CS100J Lab 05. The Vector class Fall 2003

Name ___________________ Section time _____________ Section instructor _________________

Located in package java.util, class Vector provides the ability to maintain a growable list of objects. Its purpose is to let you build a list of objects when you don't know ahead of time how many objects will be in the final list -- and this happens very often! In this lab you will gain some experience with Vectors and learn just how useful they can be.

A History Lesson:

The developers of Java knew early on that they wanted some kind of growable list, so they created class Vector and shipped it out with Java v1.0. Later, however, they wanted to generalize the idea of a list. So they created new classes that provide a more general implementation than Vector does. Rather than get rid of Vector --for "backward compatability" reasons, you can't simply throw out old stuff-- for Java v1.2 the developers added new methods to class Vector so that it would be consistent with the other, newer, classes. Many of these new methods do the same thing as the old ones.

What a Vector v contains:

Basically, a Vector v contains a list of elements, numbered 0, 1, 2, ... Function v.size() tells you how many elements are in the list. We use the following non-Java notation to refer to parts of the list. The notation helps us write things more clearly and succinctly. We refer to the elements in the list as v[0], v[1], ..., v[v.size()–1]. If we want to refer to part of the list, say elements v[h], v[h+1], ..., v[k], we write v[h..k].

Vector v also has a *capacity*, which is the number of elements for which space has been allocated in memory. When an element is to be added to v but the size is already equal to the capacity, Java allocates memory for more elements --for reasons of efficiency, the capacity is usually doubled. The capacity can also be controlled bythe programmer. However, for now, forget about the capacity. You don't have to know any more than you do about it now.

Here is a list of the old methods, the corresponding new ones, and what they do:

Old method	*New method*	*Purpose*
v.addElement(Object ob)	v.add(Object ob)	append ob to v's list
v.insertElementAt(int k, Object ob)	v.add(int k, Object ob)	change v's list to v[0..k-1], ob, v[k..]
v.elementAt(int k)	v.get(int k)	= v[k]
v.removeElement(Object ob)	v.remove(Object ob)	remove ob from the list in v (if it is there)
v.removeElementAt(int k)	v.remove(int k)	remove v[k] from v's list, changing it to v[0..k-1], v[+1..]
v.removeAllElements()	v.clear()	remove all elements from v
v.setElementAt(Object ob, int k)	set(int k, Object ob)	replace v[k] by ob

Other useful methods in the Vector class are:

v.size()	= the number of elements in v's list
v.capacity()	= the number of elements that are currently allocated for v's list --this can be different from the number of elements that are actually IN v's list!
v.indexOf(Object ob)	= i, where v[i] is ob
v.toString()	= a comma-separated list of the elements in v, enclosed in brackets

Task 1. Experimenting with Vector

Download file Lab05.java from the course website. This program will help you understand exactly what is happening when you call various methods of Vector.

Take a moment to look over the code we've provided. We have defined a Vector v that you will use throughout this lab. It is public, so you can access it from the Interactions pane of DrJava. We have also defined two constructors that will illustrate different qualities of Vector. Read the specifications so you understand what each one does. Don't worry about the stub methods yet; you'll get to them later.

Compile class Lab05 and type

```
> lab= new Lab05();
```

A window should appear at the top of your screen containing a drawing of numbered boxes. This drawing represents Vector object v in class Lab05. Note that there are 10 empty boxes numbered 0-9. The numbers are called *indices* or *indexes*. You can use them to tell Java which box you're interested in.

Note that method add has a parameter of class Object. Since all classes in Java are subclasses of Object, this means you can add ANY Java object to a Vector! You cannot, however, add primitive-type values such as **int** or **char** values. This is one area where wrapper classes are useful!

To avoid the problem of how to draw arbitrary objects in those little boxes, please add only instances of class Character. If you add an Object that is not a Character, it will be allowed because it is legal, but the program will not draw it. Instead, it will draw a red question mark.

Resize your DrJava window so that it doesn't block the drawing. Now try the following in the Interactions pane:

```
> lab.v.add(new Character('A'));
```

The Character 'A' has been added to the Vector and you can see it in box 0. Now type:

```
> lab.v.remove(new Character('A'));
```

And it's gone. Note that you passed in two different objects to methods add and remove. Vector uses method equals of each element v[i] of Vector v, and for elements of class Character, v[i].equals(ob) yields true if the character in v[i] is the same as the character in ob.

Let's set up an interesting Vector. Type in the following command:

```
> lab.initializeV();
```

Look over the drawing. Note that an object can appear multiple times in the same list ('3' and '2' both appear twice). Now try the commands on the left (in the table below) in the Interactions pane. On the right, write down what the command returned (if anything) and what happened to the Vector drawing. If you don't understand WHY certain commands do certain things, ask!

Tip 1: Use the up arrow key to get your previous command instead of repeatedly typing in "new Character..."
Tip 2: Make sure you're watching the Vector drawing when you hit Enter to execute your commands in the Interactions pane! It will be much easier to see what happened.
Tip 3: Make sure you leave off the semicolon when you make a function call-- otherwise, DrJava will not show you what the function returned.

lab.v.add(new Character('B'));	
lab.v.remove(new Character('3'));	
lab.v.remove(new Character('7'));	
lab.v.indexOf(new Character('1'))	
lab.v.indexOf(new Character('+'))	
lab.v.get(5)	
lab.v.get(12)	
lab.v.indexOf(lab.v.get(2))	What is this call doing?

lab.v.indexOf(lab.v.get(8))	Why doesn't this return 8?
lab.v.firstElement()	
lab.v.set(1, new Character('O'));	
lab.v.capacity()	
lab.v.size()	
lab.v.toString()	
lab.v.trimToSize();	
lab.v.setSize(12);	What is in the cells that have red question marks? Is the new capacity also 12?

Task 2: Writing methods to manipulate a Vector

We have written three method stubs for you to implement:

public void swap(int firstIndex, int secondIndex)	Swap the objects at v[firstIndex] and v[secondIndex]
public boolean moreThanOne(Object obj)	= there is more than one occurrence of obj in v *Hint: We did something similar in Lab 03 with Strings.*
public boolean hasExtraSpace()	= there is space allotted to v that is not being used

The API is your friend. Use it! For this lab, you can go directly to the Vector page of the API. Clicking the link will open the page in a new window.

Be sure to show your work to your TA or a consultant.