Dinamičke strukture podataka u Javi

AUDITORNE VJEŽBE

Sadržaj

Sučelje **Comparator**

Primjer implementacije sučelja **Comparator**

Primjer korištenja **TreeSet** implementacije

Primjer korištenja enumeracije

Primjer korištenja cjelobrojnih tokova

Primjer sortiranja liste po više kriterija

Pitanja s certifikata

Sučelje Comparator

- Osim podrazumijevanih algoritama unutar klase **Collections**, za definiranje specifičnih kriterija sortiranja moguće je koristiti sučelje **Comparator**
- Sučelje **Comparator** ima samo jednu apstraktnu metodu **compare** koja prima dva objekta i definira njihov odnos koji se koristi kod sortiranja: „veći”, „manji” i „jednak”
- Kako ima samo jednu apstraktnu metodu, sučelje **Comparator** je ujedno i **funkcionalno sučelje** pa se često koristi i u lambda izrazima
- Može se koristiti i kod poziva metoda za sortiranje iz klase Collections, kao i u konstruktorima zbirki koje to podržavaju

Primjer implementacije sučelja Comparator

```
public class Student {

   private String prezime;
   private String ime;
   private String jmbag;
   private double prosjek;

   //izostavljene "getter" metode i konstruktor

   @Override
   public String toString() {
      return prezime + " " + ime + "(" + prosjek + ")";
   }
}
```

Primjer implementacije sučelja Comparator

```
public class ProsjekSorter implements Comparator<Student> {

    @Override
    public int compare(Student st1, Student st2) {
        if(st1.getProsjek() > st2.getProsjek()) {
            return 1;
        }
        else if (st1.getProsjek() < st2.getProsjek()) {
            return -1;
        }
        else {
            return 0;
        }
    }
}
```

Primjer implementacije sučelja Comparator

```
Student prvi = new Student("Perić", "Pero", "0240293832", 4.11);
Student drugi = new Student("Ivić", "Ivo", "0240212322", 3.82);
Student treci = new Student("Markić", "Marko", "0240297890", 4.78);
Student cetvrti = new Student("Horvat", "Ivan", "0240294345", 3.05);

List<Student> listaStudenata = new ArrayList<>();
listaStudenata.add(prvi);
listaStudenata.add(drugi);
listaStudenata.add(treci);
listaStudenata.add(cetvrti);

Collections.sort(listaStudenata, new ProsjekSorter());

System.out.println(listaStudenata);
```

Ispis:

```
[Horvat Ivan(3.05), Ivić Ivo(3.82), Perić Pero(4.11), Markić Marko(4.78)]
```

Primjer korištenja TreeSet implementacije

```
Student prvi = new Student("Perić", "Pero", "0240293832", 4.11);
Student drugi = new Student("Ivić", "Ivo", "0240212322", 3.82);
Student treci = new Student("Markić", "Marko", "0240297890", 4.78);
Student cetvrti = new Student("Horvat", "Ivan", "0240294345", 3.05);

SortedSet<Student> setStudenata = new TreeSet<>(new ProsjekSorter());
setStudenata.add(prvi);
setStudenata.add(drugi);
setStudenata.add(treci);
setStudenata.add(cetvrti);

System.out.println("Najbolji: " + setStudenata.last());
System.out.println("Najgori: " + setStudenata.first());
```

Ispis:

```
Najbolji: Markić Marko(4.78)
Najgori: Horvat Ivan(3.05)
```

Primjer korištenja enumeracije

```
public enum Zupanija {

   ZUPANIJA_ZAGREBACKA ("1", "Zagrebačka županija", 309696, 3078d),
   ZUPANIJA_SISACKO_MOSLAVACKA ("044", "Sisačko-moslavačka", 185387, 4448d),
   ZUPANIJA_KRAPINSKO_ZAGORSKA ("049", "Krapinsko-zagorska", 142432, 1230d),
   ZUPANIJA_KARLOVACKA ("047", "Karlovačka", 141787, 3622d),
   ZUPANIJA_VARAZDINSKA ("042", "Varaždinska", 184769, 1260d),
   ZUPANIJA_KOPRIVNICKO_KRIZEVACKA ("048", "Koprivničko-križevačka", 124467, 1734d),
   ZUPANIJA_BJELOVARSKO_BILOGORSKA ("043", "Bjelovarsko-bilogorska", 133084, 2638),
   ZUPANIJA_PRIMORSKO_GORANSKA ("051", "Primorsko-goranska", 305505, 3590d),
   ZUPANIJA_LICKO_SENJSKA ("053", "Lično-senjska", 53677, 5350d),
   ZUPANIJA_VIROVITICKO_PODRAVSKA ("033", "Virovitičko-podravska", 93389, 2021d),
   ZUPANIJA_POZESKO_SLAVONSKA ("034", "Požeško-slavonska", 85831, 1821d),
   ZUPANIJA_BRODSKO_POSAVSKA ("035", "Brodsko-posavska", 176765, 2027d),
   ZUPANIJA_ZADARSKA ("023", "Zadarska", 162045, 3643d),
   ZUPANIJA_OSJECKO_BARANJSKA ("031", "Osječko-baranjska", 30506, 4149d),
   ZUPANIJA_SIBENSKO_KNINSKA ("022", "Šibensko-kninska", 112891, 2994d),
   ZUPANIJA_VUKOVARSKO_SRIJEMSKA ("032", "Vukovarsko-srijemska", 204768, 2448d),
   ZUPANIJA_SPLITSKO_DALMATINSKA ("021", "Splitsko-dalmatinska", 463676, 4524d),
   ZUPANIJA_ISTARSKA ("052", "Istarska", 206344, 2831d),
   ZUPANIJA_DUBROVACKO_NERETVANSKA ("020", "Dubrovačko-neretvanska", 122870, 1782d),
   ZUPANIJA_MEDJIMURSKA ("040", "Međimurska", 118426, 730),
   ZUPANIJA_GRAD_ZAGREB ("01", "Grad Zagreb", 779145, 3078d);
   //privatne varijable, konstruktor i "getteri"
}
```

Primjer korištenja enumeracije

```
public class GustocaNaseljenostiComparator implements Comparator<Zupanija> {
 @Override
 public int compare(Zupanija arg0, Zupanija arg1) {
   double gustocaNaseljenostiPrveZupanije = arg0.getBrojStanovnika() / arg0.getPovrsina();
   double gustocaNaseljenostiDrugeZupanije = arg1.getBrojStanovnika() / arg1.getPovrsina();

   if (gustocaNaseljenostiPrveZupanije < gustocaNaseljenostiDrugeZupanije) {
      return 1;
   }
   else if (gustocaNaseljenostiPrveZupanije > gustocaNaseljenostiDrugeZupanije) {
      return -1;
   }
   else {
      return 0;
   }
 }
}
```

Primjer sortiranja liste po više kriterija

```
Student prvi = new Student("Perić", "Pero", "0240293832", 4.11);
Student drugi = new Student("Ivić", "Ivo", "0240212322", 4.11);
Student treci = new Student("Markić", "Marko", "0240297890", 4.11);
Student cetvrti = new Student("Horvat", "Ivan", "0240294345", 4.11);
List<Student> listaStudenata = new ArrayList<Student>();
listaStudenata.add(prvi);listaStudenata.add(drugi);listaStudenata.add(treci);
listaStudenata.add(cetvrti);

Function<Student, Double> poProsjeku = Student::getProsjek;
Function<Student, String> poPrezimenu = Student::getPrezime;

Comparator<Student> poProsjekuIPrezimenu =
Comparator.comparing(poProsjeku).thenComparing(poPrezimenu);

listaStudenata.stream().sorted(poProsjekuIPrezimenu).forEach(System.out::println);
```

Ispis:

Horvat Ivan(4.11)
Ivić Ivo(4.11)
Markić Marko(4.11)
Perić Pero(4.11)

Primjer sortiranja liste po više kriterija

- U slučaju kad je potrebno sortirati zbirku po više kriterija, potrebno je koristiti sučelje „Function” koje predstavlja funkciju koja prima jedan argument (objekt klase „Student”), a vraća rezultat (prosjek ili JMBAG studenta), npr.:

```
Function<Student, Double> poProsjeku = Student::getProsjek;
```

- „Function” objekte je potrebno iskoristiti unutar **Comparator** metoda „comparing” i „thenComparing”
- Dobiveni „Comparator” objekt se može iskoristiti u metodi „sorted” kako bi se sortirali elementi po zadanim kriterijima i na kraju ispisali pomoću metode „forEach” i korištenjem funkcije „System.out::println”

Pitanja s certifikata (1)

What is the output of the following code segment?

```
ArrayList<String> sampleArrayList = new ArrayList<String>();
sampleArrayList.add("One");
sampleArrayList.add("Two");
sampleArrayList.add(1,"Three");
for(String s : sampleArrayList){
    System.out.print(s + " ");
}
```

A. `One Two Three`
B. `One Three Two`
C. `Three One Two`
D. `One Three`
E. `Three Two`
F. A compile time error will be generated.
G. A runtime exception will be thrown.

Pitanja s certifikata (2)

What best describes the result of the following code segment? The `ArrayList` `sampleArrayList` has already been declared and initialized.

```
int i = 63;
sampleArrayList.add(i);
```

A. The `int` is successfully placed into the `ArrayList`.
B. The `int` is converted to an `Integer` via autoboxing and then placed into the `ArrayList`.
C. `null` is placed into the `ArrayList`.
D. A compile time error will be generated.
E. A runtime exception will be thrown.

Pitanja s certifikata (3)

Given:

```
 2. import java.util.*;
 3. public class Vinegar {
 4.   public static void main(String[] args) {
 5.     Set<Integer> mySet = new HashSet<Integer>();
 6.     do1(mySet, "0");  do1(mySet, "a");
 7.     do2(mySet, "0");  do2(mySet, "a");
 8.   }
 9.   public static void do1(Set s, String st) {
10.     s.add(st);
11.     s.add(Integer.parseInt(st));
12.   }
13.   public static void do2(Set<Integer> s, String st) {
14.     s.add(st);
15.     s.add(Integer.parseInt(st));
16. } }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.
B. Compilation fails due to an error on line 6.
C. Compilation fails due to an error on line 13.
D. Compilation fails due to an error on line 14.
E. Compilation fails due to an error on line 15.
F. If only the line(s) of code that don't compile are removed, the code will run without exception.
G. If only the line(s) of code that don't compile are removed, the code will throw an exception.

Pitanja s certifikata (4)

Given:

```
 1. import java.util.*;
 2. public class Drunken {
 3.   public static void main(String[] args) {
 4.     Set<Stuff> s = new HashSet<Stuff>();
 5.     s.add(new Stuff(3));  s.add(new Stuff(4)); s.add(new Stuff(4));
 6.     s.add(new Stuff(5));  s.add(new Stuff(6));
 7.     s = null;
 8.     // do more stuff
 9.   }
10. }
11. class Stuff {
12.   int value;
13.   Stuff(int v) { value = v; }
14. }
```

When line 8 is reached, how many objects are eligible for garbage collection?

A. 4
B. 5
C. 6
D. 8
E. 10
F. 12

Pitanja s certifikata (5)

Given:

```
1. import java.util.*;
2. public class Piles {
3.   public static void main(String[] args) {
4.     TreeMap<String, String> tm = new TreeMap<String, String>();
5.     TreeSet<String> ts = new TreeSet<String>();
6.     String[] k = {"1", "b", "4", "3"};
7.     String[] v = {"a", "d", "3", "b"};
8.     for(int i=0; i<4; i++) {
9.       tm.put(k[i], v[i]);
10.      ts.add(v[i]);
11.    }
12.    System.out.print(tm.values() + " ");
13.    Iterator it2 = ts.iterator();
14.    while(it2.hasNext()) System.out.print(it2.next() + "-");
15. } }
```

Which of the following could be a part of the output? (Choose two.)

A. `[a, b, 3, d]`
B. `[d, a, b, 3]`
C. `[3, a, b, d]`
D. `[a, b, d, 3]`
E. `[1, 3, 4, b]`
F. `[b, 1, 3, 4]`
G. `3-a-b-d-`
H. `a-b-d-3-`
I. `a-d-3-b-`

Pitanja s certifikata (6)

Given:

```
1. import java.util.*;
2. class Fortress {
3.   private String name;
4.   private ArrayList<Integer> list;
5.   Fortress() { list = new ArrayList<Integer>(); }
6.
7.   String getName() { return name; }
8.   void addToList(int x) { list.add(x); }
9.   ArrayList getList() { return list; }
10. }
```

Which lines of code (if any) break encapsulation? (Choose all that apply.)

A. Line 3
B. Line 4
C. Line 5
D. Line 7
E. Line 8
F. Line 9
G. The class is already well encapsulated

Pitanja s certifikata (7)

Given:

```
public static void main(String[] args) {

  // INSERT DECLARATION HERE
  for (int i = 0; i <= 10; i++) {
    List<Integer> row = new ArrayList<Integer>();
    for (int j = 0; j <= 10; j++)
      row.add(i * j);
    table.add(row);
  }
  for (List<Integer> row : table)
    System.out.println(row);
  }
```

Which statements could be inserted at `// INSERT DECLARATION HERE` to allow this code to compile and run? (Choose all that apply.)

A. `List<List<Integer>> table = new List<List<Integer>>();`
B. `List<List<Integer>> table = new ArrayList<List<Integer>>();`
C. `List<List<Integer>> table = new ArrayList<ArrayList<Integer>>();`
D. `List<List, Integer> table = new List<List, Integer>();`
E. `List<List, Integer> table = new ArrayList<List, Integer>();`
F. `List<List, Integer> table = new ArrayList<ArrayList, Integer>();`
G. None of the above

Pitanja s certifikata (8)

Given:

```
public static void before() {
  Set set = new TreeSet();
  set.add("2");
  set.add(3);
  set.add("1");
  Iterator it = set.iterator();
    while (it.hasNext())
  System.out.print(it.next() + " ");
}
```

Which statements are true?

A. The `before()` method will print `1 2`
B. The `before()` method will print `1 2 3`
C. The `before()` method will print three numbers, but the order cannot be determined
D. The `before()` method will not compile
E. The `before()` method will throw an exception at runtime

Pitanja s certifikata (9)

Given:

```
import java.util.*;
class MapEQ {
  public static void main(String[] args) {
    Map<ToDos, String> m = new HashMap<ToDos, String>();
    ToDos t1 = new ToDos("Monday");
    ToDos t2 = new ToDos("Monday");
    ToDos t3 = new ToDos("Tuesday");
    m.put(t1, "doLaundry");
    m.put(t2, "payBills");
    m.put(t3, "cleanAttic");
    System.out.println(m.size());
  }
}
class ToDos{
  String day;
  ToDos(String d) { day = d; }
  public boolean equals(Object o) {
    return ((ToDos)o).day.equals(this.day);
  }
  // public int hashCode() { return 9; }
}
```

Which is correct? (Choose all that apply.)

A. As the code stands, it will not compile
B. As the code stands, the output will be 2
C. As the code stands, the output will be 3
D. If the `hashCode()` method is uncommented, the output will be 2
E. If the `hashCode()` method is uncommented, the output will be 3
F. If the `hashCode()` method is uncommented, the code will not compile

Pitanja s certifikata (10)

Which collection class(es) allows you to grow or shrink its size and provides indexed access to its elements, but whose methods are not synchronized? (Choose all that apply.)

A. `java.util.HashSet`
B. `java.util.LinkedHashSet`
C. `java.util.List`
D. `java.util.ArrayList`
E. `java.util.Vector`
F. `java.util.PriorityQueue`

Pitanja s certifikata (11)

Given:

```
3. import java.util.*;
4. class Business { }
5. class Hotel extends Business { }
6. class Inn extends Hotel { }
7. public class Travel {
8.   ArrayList<Hotel> go() {
9.     // insert code here
10.  }
11. }
```

Which statement inserted independently at line 9 will compile? (Choose all that apply.)

A. `return new ArrayList<Inn>();`

B. `return new ArrayList<Hotel>();`

C. `return new ArrayList<Object>();`

D. `return new ArrayList<Business>();`

Pitanja s certifikata (11)

Given:

```
3. import java.util.*;
4. class Dog { int size; Dog(int s) { size = s; } }
5. public class FirstGrade {
6.   public static void main(String[] args) {
7.     TreeSet<Integer> i = new TreeSet<Integer>();
8.     TreeSet<Dog> d = new TreeSet<Dog>();
9.
10.    d.add(new Dog(1));  d.add(new Dog(2));  d.add(new Dog(1));
11.    i.add(1);  i.add(2);  i.add(1);
12.    System.out.println(d.size() + " " + i.size());
13.  }
14. }
```

What is the result?

A. `1 2`
B. `2 2`
C. `2 3`
D. `3 2`
E. `3 3`
F. Compilation fails
G. An exception is thrown at runtime

Pitanja s certifikata (12)

Given:

```
3. import java.util.*;
4. public class GeoCache {
5.   public static void main(String[] args) {
6.     String[] s = {"map", "pen", "marble", "key"};
7.     Othello o = new Othello();
8.     Arrays.sort(s,o);
9.     for(String s2: s) System.out.print(s2 + " ");
10.    System.out.println(Arrays.binarySearch(s, "map"));
11.  }
12.  static class Othello implements Comparator<String> {
13.    public int compare(String a, String b) { return b.compareTo(a); }
14.  }
15. }
```

Which are true? (Choose all that apply.)

A. Compilation fails
B. The output will contain a `1`
C. The output will contain a `2`
D. The output will contain a `-1`
E. An exception is thrown at runtime
F. The output will contain `"key map marble pen"`
G. The output will contain `"pen marble map key"`

Pitanja?