СРАВНЯВАНЕ И СОРТИРАНЕ

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

23 септември 2008

СРАВНЯВАНЕ И СОРТИРАНЕ

Забележка: Тази лекция е адаптация на лекция от курса:

•6.092 Java Preparation for 6.170, Януари 2006
- Lucy Mendel
- Corey McCaffrey
- Rob Toscano
- Justin Mazolla Paluska
- Scott Osler
- Ray He

Интернет адрес:
http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-092January--IAP--2006/CourseHome/index.htm

СЪДЪРЖАНИЕ

СРАВНЯВАНЕ

- Използва се за да се прецени кой обект е по-голям или двата обекта са равни
- Изразът **`(a.compareTo(b))`** се очаква да върне:
  - <0, ако a<b
  - =0, ако a=b
  - >0, ако a>b

ПРИМЕР
Сравнения

```
Integer one = 1;
System.out.println(one.compareTo(3));          // => -1
System.out.println(one.compareTo(-50));        // => 1

String frank = "Frank";
System.out.println(frank.compareTo("Booth"));  // => 4
System.out.println(frank.compareTo("Hopper")); // => -2
```

ПРИМЕР
Азбучно сортиране на списък

```
        List<String> names = new ArrayList<String>();
        // добавяне на елементи към списъка
        names.add("Sailor");
        names.add("Lula");
        names.add("Bobby");
        names.add("Santos");
        names.add("Dell");

        // сортиране на списъка
        Collections.sort(names);

    // names => [ "Bobby", "Dell", "Lula", "Sailor",
"Santos" ]
```

ИНТЕРФЕЙСЪТ ***Comparable***

- За да бъдат сортирани елементите на списъка те трябва да имплементират интерфейса **`Comparable`**
- Методът, който трябва да бъде реализиран е:
- **`int compareTo(Object obj);`** (разгледан по-горе)
- В примера – **`String`** имплементира **`Comparable`** и за това може да бъде сортиран списък от низове

ПРИМЕР
Класът Name, със сортиране

```
public  class Name implements Comparable<Name> {

   // ...

   // сортиране по фамилия
   public int compareTo(Name o) {
      int compare = this.last.compareTo(o.last);
      if (compare != 0)
         return compare;
      else
         return this.first.compareTo(o.first);
   }
}
```

ПРИМЕР
Сортиране на списък с Name

```
        List<Name> names = new ArrayList<Name>();
        names.add(new Name("Nicolas", "Cage"));
        names.add(new Name("Laura", "Dern"));
        names.add(new Name("Harry", "Stanton"));
        names.add(new Name("Diane", "Ladd"));
        names.add(new Name("William", "Morgan"));
        names.add(new Name("Dirty", "Glover"));
        names.add(new Name("Johnny", "Cage"));
        names.add(new Name("Metal", "Cage"));

        System.out.println(names);
        Collections.sort(names);
        System.out.println(names);

// => [Johnny Cage, Metal Cage, Nicolas Cage, Laura Dern,
//  Crispin Glover,Diane Ladd, William Morgan, Harry Stanton]
```

ИНТЕРФЕЙСЪТ ***Comparator***

- Сортиране по различни критерии
- Сравнява два обекта
- Методът, който трябва да се реализира:
  - ***int compare(Object o1, Object o2);***

ПРИМЕР
Сортиране по първо име

```
import java.util.Comparator;

//записано в отделен файл: FirstNameFirst.java
public class FirstNameFirst implements Comparator<Name> {
    //сортиране по първо име
    public int compare(Name n1, Name n2) {
        int ret = n1.first.compareTo(n2.first);
        if (ret != 0)
            return ret;
        else
            return n1.last.compareTo(n2.last);
    }
}
```

ПРИМЕР
Сортиране по първо име, тест

```
        List<Name> names = new ArrayList<Name>();
        // ...

        // създаваме обект, чрез който ще
        // извършваме сравненията по време на сортиране
        Comparator<Name> first = new FirstNameFirst();
        Collections.sort(names, first);

        System.out.println(names);

// => [Crispin Glover, Diane Ladd, Harry Stanton, JohnnyCage,
// Laura Dern, Metal Cage, Nicolas Cage, WilliamMorgan]
```

ИЗИСКВАНИЯ ПРИ СРАВНЯВАНЕТО

- Резултатът при сравнението на два обекта винаги трябва да е един и същи (щом обектите не се променят)!
- Особено внимание трябва да се обърне на случаите, когато (compare(e1, e2)==0) != e1.equals(e2) (трябва да се избягват!)
- Такива случаи биха довели до неопределени резултати при сортирането на колекции като SortedSet, SortedMap и т.н.

ПОСТОЯННО СОРТИРАНИ МНОЖЕСТВА

- Използва се TreeSet
  - Или обектите в TreeSet трябва да имплементират Comparable
  - Или да се подаде Comparator при създаването на TreeSet

```
SortedSet<Name> names = new TreeSet<Name>(new
                                          FirstNameFirst());
names.add(new Name("Laura", "Dern"));
names.add(new Name("Harry", "Stanton"));
names.add(new Name("Diane", "Ladd"));

System.out.println(names);

// => [Diane Ladd, Harry Stanton, Laura Dern]
```