//Vector class Methods
/*

-In vector class Every method is synchronized
-At a time only one Thread is allow to
operate on Vector object and hence Vector
object is Thread safe.
-Relatively performance is low because
Threads are required to wait.
-Vector class is legacy and introduced in 1.0v

(1)The underlying data structur is resizable array (or) growable array
(2)Duplicate objects are allowed
(3)Insertion order is preserved
(4)Hetrogeneous objects are allowed.
(5)Null insertion is possible

*/

```
import java.util.*;
 class a2
{
   public static void main(String[] args)
   {
   //Constructor-1
      Vector<Integer> al1 = new Vector<>();
   //Creates an empty collection with initial default capacity 10
      System.out.println(al1);
      System.out.println(al1.capacity());
      al1.add(1);
      al1.add(2);
      al1.add(3);
      al1.add(4);
      al1.add(5);
      al1.add(6);
      al1.add(7);
      al1.add(8);
      al1.add(9);
      al1.add(10);
      System.out.println(al1);
      System.out.println(al1.capacity());
```

```
        al1.add(11);
        System.out.println(al1);
        System.out.println(al1.capacity());

    //Constructor-2
    //Vector<Integer> al2 = new Vector<>(int initialCapacity);
        Vector<Integer> al2 = new Vector<>(2);
    //Creates an empty collection with default capacity 2
        System.out.println(al2);
        System.out.println(al2.capacity());
        al2.add(10);
        al2.add(20);
        al2.add(30);
        System.out.println(al2);
        System.out.println(al2.capacity());

    //Constructor-3
    //Vector<Integer> al3 = new Vector<>(int initialCapacity,int
capacityIncrement));
         Vector<Integer> al3 = new Vector<>(1,1);
    //Creates an empty collection with default capacity 1 and will
    //increment by 1 when max limit reached
        System.out.println(al3);
        System.out.println(al3.capacity());
        al3.add(100);
        al3.add(200);
        System.out.println(al3);
        System.out.println(al3.capacity());

    //Constructor-4
    //Vector<Integer> al4 = new Vector<>(collection c);
        Vector<Integer> al4 = new Vector<>(al3);
    //creates a collection that is initialized with the elements of collection
        System.out.println(al4);

    //elementAt(int Index)

        System.out.println(al1);
        System.out.println(al1.elementAt(2));

    //equals(Object o)
        System.out.println(al1.equals(al2));
        System.out.println(al3.equals(al4));
```

```
    //public synchronized Object clone()
        System.out.println(al1.clone());
        Vector<Integer> al5 = (Vector)al1.clone();
        System.out.println(al5);
    }
}
```