Comparable Interface

Comparable interface is in java.lang package. It is used to impose a total ordering on objects of class that implements this interface. The objects are ordered in natural order and natural comparison method is compareTo(T o).

Natural ordering means natural order for numbers, alphabetical order for String, chronological order for dates.

Invocation

int x=thisObject.compareTo(anotherObject);

the compareTO method returns int with following:

|  |  |
| --- | --- |
| **Returns** | **Condition** |
| Negative | thisObject < anotherObject |
| Zero | thisObject == anotherObject |
| Positive | thisObject > anotherObject |
|  |  |

The Comparable interface is used by Arrays.sort() and Collections.sort() to sort array of objects and lists respectively.

Let us implement Comparable Interface for Person class and sort the List of Person by their name.

**package** org.collections;

**public** **class** Person **implements** Comparable<Person>{

**private** **int** age;

**private** String name;

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (obj **instanceof** Person) {

Person o = (Person) obj;

**if** (o.getAge() == **this**.getAge() && o.getName().equals(**this**.getName())) {

**return** **true**;

}

**return** **false**;

}

**return** **false**;

}

@Override

**public** **int** hashCode() {

StringBuilder sb = **new** StringBuilder();

sb.append(**this**.getAge()).append(**this**.getName());

**return** sb.hashCode();

}

/\*\*

\* compareTo(T o) is used as natural comparison method where

\* comparison is done on lexical order on Person's name.

\* \*/

@Override

**public** **int** compareTo(Person person) {

**return** **this**.getName().compareTo(person.getName());

}

@Override

**public** String toString() {

**return** **new** StringBuilder()

.append("{")

.append(**this**.getName())

.append(" ")

.append(**this**.getAge())

.append("}")

.toString();

}

}

Driver Class

**package** org.collections;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** ComparableDemo {

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

List<Person> list=**new** ArrayList<Person>();

list.add(**new** Person("Eddard",55));

list.add(**new** Person("Rob",23));

list.add(**new** Person("Joffery",21));

list.add(**new** Person("Sansa",19));

list.add(**new** Person("Rickon",7));

list.add(**new** Person("Brandon",9));

System.***out***.println("Before Sort "+list);

Collections.*sort*(list);

System.***out***.println("After Sort "+list);

}

}

Output

Before Sort [{Eddard 55}, {Rob 23}, {Joffery 21}, {Sansa 19}, {Rickon 7}, {Brandon 9}]

After Sort [{Brandon 9}, {Eddard 55}, {Joffery 21}, {Rickon 7}, {Rob 23}, {Sansa 19}]

But let us say that needs arise and now we need to sort by age then? Well the Comparable has limitation that only one sort sequence can be created.

So if we need multiple sort sequences like on name, age, profession etc then we use Comparator<T> interface.

In next post we will see what Comparator is and how to use it.