# JAVA.UTIL.VECTOR CLASS

 

## Introduction

The **java.util.Vector** class implements a growable array of objects. Similar to an Array, it contains components that can be accessed using an integer index. Following are the important points about Vector:

- The size of a Vector can grow or shrink as needed to accommodate adding and removing items.
- Each vector tries to optimize storage management by maintaining a *capacity* and a *capacityIncrement*.
- As of the Java 2 platform v1.2, this class was retrofitted to implement the List interface.
- Unlike the new collection implementations, *Vector* is synchronized.
- This class is a member of the Java Collections Framework.

## Class declaration

Following is the declaration for **java.util.Vector** class:

```
public class Vector<E>
   extends AbstractList<E>
   implements List<E>, RandomAccess, Cloneable, Serializable
```

Here <E> represents an Element, which could be any class. For example, if you're building an array list of Integers then you'd initialize it as follows:

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

## Class constructors

| S.N. | Constructor & Description |
|---|---|
| 1 | **Vector**<br><br>This constructor is used to create an empty vector so that its internal data array has size 10 and its standard capacity increment is zero. |
| 2 | **Vector***Collection < ?extendsE > c*<br><br>This constructor is used to create a vector containing the elements of the specified collection, in the order they are returned by the collection's iterator. |
| 3 | **Vector***intinitialCapacity*<br><br>This constructor is used to create an empty vector with the specified initial capacity and with its capacity increment equal to zero. |
| 4 | **Vector***intinitialCapacity, intcapacityIncrement*<br><br>This constructor is used to create an empty vector with the specified initial capacity and |

capacity increment.

## Class methods

| S.N. | Method & Description |
|---|---|
| 1 | boolean add*Ee*<br><br>This method appends the specified element to the end of this Vector. |
| 2 | void add*intindex, Eelement*<br><br>This method inserts the specified element at the specified position in this Vector. |
| 3 | boolean addAll*Collection < ?extendsE > c*<br><br>This method appends all of the elements in the specified Collection to the end of this Vector. |
| 4 | boolean addAll*intindex, Collection < ?extendsE > c*<br><br>This method inserts all of the elements in the specified Collection into this Vector at the specified position. |
| 5 | void addElement*Eobj*<br><br>This method adds the specified component to the end of this vector, increasing its size by one. |
| 6 | int capacity<br><br>This method returns the current capacity of this vector. |
| 7 | void clear<br><br>This method removes all of the elements from this vector. |
| 8 | clone clone<br><br>This method returns a clone of this vector. |
| 9 | boolean contains*Objecto*<br><br>This method returns true if this vector contains the specified element. |
| 10 | boolean containsAll*Collection < ? > c* |

This method returns true if this Vector contains all of the elements in the specified Collection.

11

void copyInto*Object[]anArray*

This method copies the components of this vector into the specified array.

12

E elementAt*intindex*

This method returns the component at the specified index.

13

Enumeration<E> elements

This method returns an enumeration of the components of this vector.

14

void ensureCapacity*intminCapacity*

This method increases the capacity of this vector, if necessary, to ensure that it can hold at least the number of components specified by the minimum capacity argument.

15

boolean equals*Objecto*

This method compares the specified Object with this Vector for equality.

16

E firstElement

This method returns the first component *theitematindex*0 of this vector.

17

E get*intindex*

This method returns the element at the specified position in this Vector.

18

int hashCode

This method returns the hash code value for this Vector.

19

int indexOf*Objecto*

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.

20

int indexOf*Objecto, intindex*

This method returns the index of the first occurrence of the specified element in this vector, searching forwards from index, or returns -1 if the element is not found.

21

void insertElementAt*Eobj, intindex*

This method inserts the specified object as a component in this vector at the specified index.

22

boolean isEmpty

This method tests if this vector has no components.

23

E lastElement

This method returns the last component of the vector.

24

int lastIndexOf*Objecto*

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.

25

int lastIndexOf*Objecto, intindex*

This method returns the index of the last occurrence of the specified element in this vector, searching backwards from index, or returns -1 if the element is not found.

26

E remove*intindex*

This method removes the element at the specified position in this Vector.

27

boolean remove*Objecto*

This method removes the first occurrence of the specified element in this Vector If the Vector does not contain the element, it is unchanged.

28

boolean removeAll*Collection < ? > c*

This method removes from this Vector all of its elements that are contained in the specified Collection.

29

void removeAllElements

This method removes all components from this vector and sets its size to zero.

30

boolean removeElement*Objectobj*

This method removes the first occurrence of the argument from this vector.

31

void removeElementAt*intindex*

This method deletes the component at the specified index.

32

protected void removeRange*intfromIndex, inttoIndex*

This method removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.

33

boolean retainAll*Collection < ? > c*

This method retains only the elements in this Vector that are contained in the specified Collection.

34

E set*intindex, Eelement*

This method replaces the element at the specified position in this Vector with the specified element.

35

void setElementAt*Eobj, intindex*

This method sets the component at the specified index of this vector to be the specified object.

36

void setSize*intnewSize*

This method sets the size of this vector.

37

int size

This method returns the number of components in this vector.

38

List <E> subList*intfromIndex, inttoIndex*

This method returns a view of the portion of this List between fromIndex, inclusive, and toIndex, exclusive.

39

object[ ] toArray

This method returns an array containing all of the elements in this Vector in the correct order.

40

<T> T[ ] toArray*T[ ]a*

This method returns an array containing all of the elements in this Vector in the correct order; the runtime type of the returned array is that of the specified array.

41

String toString

This method returns a string representation of this Vector, containing the String representation of each element.

42

void trimToSize

This method trims the capacity of this vector to be the vector's current size.

## Methods inherited

This class inherits methods from the following classes:

- java.util.AbstractMap
- java.lang.Object
- java.util.List