Laboratory 6

CS 1323, Spring 2015

# Learning Objectives

1. Create a main program that calls user defined methods. (30 points; 10 points for each method)
2. Transfer existing code into methods. (30 points; 10 points for each method)
3. Transfer information to a method using arguments. (10 points)
4. Receive information from a method using a return value. (10 points)
5. Declare a local variable in a method other than the main method. (10 points)

10 points will be awarded for the documentation of your program. That means using good names for variables, proper and consistent indentation of code, comments 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 and your partner do not finish the assignment by the end of the laboratory, you should both get a copy of the code and complete the work individually. If code is submitted after the laboratory, this project is due by 11:59 p.m. on February 25.

Section 1: This project is due by 11:59 p.m. on February 25.

# Description

In this week’s lab, you will take an existing program that plays a word game and modify it to make it more organized by using methods.

This code is available on Janux, under Laboratory 5. It is distributed as a zip file. You’ll need to import that file ( File -> Import -> General -> Archive File). The dictionary.txt file should be imported into the home directory of your Laboratory 5 project. The WordJumble.java file should be imported into the src directory of your project. If you do not put these files in the proper directories the programs will not run.

Then, you will need to play the game to see how it works. This game was inspired by jumbles, where you’re given the letters in a word in the wrong order and have to guess the word. The dictionary file included contains some really tough words! You may need to cheat to practice winning.

The next thing to do is to read the code carefully. Reading someone else’s code is different from writing your own. Think about which part of the game is implemented by each segment of code.

I’ve created a worksheet that describes some methods that make sense for this program, and helps you figure out how to separate parameters, local variables, and return values. I recommend that you work through the worksheet first, creating the organization of one method at a time. Then go to the code and program that one method. I filled in the first one to help you get off to a good start.

# Files

This program uses a file—something we have not yet done in Java. Luckily, files can be read with the Scanner class. So we construct a Scanner from a File, and then use the Scanner object just like any other Scanner.

There is one annoying Java problem. Whenever files are opened and closed there is a high probability of errors. So Java requires all methods that open a File (or a Scanner that was created from a File) to announce that they could fail to the whole world. The way this is done is shown below:

**public** **static** **void** main(String[] args) **throws** FileNotFoundException

This is simply an announcement and nothing else needs to be done (other than importing the proper package for FileNotFoundException—but that has already been done).

# StringBuilders

When we are doing things like shuffling the letters in a word, the String class is a poor choice. The String class is designed for words that aren’t going to be changed. Adding and deleting characters in a word is a change. There is another class that is used to handle words that can be changed. It’s called StringBuilder. It works very much like a String, except it doesn’t have any special syntax (like being able to append using + or the special constructor).

You do not need to know anything about the StringBuilder class to be able to accomplish this project. The code that is given uses it, and it will be hidden inside a method. You should not need to modify any code that uses a StringBuilder.