Laboratory 11

CS 1323, Spring 2015

# Learning Objectives

1. Declare and construct an array of String objects where the size is known in advance (perfect size array). (30 points)
2. Perform spelling checking using a standard dictionary. (20 points)
3. Use the binarySearch() method in the Arrays class. (10 points)
4. Use a for loop. (10 points)
5. Create at least three meaningful methods that pass arrays as parameters and/or return them. (20 points)

10 points will be awarded for the documentation of your program. That means using good names for variables, comments, proper and consistent indentation of code, and meaningful use of whitespace.

Section 10: When your program is completed and running, have the teaching assistants check it to get credit for the lab. If you do not complete the laboratory during the allotted time, you may submit it on Janux before Wednesday, April 8 at 11:59 p.m. Only people who attend the whole laboratory will be permitted to submit assignments on Janux.

Section 1: Submit your .java file on Janux before Wednesday, April 8 at 11:59 p.m.

# Description

Many software programs have a spelling checker. Spelling checkers work by comparing each word in a given file to an existing dictionary to see if the word is in the dictionary. If it is, it is determined to be spelled correctly. If it is not, then it is not spelled correctly. This strategy has numerous well known shortcomings, particularly that the dictionary cannot tell if the word is being used properly, just if it was somewhere in the dictionary. So if you type “here” when you meant “hear” a spelling dictionary will not be able to detect the problem.

In this lab, you will implement a spelling checker that reads from a standard dictionary file. You’ll need to find the length of the dictionary so you can use it to create your dictionary array. Then you will read the words from the dictionary into the array. The words in the dictionary are in sorted order, so this is a big advantage. The program should then read words from the console using a Scanner and print out a warning if the word is not in the spelling dictionary. Make sure you take advantage of the fact that the dictionary is sorted by using binary search.

The global dictionary file is stored in Dictionary.txt and is available on Janux. The best place to put this is in the Project main directory.

# Command Line Arguments (Optional)

This project allows you an opportunity to final use String[] args for something useful! The parameter to the main program is called a command line argument, although the rationale for this name only makes sense if you program in Java without an IDE like eclipse. If you have a full installation of Java, you can run a Java program from the Command Line. First, export your Java file from eclipse and store it on your desktop. To compile this .java file into a class file, open the command prompt, navigate to the desktop (cd directoryName will change directories), and compile your code by entering:

javac MyClass.java

If you look at your desktop, you will now see a file called MyClass.class has been added. This is the compiled version of your Java program in a language called bytecode. You run this program by entering:

java MyClass

When you run a program this way, you have the option of entering (at the command line) additional arguments—hence command line arguments. For example, you could enter:

java SpellCheck globalDictionary.txt personalDictionary.txt

Inside your program, the String array args will now have a length of 2.

args[0] will be “globalDictionary.txt”, the name of the global dictionary

args[1] will be “personalDictionary.txt”. You can use these Strings to construct your Scanner objects that read files.

Doing this in eclipse is more complicated. First go to Project -> Properties . Select the Run/Debug Setting category, and create a New Setting. Eclipse will ask you if you want a Java Applet or a Java Application. We want a Java Application, so select that and click OK.

You will see another tabbed interface appear. Select the Arguments tab and enter the file names. These will be fed in as command line arguments when your program runs.

The purpose of command line arguments is to provide another communication channel to your program. Things like resource file names are nicely handled as command line arguments.