# *Programming II (420-B20-HR)*

# *Lab 10 – Vectors and ArrayLists*

Date assigned: Wednesday, March 30, 2016

Date due: **Wednesday, March 30, 2016**

**Learning Objectives**

At the end of this lab, the student will be able to:

1. use a wrapper class to use a primitive data type as an object
2. create a Vector class object instead of an array when the number of elements is not known
3. add element objects to the end of a Vector object
4. determine the number of element objects in a Vector object data structure
5. retrieve element objects from a Vector object
6. remove element objects from a Vector object
7. insert items at a particular position in a Vector object
8. create a ArrayList class object instead of an array when the number of elements is not known
9. add element objects to the end of a ArrayList object
10. determine the number of element objects in a ArrayList object data structure
11. retrieve element objects from a ArrayList object
12. remove element objects from a ArrayList object
13. insert items at a particular position in a ArrayList object

**To Be Handed In:**

1. The ***username*\_B20\_L10\_Project** folder should be zipped and uploaded to **Moodle**.

**Commands, Methods and Structures Used:**

***To declare and instantiate a Vector object:***

**Vector** *vectorName* **= new Vector();**

**Vector** *vectorName* **= new Vector(***size***);**

***To add an element object to a Vector object:***

*vectorName***.addElement(***objectname***)**;

*vectorName***.addElement(new** *classname***(***constructorArguments***))**;

***To retrieve an element object from a Vector object:***

*vectorName***.elementAt(***index***)**;

***To obtain the number of elements in a Vector object:***

*vectorName***.size()**

***To remove an element object from a Vector object:***

*vectorName***.removeElementAt(***index***);**

***To insert an element object into position i of a Vector object:***

*vectorName***.insertElementAt(***objectname***,i);**

***To replace an element object at position i of a Vector object:***

*vectorName***.set(i,** *objectname***);**

***To declare and instantiate an ArrayList object:***

***ArrayList<Base\_Type>*** *arrayListName* **= new *ArrayList<Base\_Type>* ();**

***ArrayList<Base\_Type>*** *arrayListName* **= new *ArrayList<Base\_Type>* (***size***);**

***To add an element object to the end of an ArrayList object:***

*arrayListName***.add(***objectname***)**;

***To retrieve an element object from an ArrayList object:***

*arrayListName***.get(***index***)**;

***To obtain the number of elements in an ArrayList object:***

*arrayListName***.size()**

***To remove an element object from an ArrayList object:***

*arrayListName***.remove (***index***);**

***To insert an element object into position i of an ArrayList object:***

*arrayListName***.add(i,** *objectname***);**

***To replace an element object at position i of an ArrayList object:***

*arrayListName***.set(i,** *objectname***);**

**To Start:**

1. Download and unzip the folder **B20\_L10\_Project** from **Moodle**. Rename it to ***username*\_B20\_L10\_Project**.
2. Start **Eclipse**. Open your **Labs** workspace in your **420-B20** folder.
3. Create a new Java Project called ***username*\_B20\_L10\_Project**.

# Wrapper Classes

***Purpose:*** Learn to use a wrapper class to convert a primitive data type element to an object.

***To Do:***

## Open the **WrapperClasses** class in the **b20\_l10** project. So far all it does is read something from the keyboard. We are going to look at how and when the wrapper classes can be used.

## Declare a **char** called **ch**, an **int** called **num** and a **double** called **realNum**.

## After reading in **input**, assign the first character to **ch**.

## Try the different **Character** methods in **println()** statements displaying a message for each. For example if "a cat in a hat" was entered, the output would be:

## Character.isLetterOrDigit(ch) true

Character.isLetter(ch) true

Character.isDigit(ch) false

Character.isLowerCase(ch) true

Character.toLowerCase(ch) a

Character.isUpperCase(ch) false

Character.toUpperCase(ch) A

If "4 cats in a hat" was entered, the output would be:

## Character.isLetterOrDigit(ch) true

## Character.isLetter(ch) false

## Character.isDigit(ch) true

## Character.isLowerCase(ch) false

## Character.toLowerCase(ch) 4

## Character.isUpperCase(ch) false

## Character.toUpperCase(ch) 4

## Add statements to:

### check to see if **ch** is a number.

### If it is, check to see if **input** contains a ".".

### If it does, cast **input** to a **double** and assign the result to **realNum**.

What happens? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**You cannot cast a String object (or any other type of Object) to a primitive data type.**

## Change the assignment statement to use **Double.parseDouble()** to convert the **String** to a double.

## Add an else statement to convert **input** to the integer **num** if it does not contain a ".".

# Create and Use a Vector Object

***Purpose:*** Learn to declare and use a Vector object instead of an array to maintain multiple copies of an object in memory at once.

***To Do:***

## At the moment the **VectorDemo** program reads a filename and then reads the integers from the file and prints them out. Open **VectorDemo** and run it with "*MyNumbers.txt*" as the filename. What happens? How many numbers are there?

## Run the program again using "*SomeNumbers.txt*". What happens? Why?

The size of the integerarray is 10, but there are 19 integers in the "*SomeNumbers.txt*" file. Therefore, when the 11th integer is read, the program crashes with an **ArrayIndexOutOfBoundsException**.

Instead of changing the array size every time we run the program or declaring an enormous array that will handle very large data files, we can use a **Vector** object to store our collection of integers.

## Import **java.util.Vector** and change the declaration of **collection** from an integer array to a **Vector**:

**private Vector collection ;**

## In the constructor, replace the array instantiation with a Vector instantiation using the **Vector()** constructor.

## In the **initialize()** method, use the **addElement()** method to add a new integer to the end of **collection**:

**collection.addElement(input.nextInt());**

## You can delete the **numNumbers** instance variable now.

## In the **print()** method:

### use the **size()** method instead of **numNumbers** to determine the number of elements in the **Vector**;

### use the **elementAt()** method to get the next element in the collection:

**System.out.println(collection.elementAt(i));**

## Trying running **VectorDemo** with both input files. What happens now?

## The **getLocation()** method returns the index of the integer parameter. Write the method using the following algorithm:

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| num | 1. loop i = 0 to the size of the collection  1.1 if the element at index i1 = num  1.1.1 return i  End loop  2. return -1 | the index of num (-1 if num is not in the collection) |

1 The default object type for a **Vector** is **Object** and so it can't be compared to an integer. To declare a Vector of integers, change the declaration to **private** **Vector<Integer> collection**.

## The **deleteFromCollection()** method is called to delete an existing number. It calls **getLocation()** to get the index of the number to be deleted. Add a **println()** statement at the end of the method to display the index of the number entered. Add a call to **deleteFromCollection()** to the end of the **main()** method and run the program using the following test cases to test your **getLocation()** method.

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Input Data** | | **Expected**  **output** |
|  | **filename** | **num** | **index** |
| 1. first element in collection | somenumbers.txt | 15 | 0 |
| 1. last element in collection | somenumbers.txt | 12 | 18 |
| 1. elsewhere in collection | somenumbers.txt | 258 | 10 |
| 1. not in collection | somenumbers.txt | 5000 | -1 |

## Refer to the list of **Vector** methods on the first page of this lab to determine how to delete the element at a given index and add an **if** block at the end of the **deleteFromCollection()** method to delete the element if the element is in the collection.

## The **changeCollection**() asks the user if they want to add or remove anything from the list. Change the call to **deleteFromCollection()** in the **main()** method to call **changeCollection()**. Call **print()** again after executing **changeCollection()**.

## Run the program to test your **deleteFromCollection()** method by trying to delete one or more numbers from the collection.

## The **addToCollection()** method is called to add a new number before an existing number. Refer to the list of **Vector** methods on the first page of this lab to determine how to insert an element at a given index. Add an **if** block at the end of the **addToCollection()** method to add the new number before the specified existing number.

## Run the program to test your **addToCollection()** method by trying to insert one or more numbers at different locations in the collection.

# Create and Use an ArrayList Object

***Purpose:*** Learn to declare and use an ArrayList object instead of an array to maintain multiple copies of an object in memory at once. ArrayLists and Vectors are very similar, but the industry is moving away from Vectors towards ArrayLists. You will understand both, so that you can work in either environment.

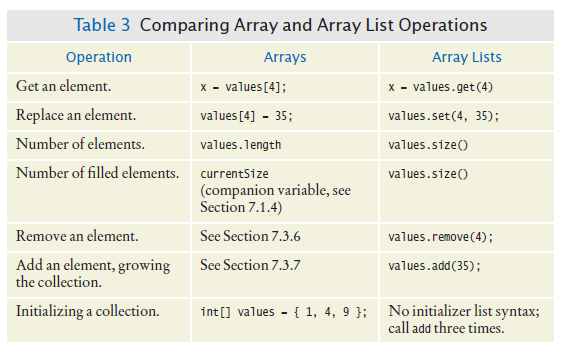
***To Do:***

## Make a copy of the **VectorDemo** class and name it **ArrayListDemo** class.

## Repeat all of the steps that you did for the **VectorDemo** class for the **ArrayListDemo** class. You should have the same functionality, but using an **ArrayList**. Refer to the list of **ArrayList** methods on the first page of this lab to determine how to create, add to, delete from, and modify an **ArrayList**.

\*Remember that ArrayLists can not be primitive types such as int and double. You will need to use the Wrapper classes such as Integer and Double instead.

Refer to your class notes to determine the methods to use. The following table will help:



## Create a new class called **ArrayListPractice**. The program should initialize an ArrayList with ten random integers and perform the following:

1. Print every element at an even index
2. Print every even element
3. Remove all odd elements from the ArrayList and print out the resulting ArrayList.
4. Replace every negative element with a 0 and print out the resulting ArrayList.

# Homework

## Complete the **Week 10 Quiz** on Moodle by April 3.