Objektno orijentirano programiranje u Javi

AUDITORNE VJEŽBE

Sadržaj

Primjer nasljeđivanja između klasa

Primjer polimorfizma

Primjer korištenja sučelja

Primjer nadjačavanja metoda u podklasama

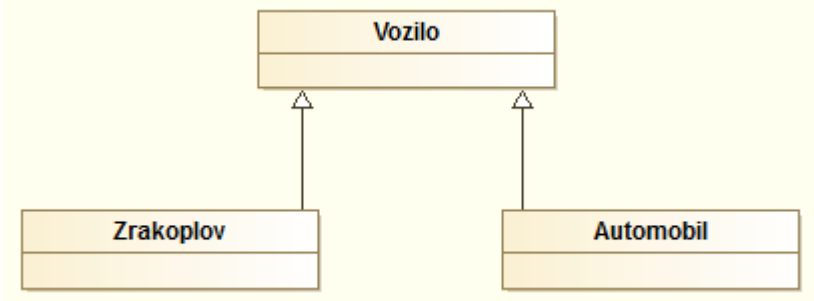
Metoda „toString”

Sortiranje elemenata u polju pomoću lambda funkcije

Primjeri pitanja s certifikata

Primjer nasljeđivanja između klasa

- Ako je potrebno kreirati hijerarhiju klasa koja uključuje entitete klase „Vozilo”, „Automobil” i „Zrakoplov”, UML Class dijagram bi izgledao ovako:



- Osnovna nadklasa „Vozilo” bi se sastojala od zajedničkih atributa za sve entitete koji predstavljaju vozilo, npr. broja putnika i potrošnje
- Klase „Zrakoplov” i „Automobil” bi nasljeđivali klasu „Vozilo” i proširili skup podataka sa svojim karakterističnim atributima kao što su vrsta motora i broj vrata (za automobil), te raspon krila i širina trupa (za zrakoplov)

Primjer nasljeđivanja između klasa

```
public class Vozilo {

   private int brojPutnika;
   private float potrosnja;

   public Vozilo(int brojPutnika, float potrosnja) {
      this.brojPutnika = brojPutnika;
      this.potrosnja = potrosnja;
   }

   //getteri i setteri
}
```

Primjer nasljeđivanja između klasa

```
public class Automobil extends Vozilo {

    public static final String VRSTA_MOTORA_BENZINSKI = "VRSTA_MOTORA_BENZINSKI";
    public static final String VRSTA_MOTORA_DIZEL = "VRSTA_MOTORA_DIZEL";
    public static final String VRSTA_MOTORA_HIBRIDNI = "VRSTA_MOTORA_HIBRIDNI";

    private String vrstaMotora;
    private int brojVrata;

    public Automobil(String vrstaMotora, int brojVrata, int brojPutnika, float potrosnja) {
        super(brojPutnika, potrosnja);
        this.vrstaMotora = vrstaMotora;
        this.brojVrata = brojVrata;
    }
    //getteri i setteri
}
```

Primjer nasljeđivanja između klasa

```
public class Zrakoplov extends Vozilo {

   private float rasponKrila;
   private float sirinaTrupa;

   public Zrakoplov(int brojPutnika, float potrosnja, float rasponKrila, float sirinaTrupa) {
      super(brojPutnika, potrosnja);
      this.rasponKrila = rasponKrila;
      this.sirinaTrupa = sirinaTrupa;
   }

   //getteri i setteri

}
```

Primjer nasljeđivanja između klasa

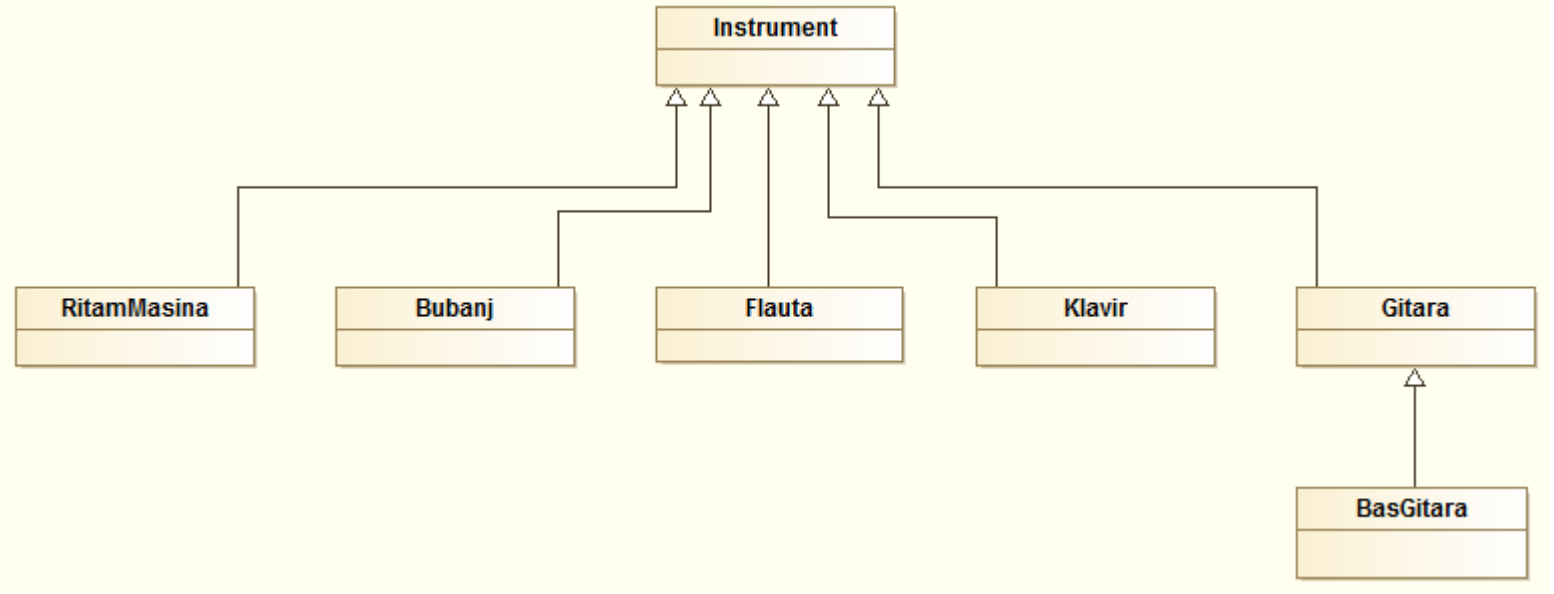
```
Automobil mojAuto = new Automobil(Automobil.VRSTA_MOTORA_DIZEL, 5, 4, 6.7f);

Zrakoplov avion = new Zrakoplov(400, 234234, 102, 36);

if (mojAuto.getBrojPutnika() > avion.getBrojPutnika()) {
   System.out.println("Ovo je jako veliki auto!!!");
}
else {
   System.out.println("Avion je mnogo veći!!!");
}
```

Primjer polimorfizma

- Polimorfizam će biti opisan na primjeru apstraktne klase „Instrument” koju nasljeđuje niz podklasa:



Primjer polimorfizma

```
public abstract class Instrument {

   public static final int VRSTA_INSTRUMENTA_PUHACKI = 1;
   public static final int VRSTA_INSTRUMENTA_UDARALJKE = 2;
   public static final int VRSTA_INSTRUMENTA_ZICANI = 3;
   public static final int VRSTA_INSTRUMENTA_ELEKTRONICKI = 4;
   public static final int VRSTA_INSTRUMENTA_KLAVIJATURE = 5;

   private int vrstaInstrumenta;

   //getter i setter za varijablu vrstaInstrumenta

   public abstract void odsvirajNotu();
}
```

Primjer polimorfizma

```
public class Gitara extends Instrument {

    public Gitara() {
        setVrstaInstrumenta(VRSTA_INSTRUMENTA_ZICANI);
    }

    @Override
        public void odsvirajNotu() {
        System.out.println("Nota odsvirana gitarom");
    }
}
```

Primjer polimorfizma

```
public class Flauta extends Instrument {

    public Flauta() {
        setVrstaInstrumenta(VRSTA_INSTRUMENTA_PUHACKI);
    }

    @Override
    public void odsvirajNotu() {
        System.out.println("Nota odsvirana flautom!");
    }
}
```

Primjer polimorfizma

```
public class Bubanj extends Instrument {

    public Bubanj() {
        setVrstaInstrumenta(VRSTA_INSTRUMENTA_UDARALJKE);
    }

    @Override
    public void odsvirajNotu() {
        System.out.println("Ritam odsviran bubnjem!");
    }
}
```

Primjer polimorfizma

```
public class Klavir extends Instrument {

    public Klavir() {
        setVrstaInstrumenta(VRSTA_INSTRUMENTA_KLAVIJATURE);
    }

    @Override
    public void odsvirajNotu() {
        System.out.println("Nota odsvirana klavirom!");
    }
}
```

Primjer polimorfizma

```
public class RitamMasina extends Instrument {

    public RitamMasina() {
        setVrstaInstrumenta(VRSTA_INSTRUMENTA_ELEKTRONICKI);
    }

    @Override
    public void odsvirajNotu() {
        System.out.println("Ritam odsviran ritam-mašinom!");
    }
}
```

Primjer polimorfizma

```
private static Instrument[] popuniPoljeRandomInstrumentima(Instrument[] instrumenti) {
    Random random = new Random();
    for (int i = 0; i < BROJ_INSTRUMENATA; i++) {
        switch (random.nextInt(5)) {
            case 0:
                instrumenti[i] = new Flauta();break;
            case 1:
                instrumenti[i] = new Bubanj();break;
            case 2:
                instrumenti[i] = new Gitara();break;
            case 3:
                instrumenti[i] = new Klavir();break;
            case 4:
                instrumenti[i] = new RitamMasina();break;
        }
    }
    return instrumenti;
}
```

Primjer polimorfizma

```
public static final int BROJ_INSTRUMENATA = 10;

public static void main(String[] args) {

    Instrument[] instrumenti = new Instrument[BROJ_INSTRUMENATA];

    instrumenti = popuniPoljeRandomInstrumentima(instrumenti);

    for (int i = 0; i < BROJ_INSTRUMENATA; i++) {
        instrumenti[i].odsvirajNotu();
        System.out.println(instrumenti[i].toString());
    }
}
```

Primjer polimorfizma

- Ispis tijekom izvršavanja programskog koda:

```
Nota odsvirana gitarom
Nota odsvirana flautom!
Ritam odsviran ritam-mašinom!
Nota odsvirana gitarom
Ritam odsviran bubnjem!
Nota odsvirana gitarom
Nota odsvirana gitarom
Nota odsvirana flautom!
Ritam odsviran bubnjem!
Nota odsvirana flautom!
```

Primjer korištenja sučelja

- Postojeće klase koje opisuju entitete koji odgovaraju karakteristikama električnih instrumenata mogu implementirati sučelje „Elektricno” koje im daje svojstvo „uključenosti” ili „isključenosti”:

```
public interface Elektricno {

    public void ukljuci();
    public void iskljuci();

}
```

Primjer korištenja sučelja

- Jedna od klasa kojoj to svojstvo „odgovara” je klasa „RitamMasina”:

```
public class RitamMasina extends Instrument implements Elektricno {

    private boolean ukljuceno;

    public RitamMasina() {
        setVrstaInstrumenta(VRSTA_INSTRUMENTA_ELEKTRONICKI);
        ukljuceno = false;
    }

    @Override
    public void odsvirajNotu() {
        if (ukljuceno == true) {
            System.out.println("Ritam odsviran ritam-mašinom!");
        }
        else {
            System.out.println("Ritam mašina je isključena!!!");
        }
    }…

    @Override
    public void ukljuci() {
        ukljuceno = true;
    }

    @Override
    public void iskljuci() {
        ukljuceno = false;
    }
}
```

Primjer nadjačavanja metoda

- Klasa „Gitara” već nasljeđuje klasu „Instrument” i nadjačava metodu „odsvirajNotu”, međutim, nju je također moguće dalje nasljeđivati i nadjačavati:

```
public class BasGitara extends Gitara {

    @Override
    public void odsvirajNotu() {
        System.out.println("Nota odsvirana bas gitarom!");
    }
}
```

Metoda „toString”

- Metodom „toString” koja je naslijeđena iz klase „java.lang.Object” moguće je definirati način na koji će se objekt klase konvertirati u String oblik
- U sklopu podrazumijevane implementacije „toString” metode ispisuje se oznaka reference objekta u memoriji, npr.:

polimorfizam.Flauta@f62373

- Ako je potrebno nadjačati to ponašanje, moguće je implementirati „toString” metodu na sljedeći način:

```
public String toString() {
  return "Flauta je vrsta instrumenta pod rednim brojem "
    + getVrstaInstrumenta();
}
```

Sortiranje elemenata u polju pomoću lambda funkcije

- Jedan od najčešćih načina za sortiranje elemenata unutar polja ili zbirke je uz korištenje implementacije sučelja „Comparator”
- To sučelje ima samo jednu apstraktnu metodu „compare” koja prima dva argumenta (npr. Stringove) i vraća jedan od mogućih rezultata:
  - Negativan cijeli broj ako je prvi argument manji od drugog
  - Nulu ako su argumenti jednaki
  - Pozitivan cijeli broj ako je prvi argument veći od drugog
- Za manipuliranje elementima polja moguće je koristiti klasu „Arrays” koja ima niz statičkih metoda s kojima je moguće unutar jedne naredbe dobiti različite funkcionalnosti
- Jedna od dostupnih metoda je i „sort” koja prima polje i definiciju kriterija pomoću kojih je moguće sortirati elemente u polju

Sortiranje elemenata u polju pomoću lambda funkcije

- Kriterije je moguće definirati pomoću klase koja implementira sučelje „Comparator” i time implementirati metodu „compare” ili koristiti lambda izraz
- Primjer u kojem se sortiraju „String” elementi je prikazan u nastavku:

```
public class Test {
    public static void main(String[] args) {
        String[] prezimena = new String[5];
        prezimena[0] = "Ivanić";
        prezimena[1] = "Žarnić";
        prezimena[2] = "Anić";
        prezimena[3] = "Perić";
        prezimena[4] = "Horvat";
        Arrays.sort(prezimena, (p1, p2) -> p1.compareTo(p2));
        System.out.println(Arrays.toString(prezimena));
    }
}
```

Primjer korištenja operatora **instanceof**

- Operator **instanceof** moguće je koristiti u slučaju kad je potrebno provjeriti je li neki objekt instanca neke klase
- Rezultat korištenja tog operatora je vraćanje vrijednosti „true” ili „false”:

```
for(Osoba osoba : osobe) {
    System.out.println("Prezime i ime: " + osoba.getPrezime() + " " + osoba.getIme());
    boolean isZaposlenik = osoba instanceof Zaposlenik;
    if(isZaposlenik) {
        System.out.println("Osoba je zaposlenik");
    }
    else {
        System.out.println("Osoba je klijent");
    }
}
```

Primjeri pitanja s certifikata

What can contain unimplemented methods and instance variables and cannot be instantiated?

A. Concrete class
B. Abstract class
C. Java class

Primjeri pitanja s certifikata

What access modifier is used to make the instance variable or method available only to the class in which it is defined?

A. `public`
B. `private`
C. `protected`
D. *package-private* (default)

Primjeri pitanja s certifikata

What is the proper signature for class X if it inherits class Z?

A. `public class X inherits Z{ … }`
B. `public class X extends Z{ … }`
C. `public class X implements Z{ … }`

How many classes can a class extend directly?

A. Zero
B. One
C. Two
D. As many as it needs

Primjeri pitanja s certifikata

Which of the following statements explain why an object can polymorphically behave as an interface?

A. By implementing the interface, the object is required to have all of the functionality that the interface represents.

B. By implementing the interface, the object inherits all the required methods it defines.

C. An object can behave as an interface because interfaces do not have a strict expected behavior and therefore any object can act as an interface.

Pitanja?