nisu statički!

6

Primer – definisanje statičkog metoda (metoda klase)

```
// Definisati statički metod ocenaFilma (int s, int g, int r) koji računa
   ocenu filma kao zbir ocena za scenario, glumu i režiju

public static int ocenaFilma(int s, int g, int r) {
       return s + g + r;
 }

// Definisati  statički metod prikaziOcenu (int s, int g, int r) koji prikazuje
   ocenu filma pozivom metoda ocenaFilma (int s,int g,int r)

 public static void prikaziOcenu(int s, int g, int r){
    System.out.print("Ocena filma je: ");
    System.out.println(Film.ocenaFilma(s, g, r));
 }
       // poziv metoda klase (statičkog metoda) iz drugog metoda
```

7

Primer – dva tipa poziva metoda klase

```
// prva klasa Film
public class Film {
    public static int ocenaFilma(int s, int g, int r) {
        return s+g+r;
    }
}
// druga klasa Demo, koja je aplikacija

public class Demo {
    public static void main (String args[]) {
        int scenario = 6, gluma = 9, rezija = 8;
        // 1.poziv statičkog metoda - iz iste klase
        prikaziOcenu(scenario, gluma, rezija);
    }
    public static void prikaziOcenu(int s, int g, int r){
        System.out.print("Ocena filma je: ");
        // 2.poziv statičkog metoda - iz druge klase
        System.out.println(Film.ocenaFilma(s, g, r));
    }
}
```

8

Specifikatori pristupa

- *public*
  - pristup dozvoljen svim klasama
- *friendly* (izostavljen)
  - pristup dozvoljen svim klasama iz tekućeg paketa
- *protected*
  - pristup dozvoljen samo klasi i njenim podklasama
- *private*
  - pristup dozvoljen samo iz date klase

Zadatak

Definisati Java klasu za prikaz pravougaonika, ako su date realne vrednosti x1,x2,y1,y2 koje određuju koordinate temena

A(x1,y1), B(x2,y1), C(x2,y2), D(x2,y1)

(slika 1.)

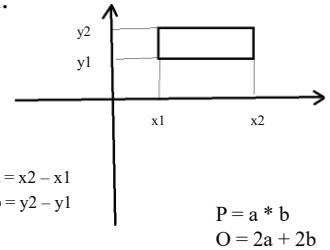
(stranice su mu paralelne sa x osom).

Definisati metod konstruktor, metode za racunanje obima i povrsine pravougaonika, kao i za predstavljanje pravougaonika preko njegovih stranica, obima i povrsine.

10

Zadatak

Slika 1.



a = x2 – x1
b = y2 – y1

P = a * b
O = 2a + 2b

* Ako je pravougaonik u ostalim kvadrantima, stranice a i b treba računati kao apsolutne vrednosti datih razlika

11

Primer - klasa za prikaz pravougaonika

```
/* Definisati Java klasu za prikaz pravougaonika, ako su
   date realne vrednosti x1,x2,y1,y2 koje određuju
   koordinate temena  (slika 1.)
            A(x1,y1), B(x2,y1), C(x2,y2), D(x2,y1)
   (stranice su mu paralelne sa x osom)
   Definisati metod konstruktor, metode za racunanje obima i
   povrsine pravougaonika, kao i za predstavljanje
   pravougaonika preko njegovih stranica, obima i povrsine.
*/


class Pravougaonik {

private float gore, dole, levo, desno;

        // metod konstruktor
```

12

Primer - klasa za prikaz pravougaonika

```
Pravougaonik(float gore, float dole, float levo, float
   desno){
          this.gore = gore;
          this.dole = dole;
          this.levo = levo;
          this.desno = desno;
    }

float povrsina(){
        return (gore - dole)*(desno - levo);
   }


float obim(){
        return 2*(gore - dole)+ 2*(desno - levo);
   }
        ...
 }   // end class
```

13

Zadatak – Student1

Definisati klasu Student1 sa metodama:

- za promenu smera,
- prelazak na sledeću godinu studija i
- predstavljanje vrednosti svih obeležja.

U aplikaciji treba da se:

- kreiraju dva objekta klase Student1,
- izlista početni spisak studenata,
- prvom studentu promeni smer,
- drugi upiše u narednu godinu studija i
- na kraju izlista promenjeni spisak studenata.

Zadatak – Student1

Jednostavni primer definicije klase Student sa promenljivima: broj indeksa, ime, smer, godina studija i metodama za:

promenu smera,
prelazak na sledecu godinu studija,
metodom za upis studenta (konstruktorom) i
metodama za predstavljanje objekata klase Student.

U aplikaciji treba da se:

kreiraju dva objekta klase Student,
prvom studentu se menja smer,
drugi student se upisuje u narednu godinu studija,
na kraju se lista promenjeni spisak studenata.

15

Zadatak – Student1

```
import java.util.*;
public class Student1 {
      private String brInd;
      private String ime;
      private String smer;
      private int godStudija;     // 1, 2, 3
 Student1(String brInd,String ime, String smer){
      this.brInd=brInd;
      this.ime=ime;
      this.smer=smer;
      this.godStudija=1;
  }
  public void promeniSmer(String noviSmer) {
     smer=noviSmer;
  }
  public void povecajGodStudija() {
     godStudija++;
  }
```

16

Zadatak – Student1

```
String dajIme() {
    return ime;
}
String dajBrInd() {
    return brInd;
}
String dajSmer() {
    return smer;
}
int dajGodSt() {
    return godStudija;
}
void prikaziSt() {
    System.out.println("Broj indeksa: " + brInd);
    System.out.println("Ime i prezime " + ime);
    System.out.println("Smer: " + smer);
    System.out.println("Godina studija: " + godStudija);
}
```

17

Zadatak – Student1

```
public static void main(String [] args) {
    Student1 a;
     a=new Student1("123/06","Pera Petrovic","informatika");
    Student1 b=new Student1("124/06","Ana Stanic","turizam");     // 2. nacin
    System.out.println("Spisak studenata");
    a.prikaziSt();
    b.prikaziSt();
    System.out.println("Unesi ime novog smera za prvog studenta: ");
     Scanner ulaz=new Scanner(System.in);
      String ns=ulaz.nextLine();
    a.promeniSmer(ns);
    System.out.println("Upisi drugog studenta u sledecu godinu");
    b.povecajGodStudija();
     System.out.println("Azurirani spisak studenata");
    a.prikaziSt();
    b.prikaziSt();
    }   // main
}   // class
```

18

Nizovi objekata

- Elementi nizova ne moraju biti samo prosti tipovi (*integer, double, float, char*,...)
- Moguće je definisati i nizove objekata: Stringova, Studenata, Knjiga i sl.
- Da bi se koristio niz objekata potrebno je:
  - kreirati niz čiji će elementi biti objekti date klase
  - u okviru ovog niza kreirati svaki pojedinačni objekat
  - to znači da će se **dva puta koristiti operator new**

19

Nizovi objekata

1. Deklarisanje niza objekata `nizStudenata` tipa `Student`

```
Student nizStudenata[];
```

2. Kreiranje niza od 10 objekata

```
nizStudenata = new Student[10];
    // obezbeđuje 10 mesta u memoriji koji će čuvati reference na
    objekte klase Student
```

ili 1. i 2. korak zajedno

```
Student nizStudenata[] = new Student[10];
```

3. Kreiranje objekata klase Student i dodela el. niza:

```
nizStudenata[0] = new Student( );
nizStudenata[1] = new Student( );
.........
```

20

Zadatak – klasa *Student2*

Definicija klase Student2 (brInd, prezIme, godR, smer) sadrži metode:

- za prikaz obeležja klase,
- promenu nekih obeležja klase i
- metod za proveru da li je određeni student sa *smera Informatika*.

U aplikaciji treba:

- kreirati niz objekata klase Student2 i
- dati izveštaj o ukupnom broju studenata na smeru *Informatika*.

Zadatak – klasa *Student2*

```
/* Primer definicije klase Student2 sa metodama za prikaz i promenu
   nekih obeležja klase, kao i metodom za proveru da li je obeležje
   smer jednako Informatika.
   U aplikaciji se kreira niz objekata klase Student2 i daje izveštaj o
   ukupnom broju studenata na smeru Informatika. **/
import java.util.*;
class Student2 {
private String brInd;
private String prezIme;
private int godR;
private String smer;
Student2 (String brInd, String prezIme, int godR, String smer) {
   this.brInd = brInd;
   this.prezIme = prezIme;
   this.godR = godR;
   this.smer = smer;
}
```

22

Zadatak – klasa *Student2*

```
String prikaziInd( ) {
   return brInd; }
String prikaziIme( ) {
   return prezIme; }
int prikaziGodR( ) {
   return godR; }
String prikaziSmer( ) {
   return smer; }
void promeniSmer(String noviSmer) {
   smer=noviSmer; }
void promeniPrezIme(String novoPrezIme) {
   prezIme=novoPrezIme; }
void prikaz( ) {
   System.out.print("\nBroj indeksa: " + brInd +" Ime i prezime: " +
   prezIme + " Godina rodjenja: " + godR + " Smer: " + smer);
}
```

23

Zadatak – klasa *Student2*

```
boolean smerInf( ) {
   if (smer.equalsIgnoreCase("informatika"))
      return true;
   else
      return false; }
public static void main(String [ ] args) {
 Scanner ulaz = new Scanner(System.in);
 System.out.print("Unesi broj studenata: ");
 int broj = ulaz.nextInt( );
 Student nizStud [ ] = new Student[broj];
 System.out.println("Unesi podatke o studentima:");
 for (int i = 0; i < broj; i++) {
    System.out.print("\nBroj indeksa: ");
    String brI = ulaz.nextLine( );
    System.out.print("\nPrezime i ime studenta: ");
    String prIm = ulaz.nextLine( );
```

24

Zadatak – klasa *Student2*

```
    System.out.print("\nGodina rodjenja: ");
    int gr = ulaz.nextInt( );
    System.out.print("\nSmer: ");
    String sm = ulaz.nextLine( );
    nizStud[i] = new Student (brI, prIm, gr, sm);
    nizStud[i].prikaz( ); }
  int brojac = 0;              // brojac studenata smera Informatika
  System.out.println("\nSpisak studenata smera Informatika:");
  for (int i = 0; i < broj; i++) {
      if (nizStud[i].smerInf( )) {
          brojac++;
          nizStud[i].prikaz( );  }
  }
  System.out.println("\nNa smeru Informatika ima " + brojac +
                     " studenata " );
  } }
```

25

Metodi preopterećenja (overload methods)

- U Java klasi moguće je da postoji više verzija istog metoda (tj. više metoda sa istim imenom), ali sa različitom listom argumenata (različitim brojem i/ili tipovima argumenata)
- To važi i za metode konstruktore
- Java bira koji će od takvih metoda pokrenuti na osnovu liste argumenata

Primer overload konstruktora:

```
class Voce {
      int grami, kalorijePoGramu;
  Voce( ) {
      grami = 50;
   }
   Voce(int a, int b) {
       grami = a;
       kalorijePoGramu = b;
   }
}
```

26

Primer overload metoda

```
public class PrimerOverload{

    int saberi ( int br1, int br2 ){
       return br1 + br2;
    }

    double saberi( double num1, double num2 ){
        return num1 + num2;
    }

    public static void main ( String args[] ){
         int x = saberi(2, 3);
         System.out.println("2 + 3 = " + x);

         double y = saberi(2.5, 3.1);
         System.out.println("2.5 + 3.1 = " + y);
   }

} // primer polimorfizma – važnog svojstva OO jezika
```

27