Generički tipovi podataka

Generičke metode

- Problem: napisati metode za ispis vrednosti niza: celih brojeva, realnih brojeva, Stringova itd?
- Rešenje 1:

```
static void ispisi(Integer[] arr){…}
static void ispisi(Float[] arr){…}
static void ispisi(String[] arr){...}
```

Generičke metode

- Rešenje 2 (upotreba generičkih tipova):

```
static <T> void ispisi(T[] arr){
        for(int i=0; i<arr.length; i++)
                System.out.println(arr[i]);
}
```

Uslovljeni generički tipovi

- Problem: napisati metode za traženje maksimuma 3 cela broja, 3 realna broja, 3 String-a i 3 učenika po proseku?
- Rešenje 1:

```
static Integer maks3(Integer x, Integer y, Integer z){…}
static Float maks3(Float x, Float y, Float z){…}
static String maks3(String x, String y, String z){…}
static Ucenik maks3(Ucenik x, Ucenik y, Ucenik z){…}
```

Uslovljeni generički tipovi

- Rešenje 2: generički tip koji nasleđuje neki uporedivi generički tip:

```
static <T extends Comparable<T>> T maks3(T x, T y, T z) {
        T max = x;
    if (y.compareTo( max ) > 0 ){
        max = y;
    if(z.compareTo( max ) > 0 )
        max = z;
    return max;
}
```

Generička klasa

```
class KljucVrednost <K, V> {
        K kljuc;
        V vrednost;

        public KljucVrednost(K k, Vrednost){
                this.kljuc = k;
                this.vrednost = v;
        }

        @Override
        public String toString(){
                return “(“+kljuc.toString()+” ”+vrednost.toString()+”)”;
        }
}
```

Generička klasa

- Nakon toga možemo praviti različite ključ vrednosti klase, npr:

KljucVrednost<String, Integer> kv1=new KljucVrednost<String, Integer>(“test”,2);
KljucVrednost<Integer, Integer> kv1=new KljucVrednost<Integer, Integer>(4543,2);
KljucVrednost<Integer, String> kv1=new KljucVrednost<Integer, String>(2, “test”);

Negenerička lista

```
ArrayList lista =new ArrayList();
lista.add(“test”);
lista.add(4.5);
…
for(int i=0; i<lista.size(); i++){
        System.out.println(lista.get(i));
}
```

Generička lista

```
ArrayList<String> listaS = new ArrayList<String>();
ArrayList<Integer> listaL = new ArrayList<Integer>();

listaS.add(“test”); //ok
listaS.add(3); //greska!
listaL.add(4); //ok
listaL.add(5.6); //greska!
```

Sortiranje liste

ArrayList<String> reci = ne ArrayList<String>();

Collections.sort(reci); //sortira na podrazumevani nacin

Ako imamo **ArrayList<T>** lista, šta mora da ispunjava tip T da bi mogli da pozovemo sortiranje?

Primer sa sortiranjem liste

- Napisati klasu Ucenik koja ima polja broj, ime, prezime i prosečna ocena.
- Učitati u listu nekoliko učenika.
- Ispisati na konzoli listu učenika sortiranu prema prosečnoj oceni, pa po prezimenu, pa po imenu, pa po broju.

Zadatak 1

- Doraditi prethodni program na sledeći način:
- Najpre, pokrenuti beskonačnu petlju u kojoj treba detektovati sledeće naredbe:
  - Moguće je uneti novog učenika sa konzole unosom: novi Broj Ime Prezime Prosek
  - Moguće je obrisati učenika: obrisi Broj

Zadatak 2

- Napisati klasu Skup <T> za rad sa generickim skupovima podataka.
- Skup se predstavlja interno tako sto ima polje ArrayList<T> vrednosti;
- U klasi Skup <T> predvideti sledece metode:

1. void dodaj(T el)
2. void unija(Skup<T> s)
3. void presek(Skup<T> s)
4. void ispisi();

Zadatak 2 - nastavak

Primer izvrsavanja:

```
Skup<Integer> s1 =  new Skup<Integer>();
s1.dodaj(2);
Skup<Integer> s2 = new Skup<Integer>();
s2.dodaj(2);
s2.dodaj(3);
s1.unija(s2);
s1.ispisi();
//ispisuje 2 3
```