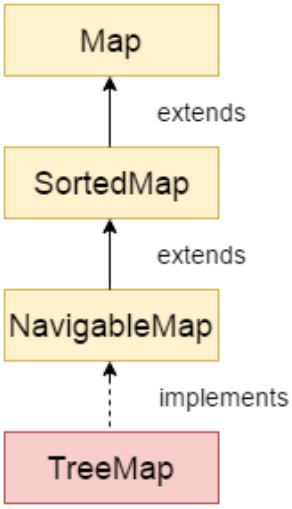
Java TreeMap class

Java TreeMap class implements the Map interface by using a tree. It provides an efficient means of storing key/value pairs in sorted order.



The important points about Java TreeMap class are:

- A TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
- It contains only unique elements.
- It cannot have null key but can have multiple null values.
- It is same as HashMap instead maintains ascending order.

TreeMap class declaration

Let's see the declaration for java.util.TreeMap class.

```
public class TreeMap<K,V> extends AbstractMap<K,V> implements NavigableMap<K,V>, Cloneable, Serializable
```

TreeMap class Parameters

Let's see the Parameters for java.util.TreeMap class.

- **K**: It is the type of keys maintained by this map.
- **V**: It is the type of mapped values.

Constructors of Java TreeMap class

Constructor	Description
TreeMap()	It is used to construct an empty tree map that will be sorted using the natural order of its key.
TreeMap(Comparator comp)	It is used to construct an empty tree-based map that will be sorted using the comparator comp.
TreeMap(Map m)	It is used to initialize a tree map with the entries from **m**, which will be sorted using the natural order of the keys.
TreeMap(SortedMap sm)	It is used to initialize a tree map with the entries from the SortedMap **sm**, which will be sorted in the same order as **sm.**

Methods of Java TreeMap class

Method	Description
boolean containsKey(Object key)	It is used to return true if this map contains a mapping for the specified key.
boolean containsValue(Object value)	It is used to return true if this map maps one or more keys to the specified value.
Object firstKey()	It is used to return the first (lowest) key currently in this sorted map.
Object get(Object key)	It is used to return the value to which this map maps the specified key.
Object lastKey()	It is used to return the last (highest) key currently in this sorted map.
Object remove(Object key)	It is used to remove the mapping for this key from this TreeMap if present.
void putAll(Map map)	It is used to copy all of the mappings from the specified map to this map.
Set entrySet()	It is used to return a set view of the mappings contained in this map.
int size()	It is used to return the number of key-value mappings in this map.
Collection values()	It is used to return a collection view of the values contained in this map.

Java TreeMap Example:

```
import java.util.*;
class TestCollection15{
 public static void main(String args[]){
  TreeMap<Integer,String> hm=new TreeMap<Integer,String>();
  hm.put(100,"Amit");
  hm.put(102,"Ravi");
  hm.put(101,"Vijay");
  hm.put(103,"Rahul");
  for(Map.Entry m:hm.entrySet()){
   System.out.println(m.getKey()+" "+m.getValue());
  }
```

```
 }
}
```

Test it Now

```
Output:100 Amit
       101 Vijay
       102 Ravi
       103 Rahul
```

Java TreeMap Example: remove()

```
import java.util.*;
public class TreeMapExample {
   public static void main(String args[]) {
   // Create and populate tree map
   Map<Integer, String> map = new TreeMap<Integer, String>();
   map.put(102,"Let us C");
   map.put(103, "Operating System");
   map.put(101, "Data Communication and Networking");
   System.out.println("Values before remove: "+ map);
   // Remove value for key 102
   map.remove(102);
   System.out.println("Values after remove: "+ map);
   }
}
```

Output:

```
Values before remove: {101=Data Communication and Networking, 102=Let us C, 103=Operating System}
Values after remove: {101=Data Communication and Networking, 103=Operating System}
```

What is difference between HashMap and TreeMap?

HashMap	TreeMap
1) HashMap can contain one null key.	TreeMap can not contain any null key.
2) HashMap maintains no order.	TreeMap maintains ascending order.

Java TreeMap Example: Book

```
import java.util.*;
```

```
class Book {
int id;
String name,author,publisher;
int quantity;
public Book(int id, String name, String author, String publisher, int quantity) {
    this.id = id;
    this.name = name;
    this.author = author;
    this.publisher = publisher;
    this.quantity = quantity;
}
}
public class MapExample {
public static void main(String[] args) {
    //Creating map of Books
    Map<Integer,Book> map=new TreeMap<Integer,Book>();
    //Creating Books
    Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
    Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);
    Book b3=new Book(103,"Operating System","Galvin","Wiley",6);
    //Adding Books to map
    map.put(2,b2);
    map.put(1,b1);
    map.put(3,b3);

    //Traversing map
    for(Map.Entry<Integer, Book> entry:map.entrySet()){
        int key=entry.getKey();
        Book b=entry.getValue();
        System.out.println(key+" Details:");
        System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
    }
}
}
```

Output:

```
1 Details:
101 Let us C Yashwant Kanetkar BPB 8
2 Details:
102 Data Communications & Networking Forouzan Mc Graw Hill 4
```

```
3 Details:
103 Operating System Galvin Wiley 6
```

← prev next →

Share this page

Latest 4 Tutorials

 CouchDB
 Docker
 Rails
 RichFaces