Collection(I)

Inside this interface, the commonly used method required for all the collection classes is present
a. boolean add(object o)=> Only one object
b. boolean addAll(Collection c)=>To add group of Object
c. boolean remove(Object o) => to remove particular object
d. boolean removeAll(Collection c)=> to remove particular group of collection
e. void clear() => to remove all the object
f. int size() => to check the size of the array
g. boolean retainAll(Collection c) => except this group of objects remaining all objects should be removed.
h. boolean contains(Object o) => to check whether a particular object exists or not
i. boolean containsAll(Collection c) => To check whether a particular Collection exists or not
j. boolean isEmpty() => To check whether the Collection is empty or not
k. Object[] toArray()=> Convert the object into Array.
l. Iterator iterator() => cursor need to iterate the collection object

Note :There is no concrete class which implements Collection interface directly.

Comparator vs Comparable Interface

Comparator

1. It is an interface present in java.util package
2. It contains 2 abstract method
   public abstract int compare(Object obj1,Object obj2)
   public abstract boolean equals(Object o)
3. int compare(Object obj1,Object obj2)
   |=> return -ve iff obj1 has to come before obj2
   |=> return +ve iff obj1 has to come after obj2
   |=> return 0 both are equal
4. Whenever we are implementing an Comparator interface compulsorily we should give body for compare().
5. Whereas for equals(),we get the body from Object class through inheritance.

eg:

```
 class MyComparator implements Comparator{
 public int compare(Object obj1,Object obj2){
  ....
  ....
       }
 }
```

Comparable(I)

=> It is a part of java.lang package
=> It contains only one method compareTo.
    public int compareTo(Object o)

=> obj1.compareTo(obj2)
    returns -ve iff obj1 has to come before obj2
    returns +ve iff obj1 has to come after obj2
    returns 0 if both are equal

eg#1.

```
System.out.println("A".compareTo("Z"));//A should  come before Z so -ve
System.out.println("Z".compareTo("K"));//Z should  come after  K so +ve
System.out.println("A".compareTo("A"));//Both are equal zero
System.out.println("A".compareTo(null));//NullPointerException
```

Comparable
=> compareTo()
  It is meant for the default natural sorting order.

Comparator
=> compare()
  It is meant for customized sorting order.

Scenario

When to go for Comparable and Comparator?

1st category

Predefined Comparable classes like String and Wrapper class
  => Default natural sorting order is already available
  => If not satisfied, then we need to go for Comparator

2nd Category

Predefined NonComparable classes like StringBuffer
  => Default natural sorting order not available so go for Comparator only aways

3rd Category

Our Own classes like Employee,Student,Customer
  =>Person who is writing this classes are responsible for implementing comparable interface to promote Natural sorting order.
  =>Person who is using this class,can define his own natural sorting order by implementing Comparator interface.

Comparable and Comparator

Comparable	=> Meant for default natural sorting order
Comparator	=> Meant for customized sorting order
Comparable	=> part of java.lang package
Comparator	=> part of java.util package
Comparable	=> only one method compareTo()
Comparator	=> 2 methods compare(),equals()
Comparable	=> It is implemented by Wrapper class and String class
Comparator	=> It is implemented by Collator and RuleBaseCollator(GUI based API)