```
{
    void meth1()
    {
        System.out.println("Implementing Set Interface\n");

        HashSet<Object> set=new HashSet<Object>();

        set.add(10); // Insertion order is NOT maintained
        set.add("Java");// Heterogeneous data is allowed
        set.add(null);// null value is allowed
        set.add(10);// Duplicates are NOT allowed
        set.add('A');// It is avilable from Java 1.2v
        set.add(true);// Its default capacity is 16 [LOAD FACTOR : 0.75]
        set.add(1);// Its size increases by DOUBLE
        set.add(77); // It is NOT synchronized

        System.out.println(set);

        
    }
```

The default capacity of haset is 16 and the size increased by double after the 12 elements not after the 16 elements

There is no index position for sets the data will be stored basing on hash vales.

So, for is not used to retrieve the data from sets.

get() is not used because of no index position.

We can pass any collection class as a parameter for another collection class.

Understanding Set Interface

- It is the child interface of Collection.
- A Set is a Collection that cannot contain duplicate elements
- The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

HashSet:

- HashSet is available since jdk1.2V
- Underlying data structure for HashSet is HashTable.
- It doesn't allows duplicates & insertion order is not maintained.
- Default capacity when creating an HashSet is 16.
- Load Factor for HashSet is 0.75 **(No of elements/ Size of the hashTable)**
- HashSet is not synchronized by default.
- Accepts 'null' value for only once.

```
HashSet hs=new HashSet();
HashSet hs=new HashSet(int initialcapacity);
```

LinkedHashSet:

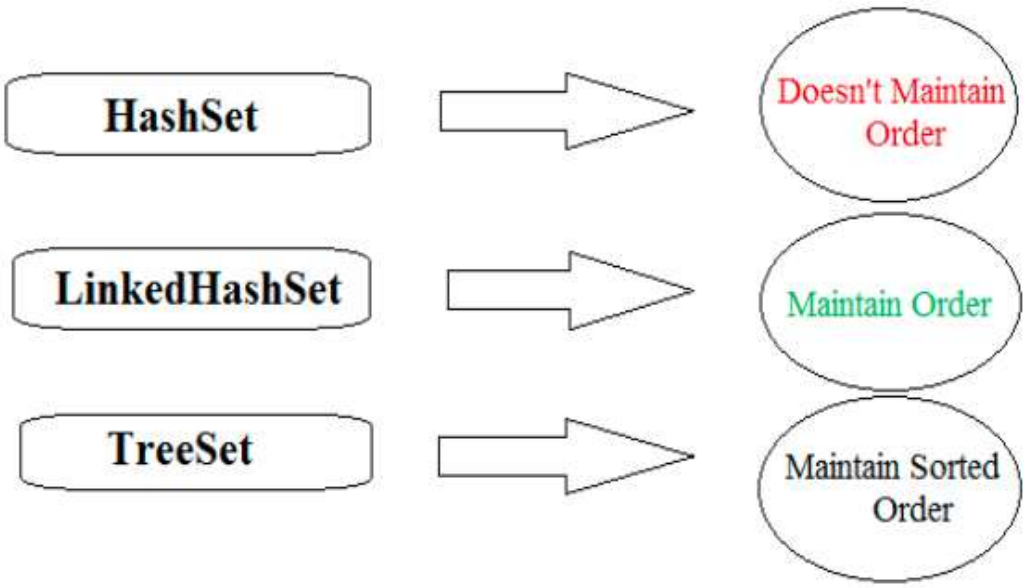
- The only difference between HashSet & LinkedHashSet is Hash set doesn't maintain the insertion order where as LinkedHashSet maintains it.
- LinkedHashSet is available since jdk1.4V.
- It inherits HashSet class and implements Set interface.

```
LinkedHashSet lhs=new LinkedHashSet();
```

TreeSet:

- TreeSet maintains sorting order of inserted elements.
- TreeSet is available since jdk1.2V.
- It will arrange the elements in ascending order using balanced binary search tree algorithm.
- TreeSet will not allow to insert Heterogeneous objects
- It doesn't allows duplicates & insertion order is not maintained.

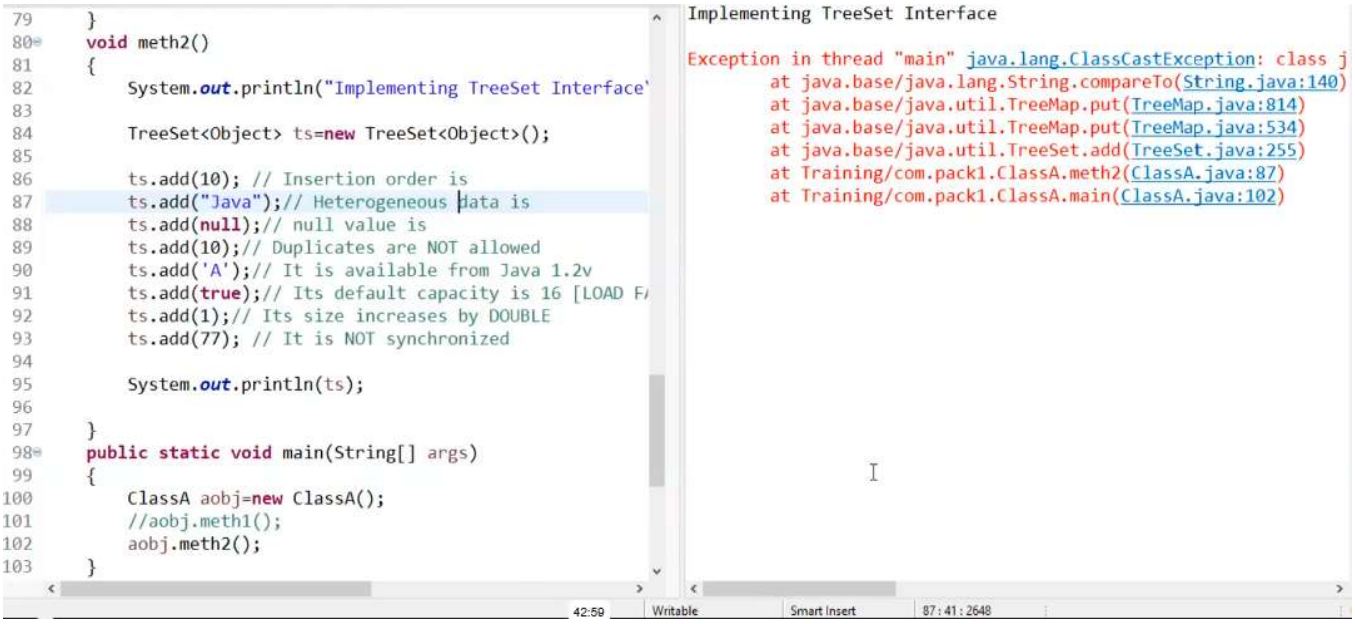
```
TreeSet t=new TreeSet();
```



All the three class doesn't accept duplicates elements.

TreeSet methods

descendingSet();	Returns a reverse order view of the elements contained in this set
descendingIterator();	Returns an iterator over the elements in this set in descending order.
headSet(E toElement);	Returns the elements less than to the specified element
tailSet(E toElement);	Returns the elements greater than or equal to the specified element

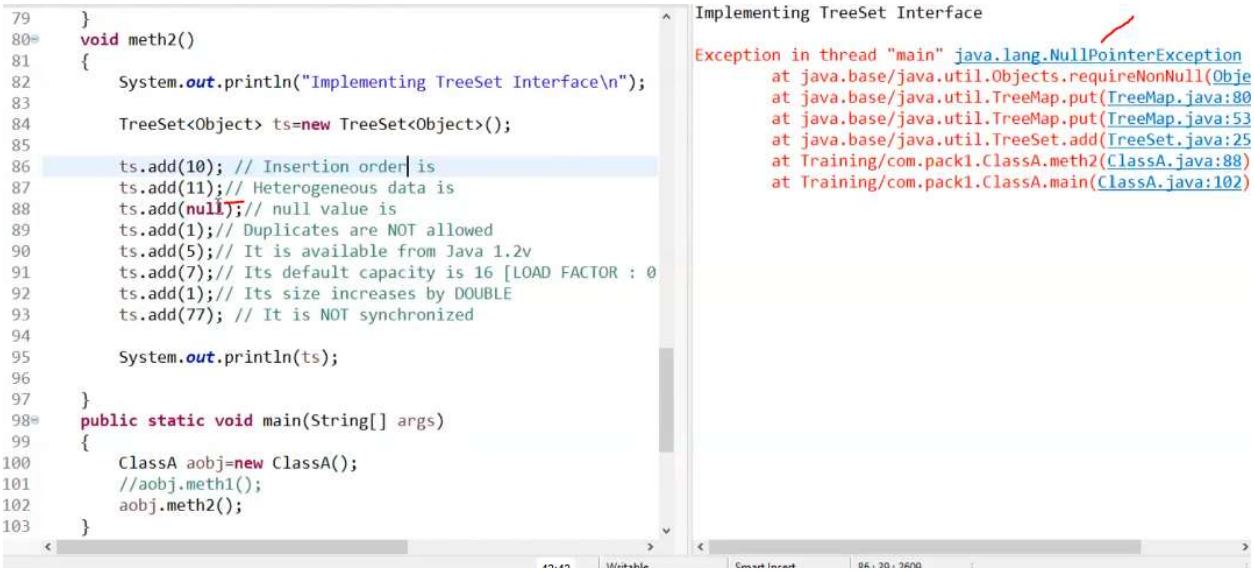


```
    }
    void meth2()
    {
        System.out.println("Implementing TreeSet Interface");

        TreeSet<Object> ts=new TreeSet<Object>();

        ts.add(10); // Insertion order is
        ts.add("Java");// Heterogeneous data is
        ts.add(null);// null value is
        ts.add(10);// Duplicates are NOT allowed
        ts.add('A');// It is available from Java 1.2v
        ts.add(true);// Its default capacity is 16 [LOAD FA
        ts.add(1);// Its size increases by DOUBLE
        ts.add(77); // It is NOT synchronized

        System.out.println(ts);

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();
        aobj.meth2();
    }
```

```
Implementing TreeSet Interface

Exception in thread "main" java.lang.ClassCastException: class j
        at java.base/java.lang.String.compareTo(String.java:140)
        at java.base/java.util.TreeMap.put(TreeMap.java:814)
        at java.base/java.util.TreeMap.put(TreeMap.java:534)
        at java.base/java.util.TreeSet.add(TreeSet.java:255)
        at Training/com.pack1.ClassA.meth2(ClassA.java:87)
        at Training/com.pack1.ClassA.main(ClassA.java:102)
```

Treeset does not allow heterogenous data

It allows only homogenous data.



```
    }
    void meth2()
    {
        System.out.println("Implementing TreeSet Interface\n");

        TreeSet<Object> ts=new TreeSet<Object>();

        ts.add(10); // Insertion order is
        ts.add(11);// Heterogeneous data is
        ts.add(null);// null value is
        ts.add(1);// Duplicates are NOT allowed
        ts.add(5);// It is available from Java 1.2v
        ts.add(7);// Its default capacity is 16 [LOAD FACTOR : 0
        ts.add(1);// Its size increases by DOUBLE
        ts.add(77); // It is NOT synchronized

        System.out.println(ts);

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();
        aobj.meth2();
    }
```

```
Implementing TreeSet Interface

Exception in thread "main" java.lang.NullPointerException
        at java.base/java.util.Objects.requireNonNull(Obje
        at java.base/java.util.TreeMap.put(TreeMap.java:80
        at java.base/java.util.TreeMap.put(TreeMap.java:53
        at java.base/java.util.TreeSet.add(TreeSet.java:25
        at Training/com.pack1.ClassA.meth2(ClassA.java:88)
        at Training/com.pack1.ClassA.main(ClassA.java:102)
```

Treeset does not allow null value

```
79      }
80      void meth2()
81      {
82          System.out.println("Implementing TreeSet Interface\n");
83
84          TreeSet<Object> ts=new TreeSet<Object>();
85
86          ts.add(10); // Insertion order is NOT maintained, but so
87          ts.add(11);// Heterogeneous data is NOT allowed
88          //ts.add(null);// null value is NOT Allowed
89          ts.add(1);// Duplicates are NOT allowed
90          ts.add(5);// It is available from Java 1.2v
91          ts.add(7);// Its default capacity is 16 [LOAD FACTOR : 0
92          ts.add(1);// Its size increases by DOUBLE
93          ts.add(77); // It is NOT synchronized
94
95          System.out.println(ts);
96
97      }
98      public static void main(String[] args)
99      {
100         ClassA aobj=new ClassA();
101         //aobj.meth1();
102         aobj.meth2();
103     }
```

```
Implementing TreeSet Interface

[1, 5, 7, 10, 11, 77]
```

```
88          //ts.add(null);// null value is NOT Allowed
89          ts.add(1);// Duplicates are NOT allowed
90          ts.add(5);// It is available from Java 1.2v
91          ts.add(7);// Its default capacity is 16 [LOAD FACTOR : 0
92          ts.add(1);// Its size increases by DOUBLE
93          ts.add(77); // It is NOT synchronized
94
95          System.out.println(ts+"\n");
96
97          for(Object o:ts)
98              System.out.print(o+" ");
99          System.out.println();
100         Iterator<Object> i=ts.descendingIterator();
101         while(i.hasNext())
102         {
103             System.out.print(i.next()+" ");
104         }
105         System.out.println("\n---------------");
106         System.out.println("headSet() : "+ts.headSet(10));
107         System.out.println("tailSet() : "+ts.tailSet(10));
108
109     }
110     public static void main(String[] args)
111     {
112         ClassA aobj=new ClassA();
```

```
Implementing TreeSet Interface

[1, 5, 7, 10, 11, 77]

1 5 7 10 11 77
77 11 10 7 5 1
---------------
headSet() : [1, 5, 7]
tailSet() : [10, 11, 77]
```

```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.ListIterator;
import java.util.TreeSet;

public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing Set Interface\n");

        //HashSet<Object> set=new HashSet<Object>();// Insertion order is NOT maintained (Java 1.2v)
        LinkedHashSet<Object> set=new LinkedHashSet<Object>();// Insertion order is  maintained (Java 1.4v)

        set.add(10);
        set.add("Java");// Heterogeneous data is allowed
        set.add(null);// null value is allowed
        set.add(10);// Duplicates are NOT allowed
        set.add('A');
        set.add(true);// Its default capacity is 16 [LOAD FACTOR : 0.75]
        set.add(1);// Its size increases by DOUBLE
        set.add(77); // It is NOT synchronized



        System.out.println(set);

        System.out.println("\nsize() : "+set.size());
        //System.out.println("get() : "+set.get(1));// C.E
        /*
        In set implementation classes data will not be stored basing
        on index positions. It uses hash values to store the data
        */

        System.out.println("\nWe cant reterive the data in Set implementation classes by using for loop");

        System.out.println("\nReteriving the data by using for-each loop");
        for(Object o:set)
        {
            System.out.print(o+" ");
        }
        System.out.println("\n\nReteriving the data by using Iterator Interface");
        Iterator<Object> i=set.iterator();
```

```
46          while(i.hasNext())
47          {
48              System.out.print(i.next()+" ");
49          }
50
51          System.out.println("\n\nReteriving the data by using List-Iterator Interface");
52          //set.listIterator();// C.E because by using ListIterator we cant reterive the data from Set
53
54          ArrayList<Object> al1=new ArrayList<Object>();
55          for(Object o:set)
56              al1.add(o);
57
58          ArrayList<Object> al2=new ArrayList<Object>(set);
59
60          System.out.println("set : "+set);
61          System.out.println("al1 : "+al1);
62          System.out.println("al2 : "+al2+"\n");
63
64          ListIterator<Object> li=al1.listIterator(al1.size());
65          while(li.hasPrevious())
66          {
67              System.out.print(li.previous()+" ");
68          }
69
70          System.out.println("\n-------------------------");
71          ArrayList<String> al3=new ArrayList<String>();
72          al3.add("Kishan");
73          al3.add("Ahmed");
74          al3.add("Raju");
75          al3.add("Kishan");
76          HashSet<String> hs=new HashSet<String>(al3);
77          System.out.println("al3 : "+al3);
78          System.out.println("hs : "+hs);
79      }
80      void meth2()
81      {
82          System.out.println("Implementing TreeSet Interface\n");
83
84          TreeSet<Object> ts=new TreeSet<Object>();
85
86          ts.add(10); // Insertion order is NOT maintained, but sorting order is maintained(Ascending)
87          ts.add(11);// Heterogeneous data is NOT allowed
88          //ts.add(null);// null value is NOT Allowed
89          ts.add(1);// Duplicates are NOT allowed
90          ts.add(5);// It is available from Java 1.2v
91          ts.add(7);// Its default capacity is 16 [LOAD FACTOR : 0.75]
92          ts.add(1);// Its size increases by DOUBLE
93          ts.add(77); // It is NOT synchronized
```

```
94
95          System.out.println(ts+"\n");
96
97          for(Object o:ts)
98              System.out.print(o+" ");
99          System.out.println();
100         Iterator<Object> i=ts.descendingIterator();
101         while(i.hasNext())
102         {
103             System.out.print(i.next()+" ");
104         }
105         System.out.println("\n---------------");
106         System.out.println("headSet() : "+ts.headSet(10));
107         System.out.println("tailSet() : "+ts.tailSet(10));
108     }
109     public static void main(String[] args)
110     {
111         ClassA aobj=new ClassA();
112         //aobj.meth1();
113         aobj.meth2();
114     }
115 }
```

Pass the user defined class object into all set classes and retrieve it back (prefer next class)

When you are passing user defined class object into hashset fine.

When your passing user defined class object into linked hashset fine.

When you are passing user defined class object into treeset not fine we will get an exception (class cast exception) why because tree set maintains the data in a sorting order for example we are passing employe class having employename

Employee salary and employee branch in it we need to inform otherwise we will get exception. We can do this by using a concept called comparable or comparator interface (coming classes)