**Java TreeSet**

**Introduction**

The TreeSet is one of two sorted collections (the other being TreeMap).TreeSet extends AbstractSet and implements the NavigableSet interface. It creates a collection that uses a tree for storage. Objects are stored in sorted, ascending order according to natural order. Optionally, you can construct a TreeSet with a constructor that lets you give the collection your own rules for what the order should be (rather than relying on the ordering defined by the elements' class) by using a Comparable or Comparator.

Access and retrieval times are quite fast, which makes TreeSet an excellent choice when storing large amounts of sorted information that must be found quickly. TreeSet might not be used when our application has requirement of modification of set in terms of frequent addition of elements.

**Constructors**

TreeSet( ); // Default Constructor

TreeSet(Collection<? extends E> c); //TreeSet from Collection C

TreeSet(Comparator<? super E> comp); // TreeSet with custom ordering as per Comparator

TreeSet(SortedSet<E>ss); //TreeSet that contains the elements of ss.

**Important Methods of TreeSet Class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void add(Object o) | Adds the specified element to this set if it is not already present. |
| void clear() | Removes all of the elements from this set. |
| Object first() | Returns the first (lowest) element currently in this sorted set. |
| Object last() | Returns the last (highest) element currently in this sorted set. |
| boolean isEmpty() | Returns true if this set contains no elements. |
| boolean remove(Object o) | Removes the specified element from this set if it is present. |
| SortedSetsubSet(Object fromElement, Object toElement) | Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive. |

**Java Code**

[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/java-treeset.php)

1. **package** treeset;


5. **import** java.util.TreeSet;

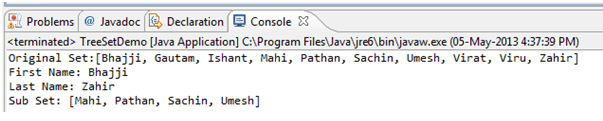

9. **public** **class** TreeSetDemo {


13. **public** **static** **void** main(String[] args) {
15. TreeSet<string> playerSet = **new** TreeSet<string>();


19. playerSet.add("Sachin");
21. playerSet.add("Zahir");
23. playerSet.add("Mahi");
25. playerSet.add("Bhajji");
27. playerSet.add("Viru");
29. playerSet.add("Gautam");
31. playerSet.add("Ishant");
33. playerSet.add("Umesh");
35. playerSet.add("Pathan");
37. playerSet.add("Virat");
39. playerSet.add("Sachin"); // This is duplicate element so will not be added again
41. //below will print list in alphabetic order
43. System.out.println("Original Set:" + playerSet);
45. System.out.println("First Name: "+ playerSet.first());
47. System.out.println("Last Name: "+ playerSet.last());


51. TreeSet<string> newPlySet = (TreeSet<string>) playerSet.subSet("Mahi", "Virat");
53. System.out.println("Sub Set: "+ newPlySet);
55. }
57. }
59. </string></string></string></string>

**Output**



In above example we are creating TreeSet of String object. String class is having comparable interface implemented by Java library. Let’s think of case when we need to have our own objects to be stored in Set and ordering of objects as per our rule. Below example shows Cricketer as object having two properties name and battingPosition. We want to store all Cricketer object as per batting position defined means when we iterate through collection we get names as per batting positions.

**Java Code ( Cricketer.java)**

[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/java-treeset.php)

1. **package** treeset;


5. **public** **class** Cricketer {


9. **private** String name;
11. **private** **int** battingPosition;


15. Cricketer(String cricketerName, **int** cBattingPosition){
17. **this**.name = cricketerName;
19. **this**.battingPosition = cBattingPosition;
21. }


25. **public** String getName() {
27. **return** name;
29. }


33. **public** **int** getBattingPosition() {
35. **return** battingPosition;
37. }


41. }

**CompareCricketer.java:** Here we are defining rules how to organize Cricketer objects in TreeMap.

**Java Code (CompareCricketer.java)**

[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/java-treeset.php)

1. **package** treeset;


5. **import** java.util.Comparator;


9. **public** **class** CompareCricketer **implements** Comparator <Cricketer> {
11. @Override
13. **public** **int** compare(Cricketer arg0, Cricketer arg1) {
15. **if**(arg0.getBattingPosition() > arg1.getBattingPosition())
17. **return** 1;
19. **else** **if** (arg0.getBattingPosition() < arg1.getBattingPosition())
21. **return** -1;
23. **else** **return** 0;


27. }


31. }

**Java Code**

[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/java-treeset.php)

1. **package** treeset;


5. **import** java.util.Iterator;
7. **import** java.util.TreeSet;

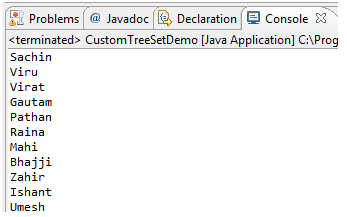

11. **public** **class** CustomTreeSetDemo {


15. **public** **static** **void** main(String[] args) {
17. TreeSet<cricketer> playerSet = **new** TreeSet<cricketer>(
19. **new** CompareCricketer());


23. playerSet.add(**new** Cricketer("Sachin", 1));
25. playerSet.add(**new** Cricketer("Zahir", 9));
27. playerSet.add(**new** Cricketer("Mahi", 7));
29. playerSet.add(**new** Cricketer("Bhajji", 8));
31. playerSet.add(**new** Cricketer("Viru", 2));
33. playerSet.add(**new** Cricketer("Gautam", 4));
35. playerSet.add(**new** Cricketer("Ishant", 10));
37. playerSet.add(**new** Cricketer("Umesh", 11));
39. playerSet.add(**new** Cricketer("Pathan", 5));
41. playerSet.add(**new** Cricketer("Virat", 3));
43. playerSet.add(**new** Cricketer("Raina", 6));


47. Iterator<cricketer> it = playerSet.iterator();
49. **while** (it.hasNext()) {
51. System.out.println(it.next().getName());
53. }
55. }
57. }
59. </cricketer></cricketer></cricketer>

**Output**



**Summary**

* TreeSet is sorted collection class
* Sorting of objects in TreeSet in natural order by default or we can provide comparator class reference for customized sorting.
* While working with large amount of data access and retrieval is faster with TreeSet