Collections class sort(List, Comparator) method

In [previous](http://data-structure-learning.blogspot.com/2015/11/collections-class-sort-method.html) post we saw sort method that accepts List as argument. We sorted the List<String> where String class implements the Comparable<T> interface. We also saw the example of Person class and implemented the Comparable<T> interface and sorted the List.

In this post we will see how to implement the Comparator interface. You can read about [Comparator](http://data-structure-learning.blogspot.com/2015/06/comparator-interface.html) interface [here](http://data-structure-learning.blogspot.com/2015/06/comparator-interface.html).

Let us take a Person class which will have two attributes namely age and name. We will write Comparator for age and name and sort the list.

Below is the Person class.

**public** **class** People {

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

**private** String name;

**public** People(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** String toString() {

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

.append("{")

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

.append(" ")

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

.append("}")

.toString();

}

}

Below is the class the uses the Comparator<T> to sort List<People> by age and name.

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.List;

**public** **class** ComparatorDemo {

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

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

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

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

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

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

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

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

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

*sortByName*(list);

*sortByAge*(list);

}

/\*\*

\* Below method sorts the List<People> using the

\* Comparator as Anonymous Inner Class.

\* \*/

**public** **static** **void** sortByName(List<People> list) {

**Collections.*sort*(list,** **new Comparator<People>() {**

**public int compare(People p1, People p2) {**

**return p1.getName().compareTo(p2.getName());**

**}**

**});**

System.***out***.println("Sorted by Name: " + list);

}

/\*\*

\* Below method sorts the List<People> using the

\* Comparator as Anonymous Inner Class.

\* \*/

**public** **static** **void** sortByAge(List<People> list) {

**Collections.*sort*(list,** **new Comparator<People>() {**

**public int compare(People p1, People p2) {**

**if (p1.getAge() == p2.getAge()) {**

**return 0;**

**} else if (p1.getAge() < p2.getAge()) {**

**return -1;**

**} else {**

**return 1;**

**}**

**}**

**});**

System.***out***.println("Sorted by Age: " + list);

}

}

Output

Before Sort [{Eddard 55},

{Rob 23},

{Joffery 21},

{Sansa 19},

{Rickon 7},

{Brandon 9}]

Sorted by Name: [{Brandon 9},

{Eddard 55},

{Joffery 21},

{Rickon 7},

{Rob 23},

{Sansa 19}]

Sorted by Age: [{Rickon 7},

{Brandon 9},

{Sansa 19},

{Joffery 21},

{Rob 23},

{Eddard 55}]

That’s all on Collections.sort(List, Comparator) method. In next post we will see how to use Comparator interface to sort the List which contains null elements.