Java tečaj

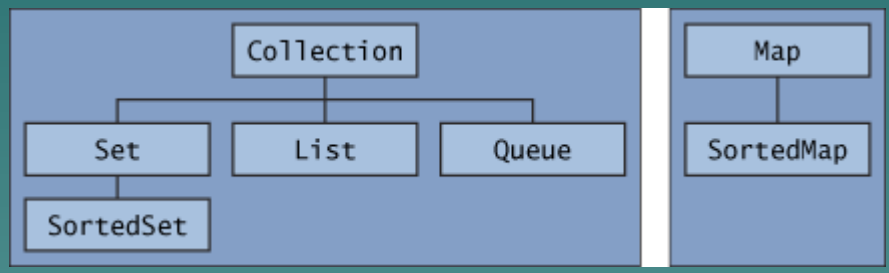
4. dio
Kolekcije, 2. dio
Java Generics

Kolekcije

- Nastavimo dalje s kolekcijama…
- Prošli puta smo vidjeli osnovne tipove kolekcija u Javi, te standardne implementacije, kao i njihova svojstva
  - Npr. složenost pojedinih operacija

Kolekcije: sučelja

- Sučelja koja definiraju kolekcije



- Kolekcija – najopćenitija grupa elemenata

Kolekcije

- Kako se kolekcije ponašaju s novim razredima?
- Primjerice, definirajmo razred Zaposlenik
- Svaki zaposlenik ima svoju sifru (jedinstveno), prezime, ime te plaću

Kolekcije

```
package hr.fer.zemris.java.tecaj_4;

public class Zaposlenik {
	private String sifra;
	private String prezime;
	private String ime;
	private double placa;

	public Zaposlenik(String sifra, String prezime, String ime) {
		super();
		this.sifra = sifra;
		this.prezime = prezime;
		this.ime = ime;
	}

	public double getPlaca() {
		return placa;
	}

	public void setPlaca(double placa) {
		this.placa = placa;
	}
	…
}
```

Kolekcije

- Zbog lakšeg ispisa dodajmo metodu toString() u Zaposlenik

```
public String toString() {
        return "Zaposlenik: sifra="+sifra
                        +", prezime="+prezime
                        +", ime="+ime+", placa="+placa;
}
```

Kolekcije

- Zanima nas kako će se kolekcija “lista” ponašati s ovim novim razredom
- Napravimo program koji će dodati nekoliko zapisa u listu, i potom ih sve ispisati

Kolekcije

```
package hr.fer.zemris.java.tecaj_4;

import java.util.ArrayList;
import java.util.List;

public class DohvatZaposlenika {

    public static void main(String[] args) {
        List<Zaposlenik> lista = new ArrayList<Zaposlenik>();

        lista.add(new Zaposlenik("1","Peric","Pero"));
        lista.add(new Zaposlenik("2","Agic","Agata"));
        lista.add(new Zaposlenik("3","Ivic","Ivana"));

        for(Zaposlenik zap : lista) {
            System.out.println(zap);
        }
    }

}
```

Kolekcije

- Rezultat izvođenja

Zaposlenik: sifra=1, prezime=Peric, ime=Pero, placa=0.0
Zaposlenik: sifra=2, prezime=Agic, ime=Agata, placa=0.0
Zaposlenik: sifra=3, prezime=Ivic, ime=Ivana, placa=0.0

Kolekcije

- Proširimo program još s nekoliko linija koda:

```
Zaposlenik zaposlenik =
        new Zaposlenik("1","Peric","Pero");
boolean sadrziZaposlenika = lista.contains(zaposlenik);
System.out.println("SadrziZaposlenika = "
                + sadrziZaposlenika);
```

- Što će biti rezultat?

Kolekcije

- Rezultat je false!
- Zašto? Razmislite!
- Ekvivalentno pitanje: jesmo li mogli zaposlenika tražiti metodom indexOf, i pronaći ga?

Kolekcije

- U razred Zaposlenik treba dodati metodu equals() koja govori kada su dva primjerka ista (ma što termin ista mogao značiti)!

```
public boolean equals(Object arg0) {
        if(arg0==null) return false;
        if(!(arg0 instanceof Zaposlenik)) return false;
        Zaposlenik drugi = (Zaposlenik)arg0;
        return
           Long.valueOf(sifra).equals(Long.valueOf(drugi.sifra))
           && prezime.equals(drugi.prezime)
           && ime.equals(drugi.ime);
}
```

Kolekcije

- Ponovimo li sada program, rezultat izvođenja je true!

- Pravilo:

  Kako bi se omogućilo ispravno pretraživanje kolekcija metodom usporedbe na jednakost, potrebno je implementirati metodu equals

Kolekcije

- Što se događa ako želimo zaposlenike dodavati u, primjerice, TreeSet?
- Pokušajmo!

- Program: ZaposleniciTree.java

Kolekcije

- Koji je rezultat izvođenja?

Exception in thread "main" java.lang.ClassCastException:
hr.fer.zemris.java.tecaj_4.Zaposlenik
at java.util.TreeMap.compare(Unknown Source)
at java.util.TreeMap.put(Unknown Source)
at java.util.TreeSet.add(Unknown Source)
at
hr.fer.zemris.java.tecaj_4.ZaposleniciTree.main(ZaposleniciTree.java:16)

- Zašto?

Kolekcije

- TreeSet objekte pokušava sortirati!
- Da bi to mogao, mora znati kako usporediti dva objekta – a u našem slučaju to nije jasno.
- Možemo napraviti vlastiti komparator zaposlenika
  - Razred koji implementira java.util.Comparator

Kolekcije

```
interface java.util.Comparator<T> {
  int compare(T arg0, T arg1);
}
```

- Metoda vraća:
  - Negativan broj ako je arg0 < arg1
  - Pozitivan broj ako je arg0 > arg1
  - 0 inače (dakle, ako je arg0 jednak arg1)

Kolekcije

- Primjer: ZaposleniciTree2.java

- Eksplicitno se definiraju dva komparatora: jednostavan i kompozitni (složen)
- Konstruktor TreeSet prima argument na komparator koji zna usporediti objekte

Kolekcije

- Međutim, što je s prirodnim poretkom? Zašto smo Integere i Stringove mogli dodavati u TreeSet?

- Ovi razredi imaju definiran "prirodni" poredak ➔ implementiraju sučelje java.lang.Comparable!

Kolekcije

```
interface java.lang.Comparable {
  int compareTo(T arg);
}
```

- Metoda vraća:
  - Negativan broj ako je this < arg
  - Pozitivan broj ako je this > arg
  - 0 inače (dakle, ako je this jednak arg)

Kolekcije

- Dakle, Zaposlenik-a treba proširiti:

```
public class Zaposlenik implements Comparable {
…
}
```

Kolekcije

- I potom dodati compareTo:

```
public int compareTo(Object arg0) {
        if(arg0==null) return 1;
        Zaposlenik drugi = (Zaposlenik)arg0;
        long res = Long.parseLong(sifra)
                            - Long.parseLong(drugi.sifra);
        if(res<0) return -1;
        else if(res>0) return 1;
        else return 0;
}
```

Kolekcije

- Ovime smo definirali poredak koji je određen kao uzlazni slijed šifri
- Sada možemo napraviti isti primjer dodavanja u TreeSet bez da sami navodimo komparator.

- Primjer: ZaposleniciTree3.java

Kolekcije

- Pravilo:

  Kako bi se omogućio ispravan rad kolekcija koje za svoj rad koriste usporedbu objekata (veće, manje, jednako), potrebno je implementirati sučelje Comparable, ili ponuditi vanjski Comparator.

Kolekcije

- Što nam treba za ispravan rad s mapama?
- Pogledajmo primjer: Bonus.java

Kolekcije

```
public class Bonus {

    public static void main(String[] args) {

        Map<Zaposlenik,Double> bonusi = new HashMap<Zaposlenik,Double>();

        bonusi.put(new Zaposlenik("1","Peric","Pero"),Double.valueOf(100.00));
        bonusi.put(new Zaposlenik("2","Agic","Agata"),Double.valueOf(200.00));
        bonusi.put(new Zaposlenik("3","Ivic","Ivana"),Double.valueOf(300.00));

        for(Zaposlenik zap : bonusi.keySet()) {
                Double bonus = bonusi.get(zap);
                System.out.println(zap.toString()+" ima bonus od "+bonus+" kn.");
        }
    }
}
```

Kolekcije

- Što je rezultat dodavanja sljedećeg koda?

```
Double bonusOd1 =
        bonusi.get(new Zaposlenik("1","Peric","Pero"));
System.out.println(
        "Bonus zaposlenika cija je sifra 1 je “
        +bonusOd1+" kn.");
```

Kolekcije

- Rezultat je false!
- Zašto? Razmislite!

Kolekcije

- Radi se, naravno, o metodi hashCode koju je potrebno implementirati u Zaposleniku
- Npr:

```
public int hashCode() {
  return Long.valueOf(sifra).hashCode()
                    ^ prezime.hashCode()
                    ^ ime.hashCode();
}
```

Kolekcije

- Da bi sve radilo kako spada, treba paziti na vezu između metoda equals i hashCode
  - Za objekte koje equals proglasi istima, hashCode mora također dati identične vrijednosti
- Ako sada ponovimo prethodni primjer, rezultat će biti true

Kolekcije

- Pravilo:

  Da bi mogli koristiti vlastite razrede u mapama, nužno je ispravno implementirati metode equals i hashCode.

Kolekcije

- Za rad s poljima također imamo na raspolaganju gotove metode!
- java.util.Arrays
- Primjerice, sortiranje polja…

- Primjer: SortiranjePolja.java

Kolekcije

- Uočiti da se u primjeru koriste anonimni razredi!
- Drugi argument metode sort je primjerak nekog razreda koji implemetira sučelje Comparator<T>
- Stoga možemo na mjestu gdje je to potrebno stvoriti anonimni razred…

Kolekcije

```
Arrays.sort(
    zaposlenici,
    new Comparator<Zaposlenik>() {
      // implementacija metoda ovog
      // razreda
    }
)
```

Kolekcije

- Za rad s kolekcijama imamo na raspolaganju razred java.util.Collections
- Primjeri sortiranja lista, reverznog poretka i sl.
- Primjer: SortiranjeListe.java

Java Generics

- Od Jave 5.0 → uvedena podrška "template"-ima
- Primjer: želimo napraviti razred Pointer, za prijenos argumenata "po adresi"

Java Generics

```
package hr.fer.zemris.java.tecaj_4;

public class Pointer<T> {

	private T object;

	public <X extends T> Pointer(X object) {
		super();
		this.object = object;
	}

	public T getObject() {
		return object;
	}

	public <X extends T> void setObject(X object) {
		this.object = object;
	}
}
```

Java Generics

- Sada možemo pisati:

```
Integer x = new Integer(10);
Pointer<Integer> p = new Pointer<Integer>(x);
// posalji pointer u metodu…
x = p.getObject();  // po tipovima kompatibilno!
```

Java Generics

```
public class ProbaPointera {

    public static void main(String[] args) {
        Integer broj1 = new Integer(10);
        System.out.println("Broj1 (prije) = "+broj1);
        uvecaj1(broj1);
        System.out.println("Broj1 (nakon) = "+broj1);
    }

    private static void uvecaj1(Integer broj1) {
        broj1 = Integer.valueOf(broj1.intValue()+1);
    }

}
```

Java Generics

- Što će se ispisati na zaslonu?

- Pokušajmo s razredom pointer…

Java Generics

```
public class ProbaPointera {

    public static void main(String[] args) {
        Integer broj1 = new Integer(10);

        Pointer<Integer> pBroj1 = new Pointer<Integer>(broj1);
        System.out.println("Broj1 (prije) = "+broj1);
        uvecaj2(pBroj1);
        broj1 = pBroj1.getObject();
        System.out.println("Broj1 (nakon) = "+broj1);
    }

    private static void uvecaj2(Pointer<Integer> pBroj1) {
        pBroj1.setObject(Integer.valueOf(pBroj1.getObject().intValue()+1));
    }
}
```

Java Generics

```
public class ProbaPointera {

    // …

    private static void uvecaj2(Pointer<Integer> pBroj1) {
          pBroj1.setObject(Integer.valueOf(pBroj1.getObject().intValue()+1));
    }

    // ekvivalent metodi uvecaj2:

    private static void uvecaj2_1(Pointer<Integer> pBroj1) {
          Integer broj1 = pBroj1.getObject();
          broj1 = Integer.valueOf(broj1.intValue()+1);
          pBroj1.setObject(broj1);
    }

    // …
}
```

Kolekcije

- Obavezno proučiti:
- http://www.javaworld.com/javaworld/jw-04-2004/jw-0426-tiger1.html
  - Sva tri dijela! (Part1, Part2, Part3)
- Pogledati src.zip u direktoriju gdje je instalirana Java – unutra su svi sourcevi!
- http://java.sun.com/developer/technicalArticles/J2SE/generics/