**Collection Framework – Part\_09**

* **Comparator(I):**

Comparator presents in java.util package and it defines two methods.

Compare() & equals()

public int compare(Object obj1, Object obj2)

Returns -ve iff obj1 has to come before obj2.

Returns +ve iff obj1 has to come after obj2.

Returns 0 iff obj1 and obj2 are equal.

public boolean equals(Object obj)

Whenever we are implementing Comparator interface compulsory we should provide implementation only for compare() method and we are not required to provide implementation for equals() method because, it is already available to our class from Object class through inheritance.

* **Write a program to insert Integer object into the TreeSet where the sorting order is descending order.**

Note: printing order of TreeSet [left, root, right]

import java.util.\*;

class TreeSetDemo{

public static void main(String[] args){

TreeSet t = new TreeSet(new MyComparator()); //(1)

t.add(10);

t.add(0);

t.add(15);

t.add(5);

t.add(20);

t.add(20);

System.out.println(t);

}

}

class MyComparator implements Comparator{

public int compare(Object obj1, Object obj2){

Integer I1 = (Integer) obj1;

Integer I2 = (Integer) obj2;

if(I1 < I2)

return +1;

else if(I1 > I2)

return -1;

else

return 0;

}

}

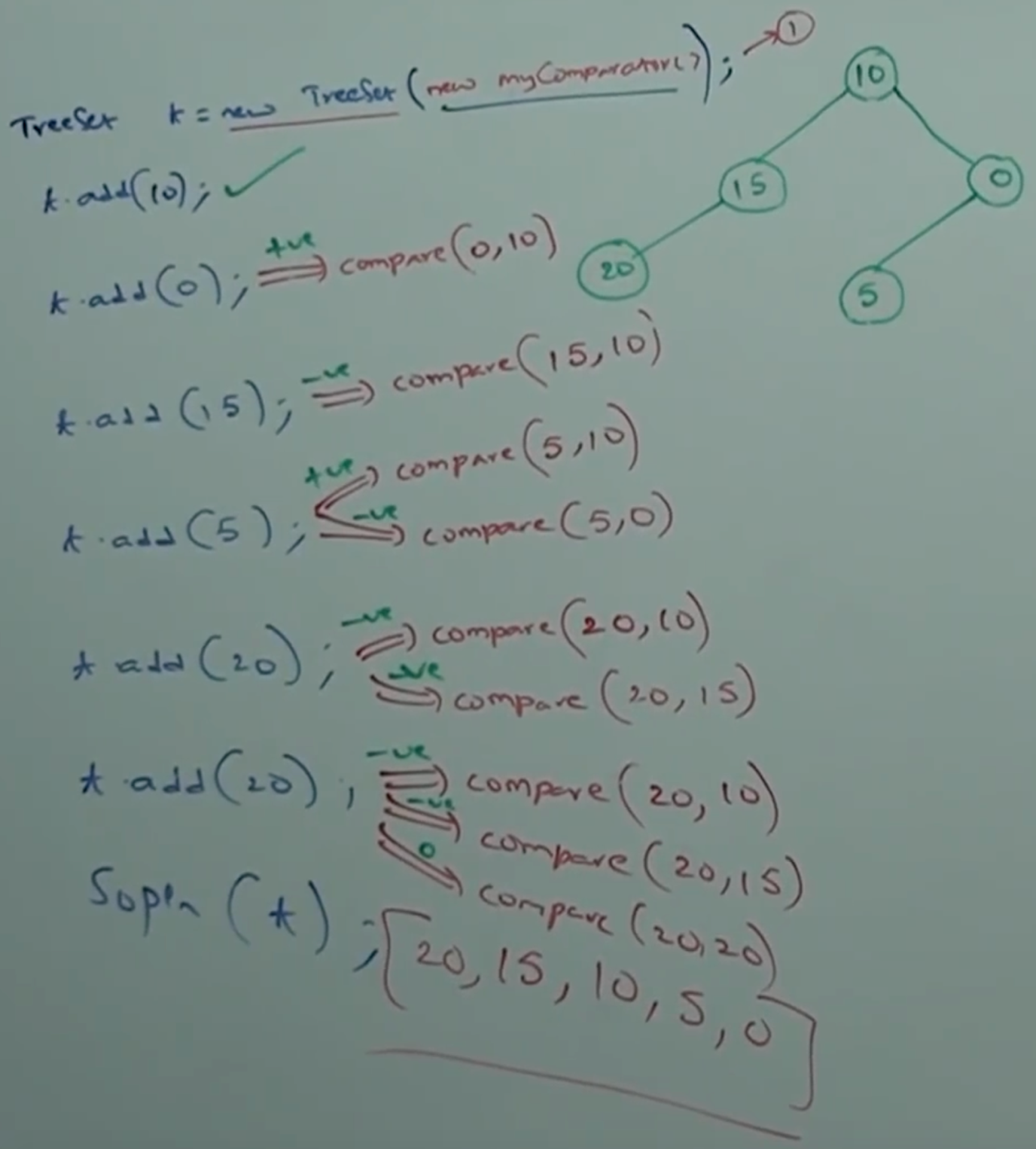
Note: For the first element insertion there won’t be any comparison.

At line1 if we are not passing Comparator object, then internally JVM will call compareTo() method which is meant for default natural sorting order. In this case the output is [0, 5, 10, 15, 20]

At line1 if we are passing Comparator then JVM will call compare() method which is meant for customized sorting. In this case output is

[20,15,5,10,0];

Internal method call:



* **Various possible implementations of compare():**

public int compare(Object obj1, Object obj2){

Integer I1 = (Integer) obj1;

Integer I2 = (Integer) obj2;

1. return I1.compareTo(I2);

Default natural sorting order [Ascending order]

[0,5,10,15,20]

1. return -I1.compareTo();

[Descending order] [20, 15, 10, 5, 0]

1. return I2.compareTo(I1);

[Descending order] [20,15,10,5,0]

1. return -I2.compareTo(I1);

[Ascending Order] [0,5,10,15,20]

1. return +1; [Insertion Order] [10,0,15,520,20]
2. return -1;

[Reverse of Insertion order] [20,20,5,15,0,10]

1. return 0;

[Only first element will be inserted and remaining all are considered as duplicate] [10]

}