**Comparable vs Comparator**

If you want to sort list of object then comparable interface is used. (Primarily used to sort complex data types).

.sort method also takes Comparable object as argument and perform operation on it.

class Laptop implements comparable {

String comp;

int ram;

int hdd;

// Here are all getters and setters for this....

// Now to use comparable and sort this complex data type (object of laptop class)

// We need to implement comparable iterface to it.

@override

public int compareTo(Laptop lap){

/\* This method takes in one para of the complex data type

and compare this with the object of this class using which this method will being called

and to refer that object using which it will be called we use 'this' keyword...

\*/

if(this.getRam() > lap.getRam()){

// this > lap = +

// this < lap = -

// this == lap = 0

return 1;

}else{

return -1;

}

return 0;

}

}

Now, let’s suppose the laptop class does not contain logic for compareTo and does not implement comparable interface. Lets say we get this class from somewhere else or from some library so we cannot change the source code of laptop class, so in that case how will we sort object of laptop class. Then ans is using comparator.

// Now use this in main class

class Main{

public static void main(String args[]){

List<Laptop> lap = new ArrayList<>();

lap.add('acer',4,500);

lap.add('asus',8,1000);

lap.add('dell',12,320);

lap.add('hp',4,500);

// Now to Sort we call sort method on collection and internally it calls laptop class and sort element using

// compareTo method defined in laptop class...

Collection.sort(lap);

}

}

The sort() method takes in either the collection to sort and also takes in comparator object.

Sort method is only concerned with the logic to sort whether you pass that using comparable interface or comparator object.

Another scenario is, lets you have comparable interface implemented in laptop class but there you have defined logic to sort element-using RAM. Now you want to sort laptops based on hard disk without changing logic of laptop class then also you can use comparator.

Now the main class becomes something like this with comparator

// Now use this in main class

class Main{

public static void main(String args[]){

List<Laptop> lap = new ArrayList<>();

lap.add('acer',4,500);

lap.add('asus',8,1000);

lap.add('dell',12,320);

lap.add('hp',4,500);

/\*\*

Sort method takes in two parameters first is the collection (can be sort if comparable interface is implemented)

and second is comparator object if comparable interface is not implemented or we need to define some other sorting logic...

\*\*/

// Create Object of Comparator ...

Comparator<Laptop> con = new Comparator<Laptop>(){

public int compare(Laptop l1, Laptop l2){

if(l1.getHarddisk > l2.getHarddisk()){

return 1;

}else{

return -1;

}

return 0;

}

}

Collection.sort(lap, con);

}

}