**Collection Framework – Part\_13**

* **SortedMap:**

It is the child interface of Map.

If we want to represent a group of key-value pairs according 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();

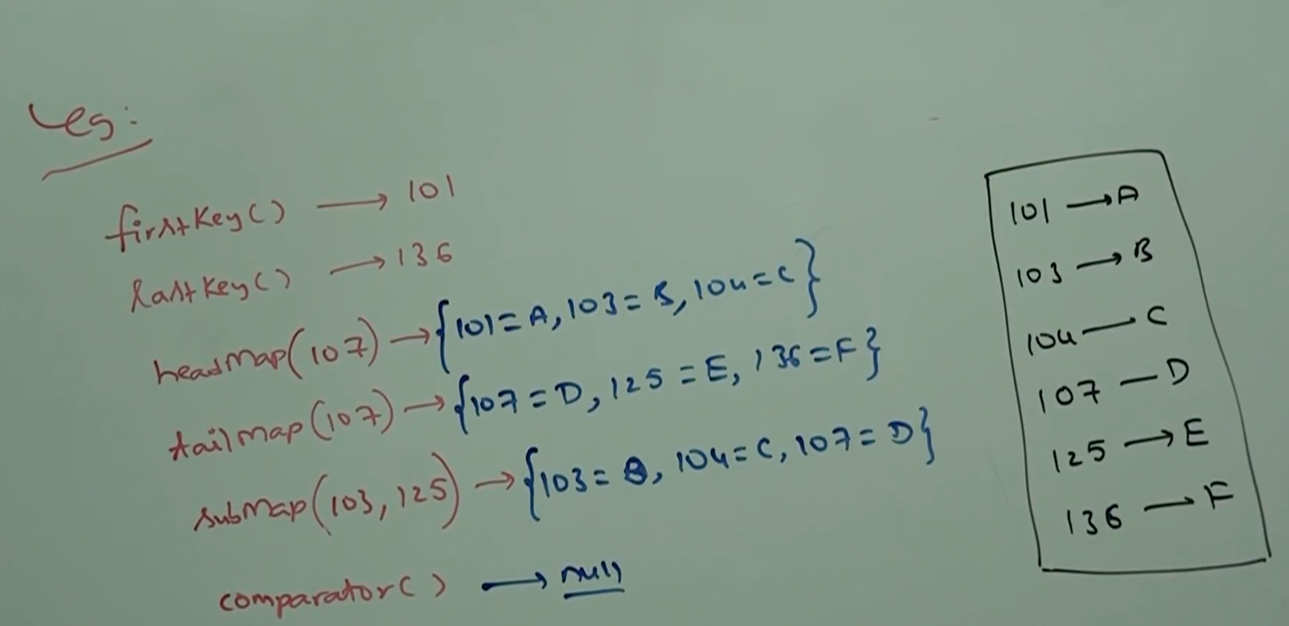
SortedMap headMap(Object key);

SortedMap tailMap(Object key);

SortedMap submap(Object key1, Object key2);

Comparator comparator();

* **Example:**

****

* **TreeMap:**

1. The underlying data structure 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 comparable, otherwise we will get runtime exception saying: ClassCastException.
5. If we are defining our own sorting by Comparator then keys need not be homogeneous and comparable. We can take heterogeneous non comparable objects also.
6. Whether depending on default natural sorting order or customized sorting order there are no restrictions for values, we can take heterogeneous non-comparable objects also.

* **Null acceptance:**

1. For non empty TreeMap if we are trying to insert an entry with null key, then we will get runtime exception saying: NullPointerException.
2. For empty TreeMap as the first entry with null key is allowed, but after inserting that entry if we are trying to insert any other entry then we will get runtime exception saying: NullPointerException.

Note:

The above null acceptance rules applicable until 1.6 version only from 1.7 version onward, null is not allowed for key. But for values we can use null any number of times, there is no restriction whether it is 1.6 or 1.7 version.

* **Constructors:**

TreeMap t = new TreeMap();

For default natural sorting order.

TreeMap t = new TreeMap(Comparator c)

For customized sorting order.

TreeMap t = new TreeMap(Map m);

TreeMap t = new TreeMap(SortedMap m);

* **Demo program for default natural sorting order:**

import java.util.\*;

class TreeMapDemo{

public static void main(String[] args){

TreeMap m = new TreeMap();

m.put(100, “ZZZ”);

m.put(103, “YYY”);

m.put(101, “XXX”);

m.put(104, 106);

//m.put(“FFF”, “XXX”); //CCE

//m.put(null, “XXX”); // NPE

System.out.println(m);

}

}

Output: {100=ZZZ, 101=XXX, 103=YYY, 104=106}

* **Demo program for customized sorting:**

import java.util.\*;

class TreeMapDemo{

public static void main(String[] args){

TreeMap m = new TreeMap(new MyComparator());

m.put(“XXX”, 10);

m.put(“AAA”, 20);

m.put(“ZZZ”, 30);

m.put(“LLL”, 40);

System.out.println(m);

}

}

class MyComparator implements Comparator{

public int compare(Object obj1, Object obj2){

String s1 = obj1.toString();

String s2 = obj2.toString();

return s1.compareTo(s2);

}

}

Output: {ZZZ=30, XXX=10, LLL=40, AAA=20}