TUTORIAL-9

Example-1: Consider the following java classes/interfaces group them into the categories (a) Collection Interface, (b) Concrete Implementation, (c) Support Interface.

Collection, Iterator, AbstractSequentialList, HashSet, ArrayList, Vector, SortedSet, SortedMap, Enumeration, Map, HashTable, TreeSet.

Collection Interface: (i) Collection, (ii) SortedSet.
Concrete Implementation: (i) HashSet, (ii) Vector, (iii) ArrayList, (iv) TreeSet
Support Interface: (i)Iterator, (ii) SortedMap, (iii) Enumeration, (iv) Map

Note: Concrete Implementation is the implementing class for an interface or an abstract class. When we need an object of an interface or an abstract class we have to create its implementing class object (because we can't create object of an interface or an abstract class). e.g. Collection c= new HashSet() or . Collection c= new Vector() or . Collection c= new ArrayList() or . Collection c= new TreeSet()
Support Interface does not belong to the collection frame work but they support collection frame work.

Example-2: Consider the following java code, write the expected.

```
String[] words= {"i", "came", "i", "saw", "i", "conquered"};
Set s= new TreeSet();
List l=new LinkedList();
Collection c=new HashSet();
for(int i=0;i<words.length;i++){
        s.add(words[i]);
        l.add(words[i]);
        }
Iterator it=s.iterator();
while(it.hasNext())
        System.out.println(it.next());
Iterator it=l.iterator();
while(it.hasNext())
        System.out.println(it.next());
l.removeAll(s);
Iterator it=l.iterator();
while(it.hasNext())
        System.out.println(it.next());
```

1st output:

```
 came
conquered
i
saw
```

2nd output:

```
i
came
i
saw
i
conquered
```

3rd output:

prints nothing because the list is empty due to remove of all elements.

Note: Because List can contain duplicate elements and it does not maintain the ordering of the elements, in the above example all the elements are added in the list without ordering.

Set can not contain duplicate elements. TreeSet maintain ordering of the elements but HashSet does not maintain the ordering of the elements. In the above example since TreeSet is used, ordering of the elements are maintained.

Example-3: Consider the following java code:

```
public static void foo(Collection c, Object ob){
        int s0=c.size();
        boolean b1=c.add(ob);
        boolean b2=c.add(ob);
        int s1=c.size()- s0;
        boolean b3=c.remove(ob);
        int s2= c.size() - s0;
        System.out.println("b1="+b1+" b2="+b2+" s1="+s1+" b3="+b3+" s2="+s2);
        }
```

What is the output of the method foo() when called with

(i) an empty List (ii) an empty Set (iii) a List that already contains ob (iv) a Set that already contains ob.

Output:

```
(i)     b1=true b2=true s1=2 b3=true s2=1

(ii)     b1=true b2=false s1=1 b3=true s2=0

(iii)   b1=true b2=true s1=2 b3=true s2=1

(iv)    b1=false b2=false s1=0 b3=true s2= -1
```

Note:
When the foo() method is called with an empty List, the initial size s0=0 (because the list is empty), b1=true (add() is a boolean method, it returns true if it can add otherwise it returns false),
b2 =true (because it is possible to add duplicate element in the list), s1=2 (because size() returns 2 and s0=0), b3 = true(because remove() is also a boolean method like add()), s2 = 1 (because after removing one element size() will return 1 and s0 =0).

When the foo() method is called with an empty Set, the initial size s0=0 (because the set is empty), b1=true (because it is possible to add the object in the set), b2 =false (because it is not possible to add duplicate element in the set), s1=1 (because the set is containing 1 element and s0=0), b3 = true(because it is possible to remove the element), s2 = 0 (because after removing one element size() will return 0 and s0 =0).

When the foo() method is called with a List that already contains ob, the initial size s0=1 (because the list is containing 1 element), b1=true (because it is possible to add duplicate element in the list),
b2 =true (because it is possible to add duplicate element in the list), s1=2 (because list is containing 3 elements and s0=1), b3 = true(because it is possible to remove the element), s2 = 1 (because after removing one element size() will return 2 and s0 =1).

When the foo() method is called with a Set that already contains ob, the initial size s0=1 (because the set is containing 1 element), b1=false (because it is not possible to add duplicate element in the set),
b2 =false (because it is not possible to add duplicate element in the set), s1=0 (because list is containing 1 elements and s0=1), b3 = true(because it is possible to remove the element), s2 = -1 (because after removing one element size() will return 0 and s0 =1).

Example-4: Write a method which return a Collection containing all the unique elements from an array which is used as an argument of the method.

```
public Collection unique(Object[] ob){

    List ls=Arrays.asList(ob);
    Collection s=new HashSet();
    s.addAll(ls);
    return s;
    }
```

Note: asList() is a static method of class Arrays and it takes an Object array as a parameter and return List object which contain all the elements of the array. For filtering all the duplicate elements, we have added all the elements of the list to the HashSet. (because we already know HashSet can’t contain duplicate elements).

Example-5: Write a method that uses an iterator to remove all the even integers from a collection of Integers.

```
public void removeEven(Collection c){

  Iterator it = c.iterator();
  while( it.hashNext()){
    Object ob=it.next();
    int n=((Integer)ob).intValue();
    if( n%2 == 0)
        it.remove();
    }
  }
```