11. Kolekcije

Sadržaj

- Liste i mape
- Generičke klase

Intefejsi kolekcija

- Paket `java.util` sadrži interfejse i klase koje omogućavaju rad sa kolekcijama objekata
- Interfejs `Collection` - proizvoljna kolekcija objekata
  - `add`
  - `remove`
  - `isEmpty`
  - `size`
- Interfejs `List` - kolekcija objekata sa utvrđenim poretkom
- Interfejs `Set` - kolekcija jedinstvenih objekata

Klasa `ArrayList`

- Najčešće korišćena klasa kolekcije
- Implementira interfejs `List` što znači da garantuje poredak elemenata
- Zamena za nizove čiji je nedostatak fiksna dužina
- Korišćenje niza za smeštanje reference na sve oblike na crtežu bi predstavljalo problem jer ne možemo da predvidimo koliko će oblika korisnik dodati na crtež

Dodavanje elemenata u `ArrayList`

- Kreiranje liste elemenata tipa `Object`

```
ArrayList myList = new ArrayList();
```

- Dodavanje elementa na kraj liste

```
myList.add(“Marko”);              // prvi element
myList.add(new Point(5, 6));      // drugi element
```

- Dodavanje elementa na željenu poziciju u listi

```
myList.add(0, “Ana”);     // na prvo mesto, ostale pomera
```

Čitanje elemenata iz `ArrayList`

- Čitanje elementa iz liste sa zadate pozicije

```
Object firstElement = myList.get(0);
```

- Prolazak kroz kompletnu listu

```
Iterator it = myList.iterator();
while(it.hasNext()) {
  Object obj = it.next();
  System.out.println(obj);
}
```

- `Iterator` poseduje metodu `remove()` koja uklanja poslednji vraćeni element

Uklanjanje elemenata iz `ArrayList`

- Uklanjanje elementa sa zadate pozicije

```
myList.remove(1);     // sve sledece pomera prema pocetku
```

- Uklanjanje zadatog elementa

```
myList.remove("Ana"); // sve sledece pomera prema pocetku
```

- Pražnjenje kompletne liste

```
myList.clear();
```

- Provera da li je lista prazna

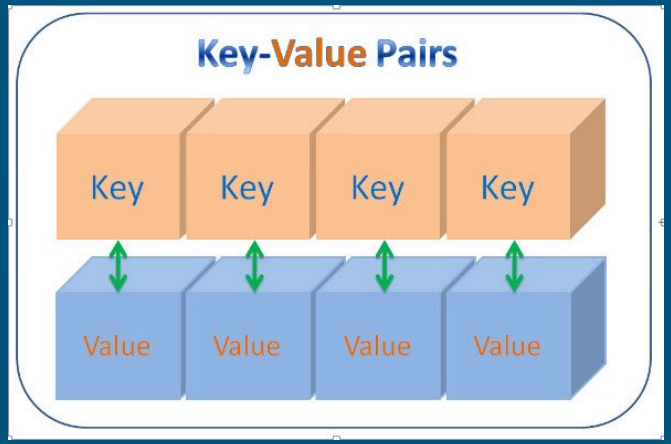
```
if (myList.isEmpty())
   …
```

- Provera broja elemenata u listi

```
int numberOfElements = myList.size();
```

Mape

- Interfejs `Map` definiše operacije nad podacima koje imaju skup ključeva (`Set`) i kolekciju vrednosti (`Collection`)
- `HashMap` klasa za generisanje ključa koristi metodu `hashCode` objekta



Dodavanje elemenata u `HashMap`

- Kreiranje mape

```
HashMap myMap = new HashMap();
```

- Dodavanje elemenata u mapu

```
myMap.put(“firstKey”, new Point(7, 8));
myMap.put(“secondKey”, new Circle(p1, 22));
```

- Dodavanje elementa sa postojećim ključem “gazi” postojeću vrednost u mapi

```
myMap.put(“firstKey”, new Line(p2, p3)); // gazi tacku 7,8
```

Čitanje elemenata iz `HashMap`

- Provera postojanja ključa

```
Set keys = myMap.keySet();
if (keys.contains("firstKey"))
   ...
```

- Čitanje vrednosti

```
Object valueFromMap = myMap.get("firstKey");
```

- Prolazak kroz sve vrednosti mape

```
Iterator it = myMap.values().iterator();
while (it.hasNext())
   ...
```

Poređenje kolekcija `ArrayList` i `HashMap`

`ArrayList`

- garantuje poredak elemenata
- može da sadrži identične elemente
- performanse pristupa jednom elementu mogu biti loše kada lista ima mnogo elemenata

`HashMap`

- ne garantuje poredak ključeva niti vrednosti
- sprečava smeštanje dva elementa sa istim ključem
- performanse pristupa jednom elementu su dobre i kada mapa ima mnogo elemenata

Generičke klase

- Prethodno opisane kolekcije rade sa svim tipovima referenci na objekte
- Ukoliko želimo da kompajler proveri da li se u kolekcijama operiše sa objektima odgovarajućeg tipa, koristimo generičke klase
- Pretpostavimo da lista treba da sadrži samo objekte klasa izvedenih iz apstraktne klase `Shape`

```
ArrayList<Shape> list = new ArrayList<Shape>();
list.add(new Point(6, 9)); // ok, Point prosiruje Shape
list.add("abc");           // kompajler javlja gresku
```

Prolazak kroz generičku listu

- Nema potrebe za konverzijom tipa

```
Iterator<Shape> it = list.iterator();
while (it.hasNext() {
  Shape s = it.next();  // ne treba downcast
  s.draw(g);
}
```

Automatska konverzija primitiva

- U kolekciju koja je deklarisana za *wrapper* tip moguće je raditi direktno sa primitivnim tipom

```
ArrayList<Integer> li = new ArrayList<Integer>();
li.add(new Integer(8));
li.add(5);                  // automatska konverzija u Integer
…
int num = li.get(0);        // automatska konverzija u int
```

Rezime

- Kolekcije omogućavaju različite načina upravljanja podacima
- Generičke klase omogućavaju proveru tipa elemenata kolekcije u vreme kompajliranja
- Kolekcije omogućavaju automatsku konverziju primitivnog tipa u ovojnu klasu i obrnuto