Objektna metodologija

Principi OOM
Klase i objekti
Metod konstruktor
Ključna reč this
Definicija klase – primeri

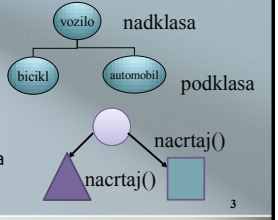
Objektna metodologija

- Svaki informacioni sistem je slika nekog realnog sistema
  - programi su modeli za rešavanje konkretnih problema iz realnog sveta
- Realni svet se sastoji od objekata koji međusobno komuniciraju (razmenjuju informacije)
- Objektno orijentisana metodologija (OOM) zasniva se na međusobnoj komunikaciji objekata koji su definisani po šablonima koje nazivamo KLASE
- Objektno orijentisano programiranje (OOP) podrazumeva upotrebu objektno orijentisane metodologije

2

Principi OOP-a

- **Enkapsulacija**
  - U klasi su 'učaurene' njene promenljive (atributi) zajedno sa metodama (funkcijama).
- **Skrivanje informacija**
  - Objekti klase **skrivaju** svoje **promenjive** i **metode.**
- **Nasleđivanje**
  - Podklasa nasleđuje sve promenjive i metode svoje nadklase.
- **Polimorfizam**
  - Primena istih funkcija na različitim tipovima podataka



3

Nasleđivanje (inheritance)

- Predstavlja odnos dve klase u kome:
  - podklasa nasleđuje sve atribute i funkcije osnovne klase i
  - podklasa definiše još i svoje specifične atribute i funkcije

Primer:
Neka postoje klase Osoba i Student
Klasa Osoba, sa atributima:
matični broj, pol, datum rođenja, adresa
Klasa Student, sa atributima:
broj indeksa, semestar, prosečna ocena

4

Nasleđivanje (inheritance)

Ako klasa Student nasleđuje klasu Osoba, tada:

1. klasa Student nasleđuje atribute klase Osoba: *matični broj, pol, datum rođenja, adresa*
2. klasa Student definiše i svoje specifične atribute: *broj indeksa*, *semestar*, *prosečna ocena*
3. Podklasa Student, pored funkcija (metoda) definisanih u klasi Osoba, ima definisane i dodatne funkcije (metode) za pristup i promenu atributa podklase Student

5

Nasleđivanje (inheritance)

- Neophodni minimum potreban da se jedan programski jezik svrsta u objektno orijentisane programske jezike jeste da podržava:
  - koncept klase
  - koncept objekta i
  - koncept nasleđivanja
- U teoriji postoji:
  - jednostruko nasleđivanje (moguć samo jedan roditelj za jednu klasu) i
  - višestruko nasleđivanje (moguće više roditelja jedne klase)
- Programski jezik Java podržava isključivo jednostruko nasleđivanje

6

Polimorfizam

- Karakteristika objektno-orijentisanih programskih jezika da se u okviru klasa mogu definisati različite funkcije sa istim nazivom i listom argumenata.
- Tek u vreme izvršenja programa određuje se koja će konkretno funkcija biti pozvana, u zavisnosti od tipa objekta nad kojim je pozvana.
- Polimorfizam i nasleđivanje omogućavaju da se razvije hijerarhija klasa (biblioteka) koje se mogu upotrebiti u razvoju različitih aplikacija.
  - Ovo su ključne osobine OOM i OO programskih jezika koje omogućavaju višestruko korišćenje već napisanog koda ('reusability')

7

Definicija klase

- Definicija klase sadrži:
  - deklaraciju promenljivih (atributa, obeležja)
  - definiciju metoda (funkcija)
  - main metod (samo opciono, ako je klasa aplikacija tj. ako je klasa izvršna klasa)
- Klase koje nisu aplikacije nemaju main metod.

8

Java aplikacija – osnovna šema

```
class  PrimerApl {
        // deklaracija promenljivih
        // definicije metoda
   zaglavlje metoda1 {
        // telo metoda1
   }
   ....
   public static void main (String [ ] args) {    // izvršni deo

        // telo main metoda

   }    // end main
}       // end class
```

Zaglavlje metoda

- Ima sledeću formu

[static] [specPristupa] povratniTip imeMetoda (lista argumenata)

static – samo za metode klase

specPristupa – *public, protected, private*

povratniTip – *void* ili bilo koji tip podatka

lista argumenata –

tipArg1 imeArg1, tipArg2 imeArg2, …

Primer – zaglavlje metoda

- public void listanje ( );
  - metod listanje je bez argumenata
- static int racun (int s1, int s2, int bod);
- protected void promeniSmer (String a);
- static float prosek (int [ ] niz, int n);

11

Metod konstruktor

- je posebna vrsta metoda koji služi za kreiranje objekata klase
- Poziva se uvek u kombinaciji sa operatorom new
- Metod konstruktor se lako prepoznaje u definiciji klase po tome što:
  - ima isto ime kao klasa
  - nema povratni tip kao ostali metodi

12

Metod konstruktor

Šta se dešava kada se pozove metod konstruktor?

- daje se ime objektu
- deklariše se tip objekta
  (to je klasa kojoj objekat pripada)
- rezerviše se mesto u memoriji za strukturu definisanu šablonom klase
- inicijalizuje se objekat – promenjlive objekta dobijaju početne vrednosti (default ili zadate)

13

Metod konstruktor

- Ako u definiciji klase nema konstruktora, za kreiranje objekata u svakoj klasi postoji default ili implicitni konstruktor
- Implicitni konstruktor svake klase ima:
  - praznu listu argumenata i
  - prazno telo metoda

    imeKlase( ) { }
- U klasi može biti više različitih metoda konstruktora (*overload konstruktori*), ali se onda mora razlikovati njihova lista argumenata

14

Primer - definicija klase Voce

Definisati klasu Voce koja je opisana sa:

- dva celobrojna atributa
  - grami i
  - kalorijePoGramu,
- tri metoda:
  - dva metoda konstruktora i
  - metodom ukupnoKalorija()
    koji izračunava ukupne kalorije
    kao proizvod težine (u gramima) i kalorija po gramu.

15

Primer - definicija klase Voce

- Definisati klasu Voce sa dva celobrojna atributa grami i kalorijePoGramu i metodom konstruktorom koji inicijalizuje oba atributa zadatim vrednostima, kao i metodom za računanje ukupnih kalorija.

```
class Voce {
     int grami;          // težina u gramima
     int kalorijePoGramu;
  Voce ( ) {}            // default konstruktor
  Voce (int a, int b){   // metod konstruktor
      grami = a;
      kalorijePoGramu = b;
   }
  int ukupnoKalorija() {
    return(grami* kalorijePoGramu);
   }
}
```

16

Kreiranje objekta (instance) klase

- Za kreiranje objekata (instanci) klase koriste se operator new i metod konstruktor

Primer: kreiranje 2 objekta (2 instance) klase Voce koje smo nazvali sljiva i kruska

```
Voce sljiva = new Voce();
// nakon kreiranja objekta sljiva atributi će imati
   default vrednosti u memoriji: 0 i 0.

Voce kruska = new Voce(50,10);
// objekat kruska biće inicijalizovan,
   tj. atributi će dobiti vrednosti 50 i 10.
```

17

Kreiranje objekata (instanci) klase

- Deo programskog koda klase Voce:
  Voce sljiva = new Voce();

pokreće metod konstruktor Voce().

Tom prilikom se:

- kreira objekat sa imenom sljiva
- rezerviše mesto u memoriji koje ima strukturu klase Voce

- Konstruktor sa argumentima Voce (50,10) inicijalizuje objekat kruska (postavlja vrednosti promenljivih instance kreiranog objekta na 50 i 10)

18

Višestruki konstruktori - overload konstruktori

- U Javi je moguće definisati više konstruktora sa različitim listama argumenata u jednoj klasi:

Npr.

Voce(),
Voce(int a),
Voce(int a, int b) ...

- Na taj način je omogućeno kreiranje objekata na različite načine vezano za inicijalizaciju njihovih promenljivih

- Ovakvi višestruki konstruktor metodi se zovu **overload konstruktori**

19

Pristup promenljivama instance (referenciranje, obraćanje)

- Pristup primenljivoj instance vrši se navođenjem njenog objekta
  imeObjekta.imePromenljive

- Tačka (dot operator) je upravljački simbol koji služi za pristup (obraćanje) promenljivama i metodama instance date klase

- Promenljiva instance konkretnog objekta ponaša se isto kao i svaka promenljiva u Javi, dakle, prema svom tipu može biti:
  - u sastavu odgovarajućeg izraza;
  - argument metoda za štampu i sl.

20

Pristup promenljivama instance - primer

```
// referenciranje na promenljive objekta sljiva i kruska
 ....
   int ukupno = sljiva.grami + kruska.grami;
 ....
  if (sljiva.kalorijePoGramu > kruska.kalorijePoGramu)
          System.out.print("Šljiva je kaloričnije voće")
  else {
     if (sljiva.kalorijePoGramu < kruska.kalorijePoGramu)
           System.out.print("Kruška je kaloričnije voće")
     else
      System.out.print("Kruška i šljiva su jednako kalorične");
}
```

21

Pristup metodama instance (referenciranje, obraćanje)

- Tačka (dot operator) je upravljački simbol koji služi za pristup i pokretanje metoda instance date klase
- Poziv metoda instance

  imeObjekta.imeMetoda( )
- Primer:

```
/*  pokretanje metoda ukupneKalorije() klase Voce na
    objektu sljiva  */
....
    int kalorije;
    kalorije = sljiva.ukupneKalorije();
```

22

Primer - definicija klase Tacka

Definisati klasu Tacka opisanu sa:

- dve realne promenljive (koordinate tačke u ravni),
- metodom konstruktorom koji inicijalizuje obe koordinate tačke,
- metodom za translaciju tačke u ravni za zadate vrednostii
- metodom za prikaz pozicije konkretne tačke u ravni

23

Primer - definicija klase Tacka

```
class Tacka{
  float x,y;

  Tacka(float a, float b){  // metod konstruktor
      x = a;
      y = b;
  }

  void transliraj(float promenaX, float promenaY){
      x = x + promenaX;
      y = y + promenaY;
  }

  String prikaz(){
    return
      "Tacka sa koordinatama (" + x + "," + y + ")";
  }
}
```

24

Primer – kreiranje objekta (instance) klase, poziv metoda instance, obraćanje promenljivoj instance

```
class TackaDemo{
  public static void main (String[] args){
      Tacka a = new Tacka(1,3);
      // kreiranje objekta (instance) a
      System.out.println(a.prikaz());
                                  //(1.0, 3.0)
      a.transliraj(1,2); // poziv metoda
      System.out.println(a.prikaz());
                                  //(2.0, 5.0)
      a.x = 0;     // obraćanje promenljivima
      a.y = 0;     // x i y objekta a

      System.out.println(a.prikaz());
                                  //(0.0, 0.0)
   }
}
```

25

Promenljive instance i promenljive klase

- Promenljive u Javi se dele na:
  - promenljive instance,
  - promenljive klase i
  - lokalne promenljive
- **Promenljiva instance je vezana za konkretni objekat**
  - Vrednosti promenljive instance su dostupne samo kad se zna konkretni objekat na koji se odnose
- **Promenljive klase imaju istu vrednost za sve objekte** jedne klase
- **Promenljive klase** se deklarišu pomoću rezervisane reči **static**

26

Promenljive instance i promenljive klase

Primer:

```
class Student {
  static  int  ukupanBrojStudenata;     //  promenljiva klase
  String brojIndeksa;                   //  promenljiva instance
  String imePrezime;                    //  promenljiva instance
  ..........
  static int brojanje( int noviUpisani) {
     int ukupno = ukupanBrojStudenata + noviUpisani;
    return ukupno;          //  ukupno je lokalna promenljiva
  }
```

27

Rezime: Klase i objekti Promenljive i metode

- Klase su šabloni za kreiranje objekata (instanci klase, primeraka klase)
- U odnosu na vidljivost u klasi razlikuju se:

Promenjive:
  - Promenjive instance
  - Promenljive klase (statičke promenjive)
  - Lokalne promenljive

Metode:
  - Metode instance
  - Metode klase (statički metodi)

28

Tipovi promenljivih u Javi

1. Promenljive instance
   - vezane za pojedinačni objekat
2. Promenljive klase (static)
   - imaju jednu zajedničku vrednost za celu klasu
   - promenljive klase i instance imaju default početne vrednosti

   (numeričke – 0, logičke – *false*, znakovne – '\0', objekti – *null*
3. Lokalne promenljive
   - oblast važenja samo u okviru bloka u kome su deklarisane
     - lokalne promenljive se moraju inicijalizovati!

Tipovi metoda u Javi

1. Metodi instance
   - vezani za pojedinačni objekat
   - pozivaju se isključivo preko konkretnog objekta koji određuje konkretnu adresu u memoriji
2. Metodi klase (*static*)
   - zajednički su za celu klasu
   - metodi klase ne zahtevaju kreiranje objekta za koji će biti pozvani
   - Pozivaju se:
     - preko imena klase, iz neke druge klase ili
     - samo svojim imenom, iz klase u kojoj su definisani

Tipovi metoda u Javi

1. Metodi instance – poziv iz neke klase

ako u klasi Klasa1 postoji objekat a

```
Klasa1 a;
```

i ako u klasi Klasa1 postoji metod instance

```
promeni (int x, int y);
```

tada se on poziva na izvrsenje na objektu a instrukcijom npr.

```
a.promeni(2,5);  // poziv metoda instanc
```

Tipovi metoda u Javi

1. Metodi klase (*static*) – pozivi

```
class Klasa2 {
   static metod  ukupnoStudenata ( ) { …}
```

1. poziv iz klase u kojoj su definisani - samo svojim imenom

```
ukupnoStudenata();
```

2. poziv iz neke druge klase preko imena svoje klase

```
Klasa2. ukupnoStudenata();
```

Ključna reč this

- vraća referencu na objekat za koji je metod pozvan
- ukazuje na aktuelni objekat koji nije poznat u trenutku definicije metoda, već tek u fazi izvođenja
- koristi se samo unutar metoda

```
this.imePromenljive
```

33

Ključna reč this

Primer:

```
class Radnik {
   String ime, prezime;
   float plata;

public Radnik(String ime, String prezime,
              float plata) {
        this.ime = ime;
        this.prezime = prezime;
        this.plata = plata;
    }
```

`this.ime` – promenljiva instance ime datog objekta
`String ime` – argument metoda, lokalna promenljiva

34

Ključna reč this

- Isto značenje ima sledeći kod:

```
public Radnik(String a, String b, float x) {
            ime = a;
            prezime = b;
            plata = x;
    }
```

`ime, prezime, plata` – promenljive instance datog objekta

`a, b, x` – argumenti metoda, lokalne promenljive

35

Primer - definicija klase Radnik

Definisati klasu Radnik opisanu sa:

- dve tekst promenljive (ime, prezime), jednom realnom promenljivom (plata)
- metodom konstruktorom koji inicijalizuje sve promenljive,
- metodom za povećanje plate za zadati procenat
- metodom za prikaz svih promenljivih konkretnog radnika
- metodom za anuliranje svih promenljivih konkretnog radnika

36

Primer - definicija klase Radnik

```
public class Radnik {
    String ime,prezime;
    float plata;

    public Radnik(String ime, String prezime, float plata) {
        this.ime = ime;
        this.prezime = prezime;
        this.plata = plata;
    }
    public void povecajPlatu(float procenat){
        plata = plata * (1 + procenat/100);
    }
    public void stampaj(){
        System.out.println("Ime radnika: " + ime);
        System.out.println("Prezime radnika: " + prezime);
        System.out.println("plata radnika: " + plata);
    }
    public void anuliraj(){
        ime = null;
        prezime = null;
        plata = 0;
    }
}
```

deklaracija klase

definicije promenjivih

definicije metoda

metod konstruktor

metod 1

metod 2

metod 3

37

Primer – Upotreba klase Radnik u okviru Java aplikacije Glavna

```
public class Glavna {

    public static void main(String[] args) {
        String ime = "Petar";
        String prezime = "Petrovic";
        float plata = 50000;

    // kreiranje objekta klase Radnik
        Radnik noviRadnik =
                new Radnik(ime, prezime, plata);
    // pozivi metoda na izvršenje
        noviRadnik.stampaj();
        noviRadnik.povecajPlatu(10);
        noviRadnik.stampaj();
        noviRadnik.anuliraj();
        noviRadnik.stampaj();
    }
}
```

38