Sorting in Collection

We can sort the elements of:

1. String objects
2. Wrapper class objects
3. User-defined class objects

|  |
| --- |
| **Collections** class provides static methods for sorting the elements of a collection.  If collection elements are of a Set type, we can use TreeSet. However, we cannot sort the  elements of List. Collections class provides methods for sorting the elements of List type elements. |

Method of Collections class for sorting List elements

**public void sort(List list):** is used to sort the elements of List. List elements must be of the Comparable type.

Note: String class and Wrapper classes implement the Comparable interface. So if you store the objects of string or wrapper classes, it will be Comparable.

Example to sort string objects

1. **import** java.util.\*;
2. **class** TestSort1{
3. **public** **static** **void** main(String args[]){
5. ArrayList<String> al=**new** ArrayList<String>();
6. al.add("Viru");
7. al.add("Saurav");
8. al.add("Mukesh");
9. al.add("Tahir");
11. Collections.sort(al);
12. Iterator itr=al.iterator();
13. **while**(itr.hasNext()){
14. System.out.println(itr.next());
15. }
16. }
17. }

**[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestSort1" \t "_blank)**

Mukesh

Saurav

Tahir

Viru

Example to sort string objects in reverse order

1. **import** java.util.\*;
2. **class** TestSort2{
3. **public** **static** **void** main(String args[]){
5. ArrayList<String> al=**new** ArrayList<String>();
6. al.add("Viru");
7. al.add("Saurav");
8. al.add("Mukesh");
9. al.add("Tahir");
11. Collections.sort(al,Collections.reverseOrder());
12. Iterator i=al.iterator();
13. **while**(i.hasNext())
14. {
15. System.out.println(i.next());
16. }
17. }
18. }

Viru

Tahir

Saurav

Mukesh

Example to sort Wrapper class objects

1. **import** java.util.\*;
2. **class** TestSort3{
3. **public** **static** **void** main(String args[]){
5. ArrayList al=**new** ArrayList();
6. al.add(Integer.valueOf(201));
7. al.add(Integer.valueOf(101));
8. al.add(230);//internally will be converted into objects as Integer.valueOf(230)
10. Collections.sort(al);
12. Iterator itr=al.iterator();
13. **while**(itr.hasNext()){
14. System.out.println(itr.next());
15. }
16. }
17. }

101

201

230

Example to sort user-defined class objects

1. **import** java.util.\*;
3. **class** Student **implements** Comparable<Student> {
4. **public** String name;
5. **public** Student(String name) {
6. **this**.name = name;
7. }
8. **public** **int** compareTo(Student person) {
9. **return** name.compareTo(person.name);
11. }
12. }
13. **public** **class** TestSort4 {
14. **public** **static** **void** main(String[] args) {
15. ArrayList<Student> al=**new** ArrayList<Student>();
16. al.add(**new** Student("Viru"));
17. al.add(**new** Student("Saurav"));
18. al.add(**new** Student("Mukesh"));
19. al.add(**new** Student("Tahir"));
21. Collections.sort(al);
22. **for** (Student s : al) {
23. System.out.println(s.name);
24. }
25. }
26. }

Mukesh

Saurav

Tahir

Viru