Lớp Vector trong Java

Lớp Vector trong Java triển khai một mảng động. Nó tương tự như ArrayList, nhưng với hai điểm khác biệt:

- Vector được đồng bộ.
- Vector chứa các phương thức legacy mà không là một phần của Collection Framework.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Cấu trúc dữ liệu (Data Structure) trong Java**.

Lớp Vector trong Java được chứng minh rất hữu ích nếu bạn không biết kích cỡ của mảng hoặc bạn chỉ cần một thứ mà có thể thay đổi kích cỡ thông qua vòng đời của một chương trình.

Lớp Vector trong Java hỗ trợ 4 constructor. Form đầu tiên tạo một vector mặc định, mà có kích cỡ khởi tạo là 10:

```
Vector( )
```

Form thứ hai tạo một vector mà dung lượng khởi tạo được xác định bởi size:

```
Vector(int size)
```

Form thứ ba tạo một vector mà dung lượng khởi tạo được xác định bởi size và lượng gia của nó được xác định bởi incr. Lượng gia xác định số phần tử để cấp phát mỗi khi một vector được resize:

```
Vector(int size, int incr)
```

Form thứ tư tạo một vector mà chứa các phần tử của collection c:

```
Vector(Collection c)
```

Ngoài những phương thức được kế thừa từ lớp cha, Vector định nghĩa các phương thức sau:

STT	Phương thức và Miêu tả
1	**void add(int index, Object element)**

	Chèn *element* đã xác định tại vị trí đã cho trong Vector này
2	**boolean add(Object o)** Phụ thêm phần tử đã cho vào cuối của Vector này
3	**boolean addAll(Collection c)** Phụ thêm tất cả phần tử trong Collection đã xác định tới cuối của Vector này, để mà chúng được trả về bởi Iterator của Collection đã cho đó
4	**boolean addAll(int index, Collection c)** Chèn tất cả phần tử trong Collection đã xác định vào trong Vector này tại vị trí đã cho
5	**void addElement(Object obj)** Thêm phần tử đã cho tới cuối của Vector này, tăng kích cỡ nó thêm 1
6	**int capacity()** Trả về dung lượng hiện tại của Vector này
7	**void clear()** Gỡ bỏ tất cả phần tử từ Vector này
8	**Object clone()** Trả về một mô phỏng của Vector này
9	**boolean contains(Object elem)** Kiểm tra nếu đối tượng đã cho là một phần tử trong Vector này
10	**boolean containsAll(Collection c)** Trả về true nếu Vector này chứa tất cả phần tử trong Collection đã cho

11	**void copyInto(Object[] anArray)** Sao chép các thành phần của Vector này vào trong mảng đã cho
12	**Object elementAt(int index)** Trả về phần tử tại index đã cho
13	**Enumeration elements()** Trả về một bản liệt kê các phần tử của Vector này
14	**void ensureCapacity(int minCapacity)** Tăng dung lượng của Vector này, nếu cần thiết, để bảo đảm rằng nó có thể giữ ít nhất số các phần tử được xác định bởi tham số minCapacity
15	**boolean equals(Object o)** So sánh Object đã cho với Vector này về sự cân bằng
16	**Object firstElement()** Trả về phần tử đầu tiên (tại chỉ mục 0) của Vector này
17	**Object get(int index)** Trả về phần tử tại vị trí đã cho trong Vector này
18	**int hashCode()** Trả về giá trị hash code cho Vector này
19	**int indexOf(Object elem)** Tìm kiếm sự xuất hiện đầu tiên của tham số đã cho, kiểm tra tính cân bằng bởi sử dụng phương thức equals

20	**int indexOf(Object elem, int index)** Tìm kiếm sự xuất hiện đầu tiên của tham số đã cho, bắt đầu tìm kiếm tại index, kiểm tra tính cân bằng bởi sử dụng phương thức equals
21	**void insertElementAt(Object obj, int index)** Chèn đối tượng đã cho như là một phần tử vào Vector này tại index đã cho
22	**boolean isEmpty()** Kiểm tra nếu Vector này không có phần tử
23	**Object lastElement()** Trả về phần tử cuối cùng của Vector này
24	**int lastIndexOf(Object elem)** Trả về chỉ mục của sự xuất hiện cuối cùng của đối tượng đã cho trong Vector này
25	**int lastIndexOf(Object elem, int index)** Tìm kiếm ngược về sau cho đối tượng đã cho, bắt đầu từ index đã xác định, và trả về một chỉ mục
26	**Object remove(int index)** Gỡ bỏ phần tử tại vị trí đã cho trong Vector này
27	**boolean remove(Object o)** Gỡ bỏ sự xuất hiện đầu tiên của phần tử đã cho trong Vector này. Nếu Vector này không chứa phần tử đó, nó không bị thay đổi
28	**boolean removeAll(Collection c)** Gỡ bỏ tất cả phần tử, mà chứa trong Collection đã cho, từ Vector này

29	**void removeAllElements()** Gỡ bỏ tất cả phần tử từ Vector này và thiết lập kích cỡ về 0
30	**boolean removeElement(Object obj)** Gỡ bỏ sự xuất hiện đầu tiên (chỉ mục thấp nhất) của tham số từ Vector này
31	**void removeElementAt(int index)** removeElementAt(int index)
32	**protected void removeRange(int fromIndex, int toIndex)** Gỡ bỏ từ danh sách này tất cả phần tử mà có index từ fromIndex tới toIndex
33	**boolean retainAll(Collection c)** Chỉ giữ lại phần tử, mà ở trong Collection đã cho, trong Vector này
34	**Object set(int index, Object element)** Thay thế phần tử tại vị trí đã cho trong Vector này với phần tử đã xác định
35	**void setElementAt(Object obj, int index)** Thiết lập phần tử tại index đã cho của Vector này thành đối tượng đã xác định
36	**void setSize(int newSize)** Thiết lập kích cỡ của Vector này
37	**int size()** Trả về số phần tử trong Vector này
38	**List subList(int fromIndex, int toIndex)**

	Trả về một danh sách phụ từ fromIndex tới toIndex
39	**Object[] toArray()** Trả về một mảng chứa tất cả phần tử trong Vector này theo đúng thứ tự
40	**Object[] toArray(Object[] a)** Trả về một mảng chứa tất cả phần tử trong Vector này theo đúng thứ tự; kiểu runtime của mảng trả về là mảng đã xác định
41	**String toString()** Trả về một biểu diễn chuỗi của Vector này, chứa biểu diễn chuỗi của mỗi phần tử
42	**void trimToSize()** Trim dung lượng của Vector này về kích cỡ hiện tại của vector

Ví dụ

Chương trình sau minh họa một số phương thức được hỗ trợ bởi lớp Vector trong Java:

```
import java.util.*;

public class VectorDemo {

   public static void main(String args[]) {
      // initial capacity is 3, increment is 2
      Vector v = new Vector(3, 2);
      System.out.println("Initial size: " + v.size());
      System.out.println("Initial capacity: " +
      v.capacity());
      v.addElement(new Integer(1));
      v.addElement(new Integer(2));
      v.addElement(new Integer(3));
```

```
      v.addElement(new Integer(4));
      System.out.println("Capacity after four additions: " +
          v.capacity());

      v.addElement(new Double(5.45));
      System.out.println("Current capacity: " +
      v.capacity());
      v.addElement(new Double(6.08));
      v.addElement(new Integer(7));
      System.out.println("Current capacity: " +
      v.capacity());
      v.addElement(new Float(9.4));
      v.addElement(new Integer(10));
      System.out.println("Current capacity: " +
      v.capacity());
      v.addElement(new Integer(11));
      v.addElement(new Integer(12));
      System.out.println("First element: " +
         (Integer)v.firstElement());
      System.out.println("Last element: " +
         (Integer)v.lastElement());
      if(v.contains(new Integer(3)))
         System.out.println("Vector contains 3.");
      // enumerate the elements in the vector.
      Enumeration vEnum = v.elements();
      System.out.println("\nElements in vector:");
      while(vEnum.hasMoreElements())
         System.out.print(vEnum.nextElement() + " ");
      System.out.println();
   }
}
```

Nó sẽ cho kết quả sau:

```
Initial size: 0

Initial capacity: 3

Capacity after four additions: 5

Current capacity: 5

Current capacity: 7

Current capacity: 9

First element: 1

Last element: 12

Vector contains 3.


Elements in vector:

1 2 3 4 5.45 6.08 7 9.4 10 11 12
```