ArrayList
Program 1

```
import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListIterator {

	public static void main(String a[]){
		ArrayList<String> arrl = new ArrayList<String>();
		//adding elements to the end
		arrl.add("First");
		arrl.add("Second");
		arrl.add("Third");
		arrl.add("Random");
		Iterator<String> itr = arrl.iterator();
		while(itr.hasNext()){
			System.out.println(itr.next());
		}
	}
```

Program 2

```
import java.util.ArrayList;
import java.util.List;

public class ArrayLst {

    public static void main(String... args)
    {
        ArrayList al = new ArrayList();

        al.add("Java4s");
        al.add(12);
        al.add(12.54f);

        for(int i=0;i<al.size();i++)
        {
            Object o = al.get(i);
            System.out.println("Value is "+o.toString());
        }

    }
}
```

Program 3

```
class Student{
  int rollno;
  String name;
  int age;
  Student(int rollno,String name,int age){
   this.rollno=rollno;
   this.name=name;
   this.age=age;
  }
}

import java.util.*;
public class TestCollection3{
 public static void main(String args[]){
  //Creating user-defined class objects
  Student s1=new Student(101,"X",23);
  Student s2=new Student(102,"Y",21);
  Student s2=new Student(103,"Z",25);
  //creating arraylist
  ArrayList<Student> al=new ArrayList<Student>();
  al.add(s1);//adding Student class object
  al.add(s2);
  al.add(s3);
  //Getting Iterator
  Iterator itr=al.iterator();
  //traversing elements of ArrayList object
  while(itr.hasNext()){
    Student st=(Student)itr.next();
    System.out.println(st.rollno+" "+st.name+" "+st.age);
  }
 }
}
```

```
101 X 23
102 Y 21
103 Z 25
```

Program 4

```
import java.util.*;
class TestCollection4{
 public static void main(String args[]){
  ArrayList<String> al=new ArrayList<String>();
  al.add("Ravi");
  al.add("Vijay");
  al.add("Ajay");
  ArrayList<String> al2=new ArrayList<String>();
  al2.add("Sonoo");
  al2.add("Hanumat");
  al.addAll(al2);//adding second list in first list
  Iterator itr=al.iterator();
  while(itr.hasNext()){
   System.out.println(itr.next());
  }
 }
}
```

```
Ravi
Vijay
Ajay
Sonoo
Hanumat
```

Java LinkedList Example

```
import java.util.*;
public class TestCollection7{
 public static void main(String args[]){

  LinkedList<String> al=new LinkedList<String>();
  al.add("Ravi");
  al.add("Vijay");
  al.add("Ravi");
  al.add("Ajay");

  Iterator<String> itr=al.iterator();
  while(itr.hasNext()){
   System.out.println(itr.next());
  }
 }
}
```

```
Output:Ravi
       Vijay
       Ravi
       Ajay
```

Java ListIterator Interface

ListIterator Interface is used to traverse the element in backward and forward direction.

```
import java.util.*;
public class TestCollection8{
public static void main(String args[]){
ArrayList<String> al=new ArrayList<String>();
al.add("Amit");
al.add("Vijay");
al.add("Kumar");
al.add(1,"Sachin");
System.out.println("element at 2nd position: "+al.get(2));
ListIterator<String> itr=al.listIterator();
System.out.println("traversing elements in forward direction...");
while(itr.hasNext()){
```

```
System.out.println(itr.next());
}
System.out.println("traversing elements in backward direction...");
while(itr.hasPrevious()){
System.out.println(itr.previous());
}
}
}
```

Output:

```
element at 2nd position: Vijay
traversing elements in forward direction...
Amit
Sachin
Vijay
Kumar
traversing elements in backward direction...
Kumar
Vijay
Sachin
Amit
```

Hash Set

```
import java.util.HashSet;

public class MyBasicHashSet {

    public static void main(String a[]){
        HashSet<String> hs = new HashSet<String>();
        //add elements to HashSet
        hs.add("first");
        hs.add("second");
        hs.add("third");
        System.out.println(hs);
        System.out.println("Is HashSet empty? "+hs.isEmpty());
        hs.remove("third");
        System.out.println(hs);
        System.out.println("Size of the HashSet: "+hs.size());
        System.out.println("Does HashSet contains first element?
                                "+hs.contains("first"));
    }
}
```

Basic Operation on treeset

```
import java.util.TreeSet;

public class MyBasicTreeset {

    public static void main(String a[]){

        TreeSet<String> ts = new TreeSet<String>();
        ts.add("one");
        ts.add("two");
        ts.add("three");
        System.out.println("Elements: "+ts);
        //check is set empty?
        System.out.println("Is set empty: "+ts.isEmpty());
        //delete all elements from set
        ts.clear();
        System.out.println("Is set empty: "+ts.isEmpty());
        ts.add("one");
        ts.add("two");
        ts.add("three");
        System.out.println("Size of the set: "+ts.size());
        //remove one string
        ts.remove("two");
        System.out.println("Elements: "+ts);
    }
}
```

OUTPUT

Elements: [one, three, two]

Is set empty: false

Is set empty: true

Size of the set: 3

Elements: [one, three]

Java Map Example:

```
import java.util.*;
public class MapExample1 {
public static void main(String[] args) {
    Map map=new HashMap();
    //Adding elements to map
    map.put(1,"Amit");
    map.put(5,"Rahul");
    map.put(2,"Jai");
    map.put(6,"Amit");
    //Traversing Map
    Set set=map.entrySet();//Converting to Set so that we can traverse
    Iterator itr=set.iterator();
    while(itr.hasNext()){
        //Converting to Map.Entry so that we can get key and value separately
        Map.Entry entry=(Map.Entry)itr.next();
        System.out.println(entry.getKey()+" "+entry.getValue());
    }
}
}
```

```
1 Amit
2 Jai
5 Rahul
6 Amit
```

VECTOR DEMO

```
import java.util.*;
public class VectorDemo {

  public static void main(String args[]) {
    // initial size is 3, increment is 2
    Vector v = new Vector(3,2);
    System.out.println("Initial size: " + v.size());
    System.out.println("Initial capacity: " + v.capacity());

    v.addElement(new Integer(1));
    v.addElement(new Integer(2));
    v.addElement(new Integer(3));
    v.addElement(new Integer(4));
    System.out.println("Capacity after four additions: " + v.capacity());

    v.addElement(new Double(5.45));
    System.out.println("Current capacity: " + v.capacity());

    v.addElement(new Double(6.08));
    v.addElement(new Integer(7));
    System.out.println("Current capacity: " + v.capacity());

    v.addElement(new Float(9.4));
    v.addElement(new Integer(10));
    System.out.println("Current capacity: " + v.capacity());

    v.addElement(new Integer(11));
    v.addElement(new Integer(12));
    System.out.println("First element: " + (Integer)v.firstElement());
```

```
    System.out.println("Last element: " + (Integer)v.lastElement());

    if(v.contains(new Integer(3)))
      System.out.println("Vector contains 3.");

    // enumerate the elements in the vector.
    Enumeration vEnum = v.elements();
    System.out.println("\nElements in vector:");

    while(vEnum.hasMoreElements())
      System.out.print(vEnum.nextElement() + " ");
    System.out.println();
  }
}
```

```
Initial size: 0
Initial capacity: 3
Capacity after four additions: 5
Current capacity: 5
Current capacity: 7
Current capacity: 9
First element: 1
Last element: 12
Vector contains 3.

Elements in vector:
1 2 3 4 5.45 6.08 7 9.4 10 11 12
```

ArrayList	Vector
1) ArrayList is **not synchronized**.	Vector is **synchronized**.
2) ArrayList **increments 50%** of current array size if number of element exceeds from its capacity.	Vector **increments 100%** means doubles the array size if total number of element exceeds than its capacity.
3) ArrayList is **not a legacy** class, it is introduced in JDK 1.2.	Vector is a **legacy** class.
4) ArrayList is **fast** because it is non-synchronized.	Vector is **slow** because it is synchronized i.e. in multithreading environment, it will hold the other threads in runnable or non-runnable state until current thread releases the lock of object.
5) ArrayList uses **Iterator** interface to traverse the elements.	Vector uses **Enumeration** interface to traverse the elements. But it can use Iterator also.