

# Java tečaj

3. dio

Kolekcije, 1. dio

# Kolekcije

- ◆ Svi programi imaju jedno zajedničko svojstvo – obrađuju podatke
- ◆ Podaci su često grupirani kroz različite strukture podataka
- ◆ Za grupe podataka Java koristi pojam Kolekcija (engl. Collection)
- ◆ Važno: osim primitivnih tipova, sve ostalo u Javi su objekti (primjerci nekog razreda, koji je izveden iz razreda Object)

# Kolekcije

- ◆ **Kolekcija** – objekt koji grupira više elemenata – drugih objekata (spremnik, container)
- ◆ Omogućava
  - Pohranu podataka
  - Dohvat podataka
  - Manipulaciju podataka
  - Razmjenu agregacija podataka

# Kolekcije

- ◆ **Kolekcija** – objekt koji grupira više elemenata – nešto tipa:

```
class Kolekcija {  
    void dodajPodatak( podatak ) {...}  
    podatak dohvatiPodatak( kriterij ) {...}  
    void zamijeniPodatke( kriterij1, kriterij2 ) {...}  
    void ukloniPodatak( kriterij ) {...}  
}
```

- ◆ U Javi je kolekcija zapravo definirana preko sučelja

# Kolekcije

- ◆ **Sučelje (engl. Interface):** popis metoda koje razred **mora** implementirati

```
interface Kolekcija {  
    void dodajPodatak( podatak );  
    podatak dohvatiPodatak( kriterij );  
    void zamijeniPodatke( kriterij1, kriterij2 );  
    void ukloniPodatak( kriterij );  
}
```

# Kolekcije

## ◆ Implementacije sučelja:

```
class MemorijskaKolekcija implements Kolekcija {  
    void dodajPodatak( podatak ) {...}  
    podatak dohvatiPodatak( kriterij ) {...}  
    void zamijeniPodatke( kriterij1, kriterij2 ) {...}  
    void ukloniPodatak( kriterij ) {...}  
}
```

## ◆ Sve podatke čuva u memoriji

# Kolekcije

## ◆ Implementacije sučelja:

```
class DiskKolekcija implements Kolekcija {  
    void dodajPodatak( podatak ) {...}  
    podatak dohvatiPodatak( kriterij ) {...}  
    void zamijeniPodatke( kriterij1, kriterij2 ) {...}  
    void ukloniPodatak( kriterij ) {...}  
}
```

## ◆ Sve podatke čuva na disku

# Kolekcije

## ◆ Zašto definicija kroz sučelja?

- Omogućava više različitih implementacija iste funkcionalnosti (sučelje gledati kao opis – što razred obećava da radi; ne i kako će to napraviti)
- Omogućava laganu reimplementaciju ukoliko se postojeća pokaže lošom
- Omogućava izradu generičkih algoritama, koji rade sa svim razredima koji implementiraju propisano sučelje, neovisno o samoj implementaciji



# Kolekcije

- ◆ **Javin okvir kolekcija** (Java collection framework) sastavljen je od:
  - **Sučelja** – rad s kolekcijama neovisno o samoj implementaciji
  - **Implementacija** – konkretne implementacije sučelja (razredi)
  - **Algoritama** – metode za sortiranje, pretraživanje, i sl.

# Kolekcije

- ◆ Prednosti uporabe okvira kolekcija
  - Smanjivanje potrebne količine koda
  - Povečanje brzine i kvalitete rada
  - Ubrzano učenje
  - Olakšan razvoj novih API-ja
  - Promovira višestruku iskoristivost koda

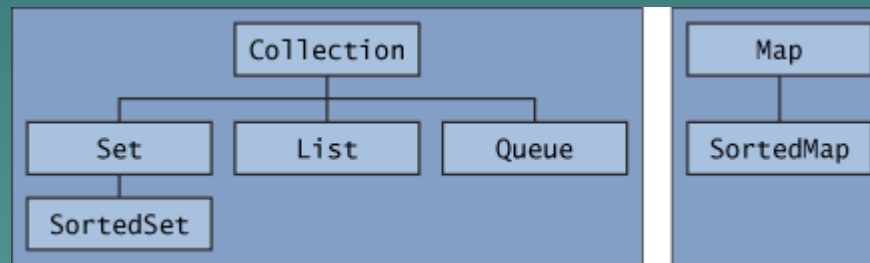
# Kolekcije

- ◆ Jedna od najvažnijih operacija nad kolekcijama: obilazak svih elemenata  
→ iteriranje
- ◆ U pseudokodu to izgleda ovako:

```
Iterator it = kolekcija.dohvatiIterator();  
Ponavljaj dok it.postojiSljedeci()  
    obradi( it.dohvatiSljedeceg() );  
Kraj ponavljaj
```

# Kolekcije: sučelja

- ◆ Koje sve vrste kolekcija postoje?
- ◆ Definirana sučelja



- ◆ Kolekcija – najopćenitija grupa elemenata

# Kolekcije: sučelje Collection

```
public interface Collection<E> extends Iterable<E> {
```

```
    //Basic operations
```

```
    int size();
```

```
    boolean isEmpty();
```

```
    boolean contains(Object element);
```

```
    boolean add(E element); //optional
```

```
    boolean remove(Object element); //optional
```

```
    Iterator<E> iterator();
```

# Kolekcije: sučelje Collection

//Bulk operations

boolean containsAll(Collection<?> c);

boolean addAll(Collection<? extends E> c); //optional

boolean removeAll(Collection<?> c); //optional

boolean retainAll(Collection<?> c); //optional

void clear(); //optional

//Array operations

Object[] toArray();

<T> T[] toArray(T[] a);

}

# Kolekcije: sučelje Collection

- ◆ Metode označene kao opcionalne ne moraju biti podržane od konkretne implementacije
- ◆ Ako metoda nije podržana, bit će izazvana `UnsupportedOperationException`
- ◆ Samo sučelje `Collection` nema direktnih implementacija

# Kolekcije: sučelje Set

- ◆ Skup (engl. Set) je kolekcija koja ne može sadržavati duplikate
- ◆ Sadrži sve metode koje propisuje Collection sučelje, i dodaje gornji uvjet



# Kolekcije: sučelje Set

- ◆ Paket `java.util` sadrži sljedeće implementacije ovog sučelja
  - `HashSet` – najbrže, ne garantira poredak kod obilaska elemenata
  - `TreeSet` – najsporije, ali elemente obilazi sortiranim poretkom
  - `LinkedHashSet` – malo sporije od `HashSet`, elemente obilazi poretkom kojim su ubačeni

# Kolekcije: sučelje Set

- ◆ Odabir koju od implementacija koristiti u određenom trenutku ovisi o zahtjevanim performansama i načinu uporabe (dominantno čitanje, dodavanje, iteriranje i sl)

# Kolekcije: sučelje Set

- ◆ **Primjer.** Napisati program koji će iz komandne linije primiti niz argumenata, te će na zaslon ispisati argumente ali bez ponavljanja duplikata
- ◆ Tipičan primjer uporabe skupova

# Kolekcije: sučelje Set

```
package hr.fer.zemris.java.tecaj_3;
```

```
import java.util.Collection;
```

```
import java.util.HashSet;
```

```
import java.util.LinkedHashSet;
```

```
import java.util.Set;
```

```
import java.util.TreeSet;
```

```
...
```

A stylized, dark teal silhouette of a mountain range is positioned in the bottom right corner of the slide, extending from the right edge towards the center.

# Kolekcije: sučelje Set

```
public class PrimjerSkupa {  
  
    public static void main(String[] args) {...}  
  
    private static void ispisiSkup(Collection<String> col) {...}  
    private static void ispisiSkup2(Collection<String> col) {...}  
  
    private static Collection<String> ukloniDuplikate1(String[]  
    polje) {...}  
    private static Collection<String> ukloniDuplikate2(String[]  
    polje) {...}  
    private static Collection<String> ukloniDuplikate3(String[]  
    polje) {...}  
}
```

# Kolekcije: sučelje Set

```
public static void main(String[] args) {  
    System.out.println("Preko HashSet-a:");  
    ispisiSkup(ukloniDuplikate1(args));  
    System.out.println();  
  
    System.out.println("Preko TreeSet-a:");  
    ispisiSkup(ukloniDuplikate2(args));  
    System.out.println();  
  
    System.out.println("Preko LinkedHashSet-a:");  
    ispisiSkup(ukloniDuplikate3(args));  
    System.out.println();  
}
```

# Kolekcije: sučelje Set

```
private static void ispisiSkup(Collection<String> col) {  
    for (String element : col) {  
        System.out.println(element);  
    }  
}  
  
private static void ispisiSkup2(Collection<String> col) {  
    Iterator<String> iterator = col.iterator();  
    while(iterator.hasNext()) {  
        System.out.println(iterator.next());  
    }  
}
```

# Kolekcije: sučelje Set

```
private static Collection<String> ukloniDuplikate1(String[]  
polje) {  
    Set<String> set = new HashSet<String>();  
    for (String element : polje) {  
        set.add(element);  
    }  
    return set;  
}
```



# Kolekcije: sučelje Set

```
private static Collection<String> ukloniDuplikate2(String[]  
polje) {  
    Set<String> set = new TreeSet<String>();  
    for (String element : polje) {  
        set.add(element);  
    }  
    return set;  
}
```

# Kolekcije: sučelje Set

```
private static Collection<String> ukloniDuplikate3(String[]  
polje) {  
    Set<String> set = new LinkedHashSet<String>();  
    for (String element : polje) {  
        set.add(element);  
    }  
    return set;  
}
```

# Kolekcije: sučelje Set

## ◆ Rezultat izvođenja

PrimjerSkupa ovo prvo je jako vazno a i ovo drugo isto

Preko HashSet-a:	Preko TreeSet-a:	Preko LinkedHashSet-a:
ovo	a	ovo
isto	drugo	prvo
i	i	je
vazno	isto	jako
a	jako	vazno
prvo	je	a
jako	ovo	i
drugo	prvo	drugo
je	vazno	isto

# Kolekcije: sučelje Set

- ◆ Sintaksa `Set<Tip>` uvedena je u Javi 5.0 (takozvani Genericsi)

```
Set<String> set1 = new HashSet<String>();  
set1.add("Pero");  
set1.add("Štefica");
```


```
Set<Integer> set2 = new HashSet<Integer>();  
set2.add(Integer.valueOf(5));  
set2.add(Integer.valueOf(7));
```

# Kolekcije: sučelje Set

- ◆ Sintaksa Set<Tip> uvedena je u Javi 5.0 (takozvani Genericsi)

```
Set<String> set3 = new HashSet<String>();  
set1.add("Pero");  
set1.add(new Integer(8)); // Compile-time GRESKA!!!
```

# Kolekcije: sučelje Set

- ◆ Napisati program koji prima imena dviju datoteka preko komandne linije
  - ◆ Svaki redak datoteka sadrži jedno ime
  - ◆ Program treba ispisati sva imena koja se nalaze u prvoj datoteci, a nisu u drugoj datoteci
  - ◆ Datoteke su relativno male
- 
- A stylized, dark teal mountain range graphic is located in the bottom right corner of the slide, extending from the right edge towards the center.

# Kolekcije: sučelje List

- ◆ Lista (engl. List) ili slijed (engl. Sequence) je uređena kolekcija koja može sadržavati duplikate
- ◆ Elementi imaju svoju poziciju
- ◆ Može se dohvatiti element na zadanoj poziciji
- ◆ Može se umetnuti element na zadanu poziciju

# Kolekcije: sučelje List

```
public interface List<E> extends Collection<E> {  
    //Positional access  
    E get(int index);  
    E set(int index, E element);    //optional  
    boolean add(E element);        //optional  
    void add(int index, E element); //optional  
    E remove(int index);           //optional  
    abstract boolean addAll(int index,  
        Collection<? extends E> c); //optional  
  
    //Search  
    int indexOf(Object o);  
    int lastIndexOf(Object o);
```



# Kolekcije: sučelje List

//Iteration

ListIterator<E> listIterator();

ListIterator<E> listIterator(int index);

//Range-view

List<E> subList(int from, int to);

}

A stylized, dark teal silhouette of a mountain range is positioned in the bottom right corner of the slide, extending from the right edge towards the center.

# Kolekcije: sučelje List

- ◆ Java nudi dvije implementacije sučelja List:
  - ArrayList – lista koja za pohranu elemenata koristi dinamičko polje
  - LinkedList – lista koja elemente pohranjuje dinamički alocirajući nove čvorove

# Kolekcije: sučelje List

- ◆ Napisati funkciju koja će primiti polje stringova, i ispisati sve elemente obrnutim poretkom, uz izbacivanje duplikata

# Kolekcije: sučelje List

```
private static void ispisiObrnuto(String[] args) {  
  
    Set<String> set = new LinkedHashSet<String>();  
    for(String element: args) {  
        set.add(element);  
    }  
  
    List<String> list = new ArrayList<String>(set);  
    Collections.reverse(list);  
  
    for(String element: list) {  
        System.out.println(element);  
    }  
}
```

# Kolekcije: sučelje List

- ◆ Napisati program koji s tipkovnice čita niz decimalnih brojeva (jedan broj po retku); quit prekida unos
- ◆ Program ispisuje samo one brojeve koji barem za 20% veći od prosjeka svih brojeva
- ◆ Ispis brojeva je od manjih prema većim

# Kolekcije: sučelje Map

- ◆ Služi za pohranu tipa Ključ-Vrijednost
- ◆ Primjeri mapiranja:
  - Osoba – Telefonski broj
  - Matični broj studenta – Prosjek ocjena
  - ...

# Kolekcije: sučelje Map

```
public interface Map<K,V> {  
  
    //Basic operations  
    V put(K key, V value);  
    V get(Object key);  
    V remove(Object key);  
    boolean containsKey(Object key);  
    boolean containsValue(Object value);  
    int size();  
    boolean isEmpty();  
  
    //Bulk operations  
    void putAll(Map<? extends K,? extends V> t);  
    void clear();  
  
    ...  
}
```

# Kolekcije: sučelje Map

```
// Collection Views  
public Set<K> keySet();  
public Collection<V> values();  
public Set<Map.Entry<K,V>> entrySet();
```

```
// Interface for entrySet elements
```

```
public interface Entry {  
    K getKey();  
    V getValue();  
    V setValue(V value);
```

```
}
```

```
}
```



# Kolekcije: sučelje Map

## ◆ Primjer uporabe:

```
Map<String,Integer> ocjena =  
    new HashMap<String,Integer>();  
ocjena.put("Ana", Integer.valueOf(5));  
ocjena.put("Ivo", Integer.valueOf(4));  
ocjena.put("Ivana", Integer.valueOf(5));  
  
System.out.println(  
    "Ivana ima ocjenu: "+ocjena.get("Ivana")  
);
```

# Kolekcije: sučelje Map

- ◆ Ugrađene su tri implementacije:
  - **HashMap** – najčešće implementacija odabira kada su u pitanju performanse
  - TreeMap
  - LinkedHashMap

# Kolekcije: sučelje Map

- ◆ **Primjer.** Napisati program koji će s tipkovnice čitati niz imena. Program završava ako se za ime unese "quit". Program prije završetka mora na zaslon ispisati sva upisana imena i uz svako ime koliko je puta upisano.

# Kolekcije: sučelje Map

- ◆ Ovo je tipičan primjer uporabe tablice raspršenog adresiranja (HashMap)!
- ◆ `HashMap<String,Integer>`
  - Za svako ime pamti koliko puta je uneseno
- ◆ Na kraju obilazak iteracijom po ključevima...

# Kolekcije: razred Collections

- ◆ Kao potpora radu s kolekcijama:
  - `Java.util.Collections` – niz generičkih metoda za rad s kolekcijama (općenitim, i specijalnim slučajevima)
  - Više o ovome sljedeći puta