But in the case of WeakHashMap, if object doesn't contain any references, it is eligible for GC even though object associated with WeakHashMap i.e. Garbage Collector dominates WeakHashMap.

Ex:

```
import java.util.*;
public class Test
{
    p s v m (String[] args) throws Exception
{
    HashMap m = new HashMap();
    Temp t = new Temp();
    m.put(t, "durga");
    S.O.p(m);
    t = null;
    System.gc();
    Thread.sleep(5000);
    S.O.p(m);
}}

class Temp
{
    public String toString()
    {
        return "temp";
    }
    public void finalize()
    {
        S.O.p("Finalize method called");
    }
}
```

⇒ In this Example, temp object not eligible for GC bcoz, it is associated with HashMap In this case o/p is {temp=durga}.

⇒ In the above prog, if we replace HashMap with WeakHashMap then temp object is eligible for GC. In this case o/p is : {}

⇒ SortedMap :-

→ It is the child interface of Map.
→ If we want to represent a group of key-value pairs a/c to some sorting order of keys then we should go for SortedMap.
→ Sorting is based on the key, but not based on value.
→ SortedMap defines the following specific methods :-

```
Object firstKey();
Object lastKey();
SortedMap headMap(Object key)
SortedMap tailMap(Object key)
SortedMap subMap(Object key1, Object key2)
Comparator comparator()
```

101 – A
103 – B
104 – C
107 – D
125 – E
136 – F

firstKey() → 101
lastKey() → 136
headMap(107) → {101=A, 103=B, 104=C}
tailMap(107) →
{107=D, 125=E, 136=F}
subMap(103, 125) →
{103=B, 104=C, 107=D}
comparator() → null

⇒ TreeMap :-

(1) The underlined data structured is RED-BLACK Tree
(2) Insertion Order is not preserved. and it is based on some sorting order of keys.
(3) Duplicate keys are not allowed, but values can be duplicated.
(4) If we are depending on default natural sorting order then keys should be homogeneous and