JAC444 - BTP400 Course Object-Oriented Software Development II - Java

Collections

Segment 5

Objects Ordering

Comparable Types

- Elements that can be compared to one another are called *mutually comparable*.

- To compare to objects, the class must implement the `Comparable` interface

```
public interface Comparable<T> {
    public int compareTo(T o);
}
```

`compareTo(T o)`

returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object

compareTo Example

The compareTo method compares the receiving object with the specified object

It returns a negative integer, 0, or a positive integer depending on whether the receiving object is less than, equal to, or greater than the specified object

```
public class Student implements Comparable<Student> {
  private String  first, last;
  //..other fields
  //equals(), hashCode(), toString() implementations

  public int compareTo(Student s) {
    int lastRes = last.compareTo(s.last);
    return (lastRes!=0 ? lastRes : first.compareTo(s.first));
  }
}
```

last.compareTo() invokes the compareTo method of class String

Comparator Interface

- The `Comparator` interface defines a comparison function, which imposes a total ordering on some collection of objects.

```
public interface Comparator<T> {
    int compare(T o1, T o2);
}
```

```
static final Comparator<Student> STUDENT_ORDER =
                       new Comparator<Student>() {
  public int compare(Student s1, Student s2) {
    return s2.getGrade().compareTo(s1.getGrade());
  }
};
```

The SortedSet Interface

`SortedSet` is a `Set` that maintains its elements in ascending order

```
public interface SortedSet<E> extends Set<E> {
    // Range-view
    SortedSet<E> subSet(E fromElement, E toElement);
    SortedSet<E> headSet(E toElement);
    SortedSet<E> tailSet(E fromElement);

    // Endpoints
    E first();
    E last();

    // Comparator access
    Comparator<? super E> comparator();
}
```

The SortedMap Interface

`SortedMap` is a `Map` that maintains its entries in ascending order, sorted according to natural ordering of its keys

```
public interface SortedMap<K, V> extends Map<K, V>{

    Comparator<? super K> comparator();
    SortedMap<K, V> subMap(K fromKey, K toKey);
    SortedMap<K, V> headMap(K toKey);
    SortedMap<K, V> tailMap(K fromKey);
    K firstKey();
    K lastKey();
}
```

Algorithms

```
import java.util.*;

public class SortWords {

  public static void main(String[] args) {
     List l = Arrays.asList(args);

     Collections.sort(l);
     System.out.println(l);
  }
}
```