Pete auditorne vježbe

Generičko programiranje u Javi

Primjer definiranja generičke klase "Stog" (1/2)

```
public class Stog<E> {

   private final int VELICINA_STOGA = 10;
   private int vrh = 0;
   E[] elementi;

   public Stog() {
      elementi = (E[]) new Object[VELICINA_STOGA];
   }

   public void push(E element) {
      if (vrh == VELICINA_STOGA - 1) {
         throw new RuntimeException("Stog je pun!");
      }
      elementi[++vrh] = element;
   }

   public E pop() {
      if (vrh == - 1) {
         throw new RuntimeException("Stog je prazan");
      }
      return elementi[vrh--];
   }
}
```

Primjer definiranja generičke klase "Stog" (2/2)

- Iz navedene klase “Stog” kreiraju se dva različita objekta koji primaju različite parametre: Double i Integer

```
double[] doubleElements = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6 };
int[] integerElements = { 1, 2, 3, 4, 5, 6, 7, 8, 9};

Stog<Double> stog1 = new Stog<Double>();
Stog<Integer> stog2 = new Stog<Integer>();

for (Double broj : doubleElements) {
   stog1.push(broj);
}

for (Integer broj : integerElements) {
   stog2.push(broj);
}
```

Primjer definiranja generičke klase s parametrom tipa (1/6)

- U slučaju da je potrebno definirati generičku klasu kojoj će primati samo određene vrste parametara, potrebno je ograničiti podskup tipova pomoću ključne riječi **extends**
- Na primjer, neka je zadana klasa "Spremiste" koja može sadržavati objekte podklasa klase "Stvar":

```
public class Spremiste<T extends Stvar> {...}
```

- Objekte klase "Spremiste" moguće je kreirati samo ukoliko se za parametar definira klasa koja nasljeđuje klasu "Stvar"
- Unutar same klase "Spremiste" je u tom slučaju moguće koristiti tip objekta koji je definiran pomoću naziva "T" i na taj način koristiti metode koje su specifične samo za taj određeni tip

Primjer definiranja generičke klase s parametrom tipa (2/6)

- Neka su zadane klase, npr:

```
public abstract class Stvar {

    public String toString() {
        return "Stvar";
    }

}
public abstract class Odjeca extends Stvar {
    private String boja;

    public String getBoja() {
        return boja;
    }

    public void setBoja(String boja) {
        this.boja = boja;
    }
}
```

Primjer definiranja generičke klase s parametrom tipa (3/6)

```
public class Hlace extends Odjeca {

   private boolean jeans;

   public String toString() {
      return "Hlače";
   }

   public boolean isJeans() {
      return jeans;
   }

   public void setJeans(boolean jeans) {
     this.jeans = jeans;
   }
}
```

Primjer definiranja generičke klase s parametrom tipa (4/6)

```
public class Majica extends Odjeca {

    private boolean dugiRukavi;

    public String toString() {
        return "Majica";
    }

    public boolean isDugiRukavi() {
        return dugiRukavi;
    }

    public void setDugiRukavi(boolean dugiRukavi) {
        this.dugiRukavi = dugiRukavi;
    }

}
```

Primjer definiranja generičke klase s parametrom tipa (5/6)

```
public class Spremiste<T extends Stvar> {
   private List<T> listaOdjece;
   public Spremiste(int kapacitet) {
      listaOdjece = new ArrayList<T>(kapacitet);
   }
   public void dodajOdjecu(T komadOdjece) {
      listaOdjece.add(komadOdjece);
   }
   public void isprazni() {
      listaOdjece.clear();
   }
   public String toString() {
      String opisOdjece = "";
      for (T komadOdjece : listaOdjece) {
         opisOdjece += " " + komadOdjece;
         if (komadOdjece instanceof Hlace) {
            opisOdjece += " jeans : " + ((Hlace)komadOdjece).isJeans();
         }
         else if (komadOdjece instanceof Majica) {
            opisOdjece += " dugi rukavi : "+ ((Majica)komadOdjece).isDugiRukavi();
         }
         opisOdjece += "\n";
      }
      return opisOdjece;
   }
}
```

Primjer definiranja generičke klase s parametrom tipa (6/6)

- Ako se kreira objekt klase "Spremiste" s parametrom "Odjeca", znači da je time ograničen skup tipova objekata koji mogu biti dodani u "spremište":

```
Spremiste<Odjeca> ormar = new Spremiste<>(2);
Hlace hlace1 = new Hlace();
hlace1.setJeans(true);
Majica majica1 = new Majica();
majica1.setDugiRukavi(true);
ormar.dodajOdjecu(hlace1);
ormar.dodajOdjecu(majica1);
```

- Osim toga moguće je kreirati i objekt klase "Spremiste" koji prima samo objekte podklase "Stvar":

```
Spremiste<Stvar> kutija = new Spremiste<Stvar>(10);
```

Primjer korištenja zamjenskog simbola

- U slučaju da je potrebno napisati metodu koja će isprazniti svako "spremište", bez obzira koje objekte ono sadržavalo, to je moguće napraviti na sljedeći način:

```
private static void isprazniSpremiste(
                    Spremiste<? extends Stvar> spremiste) {
    spremiste.isprazni();
}
```

- Pomoću znaka "?" omogućeno je metodi predati sve objekte klase "Spremiste" koji u sebi sadrže objekte podklase klase "Stvar"
- Pomoću te metode moguće je "isprazniti" objekte "ormar" i "kutija"
- Ako bi umjesto "? extends Stvar" napisali "? extends Odjeca", objekt "kutija" ne bi bilo moguće "isprazniti"

Korištenje generičkih metoda

- Moguće je napisati i generičku metodu koja će ispravno funkcionirati samo sa zadanim tipovima, npr:

```
private static <T extends Odjeca> void popuniOdjecom(
                    Spremiste<T> spremiste, List<T> listaOdjece) {
   for (T odjeca : listaOdjece) {
      spremiste.dodajOdjecu(odjeca);
   }
}
```

- Tu metodu moguće je koristiti samo s podklasama klase "Odjeca":

```
Spremiste<Odjeca> ladica = new Spremiste<>(2);
List<Odjeca> odjeca = new ArrayList<Odjeca>();
listaOdjece.add(hlace1);
listaOdjece.add(majica1);
popuniOdjecom(ladica, odjeca);
```