Java Vector

The Vector class is an implementation of the List interface that allows us to create resizable-arrays similar to the ArrayList class.

The Vector class implements a growable array of objects. Vectors fall in legacy classes, but now it is fully compatible with collections. It is found in java.util package and implement the List interface.

Creating a Vector

Here is how we can create vectors in Java.

```
Vector<Type> vector = new Vector<>();
```

Here, `Type` indicates the type of a linked list. For example,

```
// create Integer type linked list
Vector<Integer> vector= new Vector<>();

// create String type linked list
Vector<String> vector= new Vector<>();
```

Methods of Vector

The Vector class also provides the resizable-array implementations of the List interface (similar to the ArrayList class). Some of the Vector methods are:

Add Elements to Vector

- `add(element)` - adds an element to vectors
- `add(index, element)` - adds an element to the specified position
- `addAll(vector)` - adds all elements of a vector to another vector

For example,

```
import java.util.Vector;

class Main {
    public static void main(String[] args) {
        Vector<String> mammals= new Vector<>();

        // Using the add() method
        mammals.add("Dog");
        mammals.add("Horse");

        // Using index number
        mammals.add(2, "Cat");
        System.out.println("Vector: " + mammals);

        // Using addAll()
        Vector<String> animals = new Vector<>();
        animals.add("Crocodile");

        animals.addAll(mammals);
        System.out.println("New Vector: " + animals);
    }
}
```

Output

```
Vector: [Dog, Horse, Cat]
New Vector: [Crocodile, Dog, Horse, Cat]
```

Access Vector Elements

- `get(index)` - returns an element specified by the index
- `iterator()` - returns an iterator object to sequentially access vector elements

For example,

```
import java.util.Iterator;
import java.util.Vector;

class Main {
    public static void main(String[] args) {
        Vector<String> animals= new Vector<>();
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");

        // Using get()
        String element = animals.get(2);
        System.out.println("Element at index 2: " + element);

        // Using iterator()
        Iterator<String> iterate = animals.iterator();
        System.out.print("Vector: ");
        while(iterate.hasNext()) {
            System.out.print(iterate.next());
            System.out.print(", ");
        }
    }
}
```

Output

```
Element at index 2: Cat
Vector: Dog, Horse, Cat,
```

Remove Vector Elements

- `remove(index)` - removes an element from specified position
- `removeAll()` - removes all the elements
- `clear()` - removes all elements. It is more efficient than `removeAll()`

For example,

```
import java.util.Vector;

class Main {
    public static void main(String[] args) {
        Vector<String> animals= new Vector<>();
        animals.add("Dog");
        animals.add("Horse");
        animals.add("Cat");

        System.out.println("Initial Vector: " + animals);

        // Using remove()
        String element = animals.remove(1);
        System.out.println("Removed Element: " + element);
        System.out.println("New Vector: " + animals);

        // Using clear()
        animals.clear();
        System.out.println("Vector after clear(): " + animals);
    }
}
```

Output

```
Initial Vector: [Dog, Horse, Cat]
Removed Element: Horse
New Vector: [Dog, Cat]
Vector after clear(): []
```

Others Vector Methods

Methods	Descriptions
`set()`	changes an element of the vector
`size()`	returns the size of the vector
`toArray()`	converts the vector into an array
`toString()`	converts the vector into a String
`contains()`	searches the vector for specified element and returns a boolean result

Example

```
import java.util.*;
public class VectorExample {
    public static void main(String args[]) {
      //Create a vector
      Vector<String> vec = new Vector<String>();
      //Adding elements using add() method of List
      vec.add("Tiger");
      vec.add("Lion");
      vec.add("Dog");
      vec.add("Elephant");
      //Adding elements using addElement() method of Vector
      vec.addElement("Rat");
      vec.addElement("Cat");
      vec.addElement("Deer");

      System.out.println("Elements are: "+vec);
    }
}
```

Output:

```
Elements are: [Tiger, Lion, Dog, Elephant, Rat, Cat, Deer]
```

```
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
```

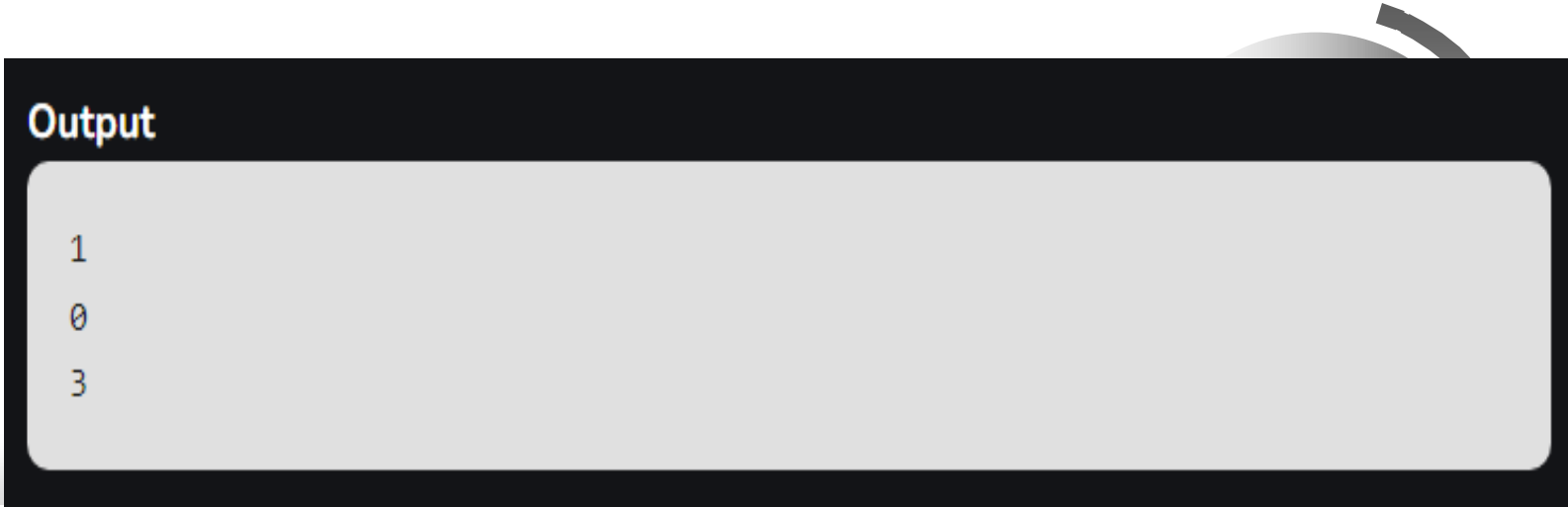
```
 if(vec.contains("Tiger"))
          {
            System.out.println("Tiger is present at the index "
+vec.indexOf("Tiger"));
          }
          else
          {
            System.out.println("Tiger is not present in the list.");
          }
//Get the first element
          System.out.println("The 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
```

```
// Java Program Implementing Vector
import java.util.Vector;

public class VectorExample
{
    public static void main(String[] args)
    {

        // Create a new vector
        Vector<Integer> v = new Vector<>(3, 2);

        // Add elements to the vector
        v.addElement(1);
        v.addElement(2);
        v.addElement(3);
```

```
        // Insert an element at index 1
        v.insertElementAt(0, 1);

        // Remove the element at index 2
        v.removeElementAt(2);

        // Print the elements of the vector
        for (int i : v) {
            System.out.println(i);
        }
    }
}
```



```
// Java code to change the
// elements in vector class
import java.util.*;

// Driver Class
public class UpdatingVector {
      // Main Function
    public static void main(String args[])
    {
        // Creating an empty Vector
        Vector<Integer> vec_tor = new Vector<Integer>();

        // Use add() method to add elements in the vector
        vec_tor.add(12);
        vec_tor.add(23);
        vec_tor.add(22);
        vec_tor.add(10);
        vec_tor.add(20);
```

```
        // Displaying the Vector
        System.out.println("Vector: " + vec_tor);

        // Using set() method to replace 12 with 21
        System.out.println("The Object that is replaced is: "
                + vec_tor.set(0, 21));

        // Using set() method to replace 20 with 50
        System.out.println("The Object that is replaced is: "
                + vec_tor.set(4, 50));

        // Displaying the modified vector
        System.out.println("The new Vector is:" + vec_tor);
    }
}
```