java.util

Class Vector

java.lang.Object
    java.util.Vector

Direct Known Subclasses:

Stack

```
public class Vector
extends Object
```

The `Vector` class implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a `Vector` can grow or shrink as needed to accommodate adding and removing items after the `Vector` has been created.

Each vector tries to optimize storage management by maintaining a `capacity` and a `capacityIncrement`. The `capacity` is always at least as large as the vector size; it is usually larger because as components are added to the vector, the vector's storage increases in chunks the size of `capacityIncrement`. An application can increase the capacity of a vector before inserting a large number of components; this reduces the amount of incremental reallocation.

Since:

JDK1.0, CLDC 1.0

Version:

12/17/01 (CLDC 1.1)

Field Summary

Fields

Modifier and Type	Field and Description
`protected int`	**`capacityIncrement`** The amount by which the capacity of the vector is automatically incremented when its size becomes greater than its capacity.
`protected int`	**`elementCount`** The number of valid components in the vector.
`protected Object[]`	**`elementData`** The array buffer into which the components of the vector are stored.

Constructor Summary

Constructors

Constructor and Description
`Vector()` Constructs an empty vector.
`Vector(int initialCapacity)` Constructs an empty vector with the specified initial capacity.
`Vector(int initialCapacity, int capacityIncrement)` Constructs an empty vector with the specified initial capacity and capacity increment.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`addElement(Object obj)` Adds the specified component to the end of this vector, increasing its size by one.
`int`	`capacity()` Returns the current capacity of this vector.
`boolean`	`contains(Object elem)` Tests if the specified object is a component in this vector.
`void`	`copyInto(Object[] anArray)` Copies the components of this vector into the specified array.
`Object`	`elementAt(int index)` Returns the component at the specified index.
`Enumeration`	`elements()` Returns an enumeration of the components of this vector.
`void`	`ensureCapacity(int minCapacity)` 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.
`Object`	`firstElement()` Returns the first component of this vector.
`int`	`indexOf(Object elem)` Searches for the first occurrence of the given argument, testing for equality using the `equals` method.
`int`	`indexOf(Object elem, int index)` Searches for the first occurrence of the given argument, beginning the search at `index`, and testing for equality using the `equals` method.
`void`	`insertElementAt(Object obj, int index)` Inserts the specified object as a component in this vector at the specified `index`.
`boolean`	`isEmpty()` Tests if this vector has no components.

`Object`	`lastElement()` Returns the last component of the vector.
`int`	`lastIndexOf(Object elem)` Returns the index of the last occurrence of the specified object in this vector.
`int`	`lastIndexOf(Object elem, int index)` Searches backwards for the specified object, starting from the specified index, and returns an index to it.
`void`	`removeAllElements()` Removes all components from this vector and sets its size to zero.
`boolean`	`removeElement(Object obj)` Removes the first occurrence of the argument from this vector.
`void`	`removeElementAt(int index)` Deletes the component at the specified index.
`void`	`setElementAt(Object obj, int index)` Sets the component at the specified `index` of this vector to be the specified object.
`void`	`setSize(int newSize)` Sets the size of this vector.
`int`	`size()` Returns the number of components in this vector.
`String`	`toString()` Returns a string representation of this vector.
`void`	`trimToSize()` Trims the capacity of this vector to be the vector's current size.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

elementData

```
protected Object[] elementData
```

The array buffer into which the components of the vector are stored. The capacity of the vector is the length of this array buffer.

Since:

JDK1.0

elementCount

```
protected int elementCount
```

The number of valid components in the vector.

Since:

JDK1.0

capacityIncrement

```
protected int capacityIncrement
```

The amount by which the capacity of the vector is automatically incremented when its size becomes greater than its capacity. If the capacity increment is `0`, the capacity of the vector is doubled each time it needs to grow.

Since:

JDK1.0

Constructor Detail

Vector

```
public Vector(int initialCapacity,
      int capacityIncrement)
```

Constructs an empty vector with the specified initial capacity and capacity increment.

Parameters:

`initialCapacity` - the initial capacity of the vector.

`capacityIncrement` - the amount by which the capacity is increased when the vector overflows.

Throws:

`IllegalArgumentException` - if the specified initial capacity is negative

Vector

```
public Vector(int initialCapacity)
```

Constructs an empty vector with the specified initial capacity.

Parameters:

`initialCapacity` - the initial capacity of the vector.

Since:

JDK1.0

Vector

```
public Vector()
```

Constructs an empty vector.

Since:

JDK1.0

Method Detail

copyInto

```
public void copyInto(Object[] anArray)
```

Copies the components of this vector into the specified array. The array must be big enough to hold all the objects in this vector.

Parameters:

`anArray` - the array into which the components get copied.

Since:

JDK1.0

trimToSize

```
public void trimToSize()
```

Trims the capacity of this vector to be the vector's current size. An application can use this operation to minimize the storage of a vector.

Since:

JDK1.0

ensureCapacity

```
public void ensureCapacity(int minCapacity)
```

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.

Parameters:

`minCapacity` - the desired minimum capacity

Since:

JDK1.0

setSize

```
public void setSize(int newSize)
```

Sets the size of this vector. If the new size is greater than the current size, new `null` items are added to the end of the vector. If the new size is less than the current size, all components at index `newSize` and greater are discarded.

Parameters:

`newSize` - the new size of this vector.

Throws:

`ArrayIndexOutOfBoundsException` - if new size is negative.

Since:

JDK1.0

capacity

```
public int capacity()
```

Returns the current capacity of this vector.

Returns:

the current capacity of this vector.

Since:

JDK1.0

size

```
public int size()
```

Returns the number of components in this vector.

Returns:

the number of components in this vector.

Since:

JDK1.0

isEmpty

```
public boolean isEmpty()
```

Tests if this vector has no components.

Returns:

`true` if this vector has no components; `false` otherwise.

Since:

JDK1.0

elements

```
public Enumeration elements()
```

Returns an enumeration of the components of this vector.

Returns:

an enumeration of the components of this vector.

Since:

JDK1.0

See Also:

`Enumeration`

contains

```
public boolean contains(Object elem)
```

Tests if the specified object is a component in this vector.

Parameters:

`elem` - an object.

Returns:

`true` if the specified object is a component in this vector; `false` otherwise.

Since:

JDK1.0

indexOf

```
public int indexOf(Object elem)
```

Searches for the first occurrence of the given argument, testing for equality using the `equals` method.

Parameters:

`elem` - an object.

Returns:

the index of the first occurrence of the argument in this vector; returns `-1` if the object is not found.

Since:

JDK1.0

See Also:

`Object.equals(java.lang.Object)`

indexOf

```
public int indexOf(Object elem,
                   int index)
```

Searches for the first occurrence of the given argument, beginning the search at `index`, and testing for equality using the `equals` method.

Parameters:

`elem` - an object.

`index` - the index to start searching from.

Returns:

the index of the first occurrence of the object argument in this vector at position `index` or later in the vector; returns `-1` if the object is not found.

Since:

JDK1.0

See Also:

`Object.equals(java.lang.Object)`

lastIndexOf

```
public int lastIndexOf(Object elem)
```

Returns the index of the last occurrence of the specified object in this vector.

Parameters:

`elem` - the desired component.

Returns:

the index of the last occurrence of the specified object in this vector; returns `-1` if the object is not found.

Since:

JDK1.0

lastIndexOf

```
public int lastIndexOf(Object elem,
                       int index)
```

Searches backwards for the specified object, starting from the specified index, and returns an index to it.

Parameters:

`elem` - the desired component.

`index` - the index to start searching from.

Returns:

the index of the last occurrence of the specified object in this vector at position less than `index` in the vector; `-1` if the object is not found.

Throws:

`IndexOutOfBoundsException` - if `index` is greater than or equal to the current size of this vector.

Since:

JDK1.0

elementAt

```
public Object elementAt(int index)
```

Returns the component at the specified index.

Parameters:

`index` - an index into this vector.

Returns:

the component at the specified index.

Throws:

`ArrayIndexOutOfBoundsException` - if an invalid index was given.

Since:

JDK1.0

firstElement

```
public Object firstElement()
```

Returns the first component of this vector.

Returns:

the first component of this vector.

Throws:

`NoSuchElementException` - if this vector has no components.

Since:

JDK1.0

lastElement

```
public Object lastElement()
```

Returns the last component of the vector.

Returns:

the last component of the vector, i.e., the component at index `size() - 1`.

Throws:

`NoSuchElementException` - if this vector is empty.

Since:

JDK1.0

setElementAt

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

Sets the component at the specified `index` of this vector to be the specified object. The previous component at that position is discarded.

The index must be a value greater than or equal to `0` and less than the current size of the vector.

Parameters:

`obj` - what the component is to be set to.

`index` - the specified index.

Throws:

`ArrayIndexOutOfBoundsException` - if the index was invalid.

Since:

JDK1.0

See Also:

`size()`

removeElementAt

```
public void removeElementAt(int index)
```

Deletes the component at the specified index. Each component in this vector with an index greater or equal to the specified `index` is shifted downward to have an index one smaller than the value it had previously.

The index must be a value greater than or equal to `0` and less than the current size of the vector.

Parameters:

`index` - the index of the object to remove.

Throws:

`ArrayIndexOutOfBoundsException` - if the index was invalid.

Since:

JDK1.0

See Also:

`size()`

insertElementAt

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

Inserts the specified object as a component in this vector at the specified `index`. Each component in this vector with an index greater or equal to the specified `index` is shifted upward to have an index one greater than the value it had previously.

The index must be a value greater than or equal to `0` and less than or equal to the current size of the vector.

Parameters:

`obj` - the component to insert.

`index` - where to insert the new component.

Throws:

`ArrayIndexOutOfBoundsException` - if the index was invalid.

Since:

JDK1.0

See Also:

`size()`

addElement

```
public void addElement(Object obj)
```

Adds the specified component to the end of this vector, increasing its size by one. The capacity of this vector is increased if its size becomes greater than its capacity.

Parameters:

`obj` - the component to be added.

Since:

JDK1.0

removeElement

```
public boolean removeElement(Object obj)
```

Removes the first occurrence of the argument from this vector. If the object is found in this vector, each component in the vector with an index greater or equal to the object's index is shifted downward to have an index one smaller than the value it had previously.

Parameters:

`obj` - the component to be removed.

Returns:

`true` if the argument was a component of this vector; `false` otherwise.

Since:

JDK1.0

removeAllElements

```
public void removeAllElements()
```

Removes all components from this vector and sets its size to zero.

Since:

JDK1.0

toString

```
public String toString()
```

Returns a string representation of this vector.

Overrides:

`toString` in class `Object`

Returns:

a string representation of this vector.

Since:

JDK1.0