→ Customized Sorting of TreeSet :→

```
import java.util.*;
public class Test
    {
        p s v m (String[] args)
{
  TreeMap t = new TreeMap (new MyComparator());
  t.put("xxx", 10);   t.put("AAA", 20);
  t.put("zzz" 30);    t.put("LLL", 40);
        s.o.p (t);
  // {zzz=30, XXX=10, LLL = 40, AAA = 20}
}
}
class MyComparator implements Comparator
  {
  public Int compare (Object Obj1, Object Obj2)
{
  String S1= Obj1.toString();
  String S2= Obj2.toString();

  return S2.compareTo(S1);
}}
```

Ch-14 HashTable :→

→ The underlined data structure is HashTable.
→ Insertion order is not preserved and it is based on hashcode of keys.
→ Duplicate keys are not allowed but duplicate values are allowed.
→ Heterogenious Objects are allowed for both keys and values.
→ Null is not allowed for both key and value otherwise we will get RTE saying NPE.
→ It implements Serializable and Cloneable interfaces but not Random Access Interface.

→ Every method present in HashTable is synchronized and hence, HashTable object is a Thread Safe.

→ HashTable is the best choice if our frequent oprn is Retrieval / search oprn.

→ Constructors :→

(I) HashTable h = new HashTable();
creates an empty HashTable object with default initial capacity 11 and default field ratio is 0.75.

(II) HashTable h = new HashTable(int initialCapacity);

(III) HashTable h = new HashTable
(int initialCapacity, float fillRatio);

(IV) HashTable h = new HashTable(Map m);

Eg:-

```
import java.util.*;
class Test {
    p s v m (String[] args)
{
HashTable h = new Hashtable();
h.put(new Temp(5), "A");
- - - - - - - - (2), "B");
- - - - - - (6, "C");
- - - - - - (15, "D");   15%11 = 4
- - - - - - (23, "E");   23%11 = 1
- - - - - - (16, "F");   16%11 = 5
// h.put("durga", null); →NPE
S.o.p(h);
}
class Temp {
    int i;
Temp(int i)
{ this.i = i;
}
```