Vector class

The Vector class available in java.util package implements a growable array of objects.

- Vector implements a dynamic array that means it can grow or shrink as required. Like an array, it contains components that can be accessed using an integer index.
- They are very similar to ArrayList but Vector is synchronised and have some legacy method which collection framework does not contain.

Constructor:

- **Vector():** Creates a default vector of initial capacity is 10.
- **Vector(int size):** Creates a vector whose initial capacity is specified by size.
- **Vector(int size, int incr):** Creates a vector whose initial capacity is specified by size and increment is specified by incr. It specifies the number of elements to allocate each time that a vector is resized upward.
- **Vector(Collection c):** Creates a vector that contains the elements of collection c.

Three ways to create vector class object:

Method 1:

```
Vector vec = new Vector();
```

It creates an empty Vector with the default initial capacity of 10. It means the Vector will be re-sized when the 11th elements needs to be inserted into the Vector. Note: By default vector doubles its size. i.e. In this case the Vector size would remain 10 till 10 insertions and once we try to insert the 11th element It would become 20 (double of default capacity 10).

Method 2:

Syntax:

```
Vector object= new Vector(int initialCapacity)
```

Example:

```
Vector vec = new Vector(3);
```

It will create a Vector of initial capacity of 3.

Method 3:

Syntax:

Vector object= new vector(int initialcapacity, capacityIncrement)

Example:

Vector vec= new Vector(4, 6)

Here we have provided two arguments. The initial capacity is 4 and capacityIncrement is 6. It means upon insertion of 5th element the size would be 10 (4+6) and on 11th insertion it would be 16(10+6).

- **Commonly used methods of Vector Class:**
- **void addElement(Object element):** It inserts the element at the end of the Vector.
- **int capacity():** This method returns the current capacity of the vector.
- **int size():** It returns the current size of the vector.
- **void setSize(int size):** It changes the existing size with the specified size.
- **boolean contains(Object element):** This method checks whether the specified element is present in the Vector. If the element is been found it returns true else false.
- **boolean containsAll(Collection c):** It returns true if all the elements of collection c are present in the Vector.
- **Object elementAt(int index):** It returns the element present at the specified location in Vector.
- **Object firstElement():** It is used for getting the first element of the vector.
- **Object lastElement():** Returns the last element of the array.
- **Object get(int index):** Returns the element at the specified index.
- **boolean isEmpty():** This method returns true if Vector doesn't have any element.
- **boolean removeElement(Object element):** Removes the specifed element from vector.
- **boolean removeAll(Collection c):** It Removes all those elements from vector which are present in the Collection c.
- **void setElementAt(Object element, int index):** It updates the element of specifed index with the given element.

// Java code illustrating add() method

```
import java.util.*;
class Vector_demo
{
    public static void main(String[] arg)
    {
        // create default vector
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("geeks");
        v.add("forGeeks");
        v.add(3);
        System.out.println("Vector is " + v);
    }
}
```

Output:

```
[1, 2, geeks, forGeeks, 3]
```

void add(int index, Object obj): This method inserts the specified element at the specified position in this Vector.

Syntax:

```
public void add(int index, Object obj)
```

```
// Java code illustrating add() method
import java.util.*;
class Vector_demo
{
    public static void main(String[] arg)
    {
```

```
        // create default vector
        Vector v = new Vector();
        v.add(0, 1);
        v.add(1, 2);
        v.add(2, "geeks");
        v.add(3, "forGeeks");
        v.add(4, 3);
        System.out.println("Vector is " + v);
    }
}
```

Output:

Vector is: [1, 2, geeks, forGeeks, 3]

boolean contains(Object o): This method returns true if this vector contains the specified element.

Syntax: public boolean contains(object o)

```
// Java code illustrating contains() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("geeks");
        v.add("forGeeks");
```

```
        v.add(3);
        // check whether vector contains "forGeeks"
        if (v.contains("forGeeks"))
            System.out.println("forGeeks exists");
    }
}
```

Output:

```
  forGeeks exists
```

Object get(int index): This method returns the element at the specified position in this Vector.

Syntax:

```
    public Object get(int index)
```

// Java code illustrating get() methods

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

        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // get the element at index 2
        System.out.println("element at indexx 2 is: " + v.get(2));
    }
```

```
    }
```

Output:

```
element at indexx 2 is: Geeks
```

int indexOf(Object o): This method returns the index of the first occurrence of the specified element in this vector, or -1 if this vector does not contain the element.

Syntax: public int indexOf(Object o)

```
// Java code illustrating indexOf() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // get the element at index of Geeks
        System.out.println("index of Geeks is: " + v.indexOf("Geeks"));
    }
}
```

Output:

```
index of Geeks is: 2
```

boolean isEmpty(): This method tests if this vector has no components.

Syntax: public boolean isEmpty()

```
// Java code illustrating isEmpty() method
import java.util.*;
```

```
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        v.clear();
        // check whether vector is empty or not
        if (v.isEmpty())
            System.out.println("Vector is clear");
    }
}
```

Output:

Vector is clear

int lastIndexOf(Object o): This method returns the index of the last occurrence of the specified element in this vector, or -1 if this vector does not contain the element.

Syntax: public int lastIndexOf(Object o)

Returns: returns the index of the last occurrence of the

specified element in this list, or -1 if this list does not contain the element

```
// Java code illustrating lastIndexof()
import java.util.*;
class Vector_demo
 {
    public static void main(String[] arg)
    {
```

```
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // checking last occurrence of 2
        System.out.println("last occurrence of 2 is: " + v.lastIndexOf(2));
    }
}
```

Output:

last occurrence of 2 is: 1

boolean remove(Object o): This method removes the first occurrence of the specified element in this Vector If the Vector does not contain the element, it is unchanged.

Syntax: public boolean remove(Object o)

Returns: Returns the first occurrence of element

```
// Java code illustrating remove method()
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();

        v.add(1);
        v.add(2);
        v.add("Geeks");
```

```
            v.add("forGeeks");
            v.add(4);
            // removing first occurrence element at 1
            v.remove(1);
            // checking vector
            System.out.println("after removal: " + v);
        }
    }
```

Output:

after removal: [1, Geeks, forGeeks, 4]

boolean equals(Object o): This method compares the specified Object with this Vector for equality.

Syntax: public boolean equal(Object o)

Returns: true if operation succeeded otherwise false.

```
// Java code illustrating equals() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {

        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // second vector
        Vector v_2nd = new Vector();
        v_2nd.add(1);
```

```
            v_2nd.add(2);
            v_2nd.add("Geeks");
            v_2nd.add("forGeeks");
            v_2nd.add(4);
            if (v.equals(v_2nd))
                System.out.println("both vectors are equal");
        }
    }
```

Output:

```
both vectors are equal
```

Object firstElement(): This method returns the first component (the item at index 0) of this vector.

Syntax: public Object firstElement()

```
// Java code illustrating firstElement() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // first element of vector
        System.out.println("first element of vector is: " + v.firstElement());
    }
}
```

Output:

first element of vector is: 1

int size(): This method returns the number of components in this vector.

Syntax: public int size()

Returns: returns the number of components in this vector.

```
// Java code illustrating size() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // size of vector
        System.out.println(" size of vector: " + v.size());
    }
}
```

Output:

size of vector: 5

void setSize(int newSize): This method sets the size.

Syntax: public void setSize(int newSize)

```
// Java code illustrating setSize() method
import java.util.*;
class Vector_demo {
```

```
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
        v.add("Geeks");
        v.add("forGeeks");
        v.add(4);
        // setting new size of vector
        v.setSize(13);
        // size of vector
        System.out.println("size of vector: " + v.size());
    }
}
```

Output:

```
size of vector: 13
```

void setElementAt(Object obj, int index): This method sets the component at the specified index of this vector to be the specified object.

Syntax: public void setElementAt(E obj, int index)

```
// Java code illustrating setElementAt() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        // create default vector of capacity 10
        Vector v = new Vector();
        v.add(1);
        v.add(2);
```

```
            v.add("Geeks");
            v.add("forGeeks");
            v.add(4);
            // set 4 at the place of 2
            v.setElementAt(4, 1);
            System.out.println("vector: " + v);
        }
    }
```

Output:

```
vector: [1, 4, Geeks, forGeeks, 4
```

void removeAllElements(): This method removes all components from this vector and sets its size to zero.

Syntax: public void removeAllElements()

// **Java code illustrating removeAllElements() method**

```
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        Vector vec = new Vector(7);

        // use add() method to add elements in the vector
        vec.add(1);
        vec.add(2);
        vec.add(3);
        vec.add(4);
        vec.add(5);
        vec.add(6);
        vec.add(7);
        // remove all elements
        vec.removeAllElements();
```

```
        // checking vector's size
        System.out.println("Size: " + vec.size());
        // checking vector's componenets
        System.out.println("vector's componenets: " + vec);
    }
}
```

Output:

Size: 0

vector's componenets: []

Object lastElement(): This method returns the last component of the vector.

Syntax: public Object lastElement()

Returns: returns the last component of the vector,

i.e., the component at index size() - 1.

// Java code illustrating lastElement() method

```
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        Vector vec = new Vector(7);
        // use add() method to add elements in the vector
        vec.add(1);
        vec.add(2);
        vec.add(3);
        vec.add(4);
        vec.add(5);
        vec.add(6);
        vec.add(7);

        // checking last element of vector
```

```
        System.out.println("vector's last componenets: " + vec.lastElement());
    }
}
```

Output:

```
    vector's last componenets: 7
```

boolean removeElement(Object obj): This method removes the first occurrence of the argument from this vector.

Syntax: public boolean removeElement(Object obj)

Returns: true if operation is succeeded otherwise false.

```
// Java code illustrating remove Element()
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        Vector vec = new Vector(7);
        // use add() method to add elements in the vector
        vec.add(1);
        vec.add(2);
        vec.add(3);
        vec.add(4);
        vec.add(5);
        vec.add(6);
        vec.add(7);
        // remove an element
        vec.removeElement(5);
        // checking vector
        System.out.println("Vector after removal: " + vec);
    }
}
```

Output:

Vector after removal: [1, 2, 3, 4,

int capacity(): This method returns the current capacity of this vector.

Syntax: public int capacity()

returns: returns the current capacity of the

vector as an integer value. Here capacity means the length of

its internal data array, kept in the field elementData of this vector.

```
// Java code illustrating capacity() method
import java.util.*;
class Vector_demo {
    public static void main(String[] arg)
    {
        Vector vec = new Vector(7);
        // use add() method to add elements in the vector
        vec.add(1);
        vec.add(2);
        vec.add(3);
        vec.add(4);
        vec.add(5);
        vec.add(6);
        vec.add(7);
        // checking capacity
        System.out.println("Capacity of vector: " + vec.capacity());
    }
}
```

Output:

Capacity of vector: 7