EXPERIMENT NO.- 07

AIM: Edit, compile, execute and test a Java program that uses vectors.

THEORY:

Java Vector Class

Java Vector class comes under the java.util package. The vector class implements a growable array of objects. Like an array, it contains the component that can be accessed using an integer index.

Vector is very useful if we don't know the size of an array in advance or we need one that can change the size over the lifetime of a program.

Vector implements a dynamic array that means it can grow or shrink as required. It is similar to the ArrayList, but with two differences

- Vector is synchronized.
- The vector contains many legacy methods that are not the part of a collections framework

Java Vector Class Declaration

1. **public class** Vector<E>
2. **extends** Object<E>
3. **implements** List<E>, Cloneable, Serializable

Java Vector Class Constructors

Vector class supports four types of constructors. These are:

SN	Constructor	Description
1)	vector()	It constructs an empty vector with the default size as 10.
2)	vector(intinitialCapacity)	It constructs an empty vector with the specified initial capacity and with its capacity increment equal to zero.
3)	vector(intinitialCapacity, intcapacityIncrement)	It constructs an empty vector with the specified initial capacity and capacity increment.
4)	Vector(Collection<? extends E> c)	It constructs a vector that contains the elements of a collection c.

Java Vector Class Methods

The following are the list of vector class methods:

SN	Method	Description
1)	add()	It is used to append the specified element in the given vector.
2)	addAll()	It is used to append all of the elements in the specified collection to the end of this Vector.
3)	addElement()	It is used to append the specified component to the end of this vector. It increases the vector size by one.
4)	capacity()	It is used to get the current capacity of this vector.
5)	clear()	It is used to delete all of the elements from this vector.
6)	clone()	It returns a clone of this vector.
7)	contains()	It returns true if the vector contains the specified element.
8)	containsAll()	It returns true if the vector contains all of the elements in the specified collection.
9)	copyInto()	It is used to copy the components of the vector into the specified array.
10)	elementAt()	It is used to get the component at the specified index.
11)	elements()	It returns an enumeration of the components of a vector.
12)	ensureCapacity()	It is used to increase the capacity of the vector which is in use, if necessary. It ensures that the vector can hold at least the number of components specified by the minimum capacity argument.

13)	equals()	It is used to compare the specified object with the vector for equality.
14)	firstElement()	It is used to get the first component of the vector.
15)	forEach()	It is used to perform the given action for each element of the Iterable until all elements have been processed or the action throws an exception.
16)	get()	It is used to get an element at the specified position in the vector.
17)	hashCode()	It is used to get the hash code value of a vector.
18)	indexOf()	It is used to get the index of the first occurrence of the specified element in the vector. It returns -1 if the vector does not contain the element.
19)	insertElementAt()	It is used to insert the specified object as a component in the given vector at the specified index.
20)	isEmpty()	It is used to check if this vector has no components.
21)	iterator()	It is used to get an iterator over the elements in the list in proper sequence.
22)	lastElement()	It is used to get the last component of the vector.
23)	lastIndexOf()	It is used to get the index of the last occurrence of the specified element in the vector. It returns -1 if the vector does not contain the element.
24)	listIterator()	It is used to get a list iterator over the elements in the list in proper sequence.
25)	remove()	It is used to remove the specified element from the vector. If the vector does not contain the element, it is unchanged.
26)	removeAll()	It is used to delete all the elements from the vector that

		are present in the specified collection.
27)	removeAllElements()	It is used to remove all elements from the vector and set the size of the vector to zero.
28)	removeElement()	It is used to remove the first (lowest-indexed) occurrence of the argument from the vector.
29)	removeElementAt()	It is used to delete the component at the specified index.
30)	removeIf()	It is used to remove all of the elements of the collection that satisfy the given predicate.
31)	removeRange()	It is used to delete all of the elements from the vector whose index is between fromIndex, inclusive and toIndex, exclusive.
32)	replaceAll()	It is used to replace each element of the list with the result of applying the operator to that element.
33)	retainAll()	It is used to retain only that element in the vector which is contained in the specified collection.
34)	set()	It is used to replace the element at the specified position in the vector with the specified element.
35)	setElementAt()	It is used to set the component at the specified index of the vector to the specified object.
36)	setSize()	It is used to set the size of the given vector.
37)	size()	It is used to get the number of components in the given vector.
38)	sort()	It is used to sort the list according to the order induced by the specified Comparator.
39)	spliterator()	It is used to create a late-binding and fail-fast Spliterator over the elements in the list.
40)	subList()	It is used to get a view of the portion of the list between

		fromIndex, inclusive, and toIndex, exclusive.
41)	toArray()	It is used to get an array containing all of the elements in this vector in correct order.
42)	toString()	It is used to get a string representation of the vector.
43)	trimToSize()	It is used to trim the capacity of the vector to the vector's current size.

EXPERIMENT NO.- 07

AIM: Edit, compile, execute and test a Java program that uses vectors.

PROGRAM:

```
import java.util.*;
public class VectorExample1 {
      public static void main(String args[]) {
        //Create an empty vector with initial capacity 4
         Vector<String> vec = new Vector<String>(4);
         //Adding elements to a vector
         vec.add("Tiger");
         vec.add("Lion");
         vec.add("Dog");
         vec.add("Elephant");
         //Check size and capacity
         System.out.println("Size is: "+vec.size());
         System.out.println("Default capacity is: "+vec.capacity());
         //Display Vector elements
         System.out.println("Vector element is: "+vec);
         vec.addElement("Rat");
         vec.addElement("Cat");
         vec.addElement("Deer");
         //Again check size and capacity after two insertions
         System.out.println("Size after addition: "+vec.size());
         System.out.println("Capacity after addition is: "+vec.capacity());
         //Display Vector elements again
    System.out.println("Elements are: "+vec);
   //Checking if Tiger is present or not in this vector
     if(vec.contains("Tiger"))
           {
        System.out.println("Tiger is present at the index " +vec.indexOf("Tiger"));
           }
           else
           {
             System.out.println("Tiger is not present in the list.");
           }
           //Get the first element
         System.out.println("The first animal of the vector is = "+vec.firstElement());
         //Get the last element
         System.out.println("The last animal of the vector is = "+vec.lastElement());
```

```
    }
}
```

Output:

```
Size is: 4
Default capacity is: 4
Vector element is: [Tiger, Lion, Dog, Elephant]
Size after addition: 7
Capacity after addition is: 8
Elements are: [Tiger, Lion, Dog, Elephant, Rat, Cat, Deer]
Tiger is present at the index 0
The first animal of the vector is = Tiger
The last animal of the vector is = Deer
```

PROGRAM:

```
import java.util.*;
public class VectorExample2 {
    public static void main(String args[]) {
     //Create an empty Vector
     Vector < Integer > in = new Vector < > ();
       //Add elements in the vector
     in.add(100);
     in.add(200);
     in.add(300);
     in.add(200);
     in.add(400);
     in.add(500);
     in.add(600);
     in.add(700);
     //Display the vector elements
     System.out.println("Values in vector: " +in);
     //use remove() method to delete the first occurence of an element
     System.out.println("Remove first occourence of element 200: "+in.remove((Integer)200));
     //Display the vector elements afre remove() method
     System.out.println("Values in vector: " +in);
     //Remove the element at index 4
     System.out.println("Remove element at index 4: " +in.remove(4));
     System.out.println("New Value list in vector: " +in);
```

```
        //Remove an element
        in.removeElementAt(5);
        //Checking vector and displays the element
        System.out.println("Vector element after removal: " +in);
        //Get the hashcode for this vector
        System.out.println("Hash code of this vector = "+in.hashCode());
        //Get the element at specified index
        System.out.println("Element at index 1 is = "+in.get(1));
          }
}
```

Output:

```
Values in vector: [100, 200, 300, 200, 400, 500, 600, 700]
Remove first occourence of element 200: true
Values in vector: [100, 300, 200, 400, 500, 600, 700]
Remove element at index 4: 500
New Value list in vector: [100, 300, 200, 400, 600, 700]
Vector element after removal: [100, 300, 200, 400, 600]
Hash code of this vector = 130123751
Element at index 1 is = 300
```

CONCLUSION: Thus we have successfully studied about editing, compiling, executing and testing a Java program that uses vectors.