Lab 9: Building the ArrayList Class
Steven K. Andrianoff
Computer Science Department
St. Bonaventure University
Copyright, 2007

Objective:

This lab involves implementing methods of the ArrayList class.

Instructions:

1. Create a Java project in Eclipse. Add the files ArrayList.java and WordLister.java to your project. ArrayList.java is a partially completed generic ArrayList class that we developed in class. WordLister.java is a main class that uses the ArrayList class to store a list of words obtained from a text file. Add the data files catnfiddle.txt, jacknjill.txt, and pease.txt to your project. Run the program using one of the text files to be sure everything is set up correctly.

2. Add the contains method to the ArrayList class. You may need to review the API for ArrayList to see what the signature of the method should be.

Create a JUnit test class to test the ArrayList methods you write. Add two tests of the contains method, one to test the list for a word that is present and one to test for a word that is not present. Run the tests to make sure your code behaves correctly.

3. Review the API for ArrayList to understand the isEmpty method. Add the isEmpty method to your ArrayList class.

Add two more JUnit tests to your test class, one that tests isEmpty on an empty list and one that tests it on a non-empty list.

4. Review the API for ArrayList to understand the set method. Add the set method to your ArrayList class. Be sure to throw the appropriate exception when required.

Add a JUnit test that verifies that the set method works correctly. Add another JUnit test that verifies that the exception is thrown when the argument is not in the correct range.

5. Review the API for ArrayList to understand the indexOf method. Add the indexOf method to your ArrayList class.

Add two JUnit tests to test this new method, one to test the behavior on an item in the list and one to test the behavior on an item not in the list.

6. Add the toString method to the ArrayList class. Have the toString method behave in the same way as it behaves in the ArrayList class in the Java library.

Add two JUnit tests to test the toString method. One of the two tests should test an empty list.

Modify the main method to use the toString method to display the list that is built. Run the program on pease.txt, print the results, and hand in the results with your lab.
7. Add the `equals` method to the `ArrayList` class. When designing the `equals` method you need to think about what it means for two lists to be equal. A reasonable expectation is to say two lists are equal if and only if they contain the same data values in the same order. Write the `equals` method using this idea.

Note: Conceptually the `equals` method should have the same logic you used in Programming Assignment 2.

Again, add two JUnit tests to test the `equals` method, one to compare two lists that are the same and one to compare two lists that are not the same.

Modify the main method to read words from two text files into two different lists. Then display whether or not the two lists are equal. Run the program twice, once in the case where the lists are the same and once in the case where the lists are different. Print the results, and hand in the results with your lab.

Take a “snapshot” of the results of running your full set of JUnit tests and include this with your lab write-up. Be sure you have properly documented your code then print the files `ArrayList.java`, `WordLister.java`, and the JUnit test class and hand them in with your lab.

Extra Credit:

Add the `remove` method to the `ArrayList` class. The signature of the method should be:

```
public E remove(int index)
```

Add a JUnit test that verifies that the method works correctly. Also add a JUnit test that shows that the exception is thrown if the index isn’t in the correct range.

Modify the main method by adding code at the end of main to go through the list `wordList` and remove all words that start with ‘t’. Run the program on the file `catnfiddle.txt`, print the results, and hand in the results with your lab.

Hand in the modified `ArrayList.java`, `WordLister.java`, and the JUnit test class.

Hand in:

The write-up you hand in for this lab should include:
- the results requested in Steps 6 and 7.
- copies of the source code `ArrayList.java`, `WordLister.java`, and the JUnit test class requested in Step 7.

Help Policy:

Help Policy in Effect for This Assignment: **Group Project with Limited Collaboration**

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 132.

You may use the following materials produced by other students: **materials produced by members of your group.**

You may use the following materials produced by other students: **NONE.**