CPSC250L Lab 1
Eclipse, jUnit, and Debugging

Spring 2018

1 Introduction

The CPSC250L course is similar to the CPSC150L course in that you will read the labs and complete the exercises. The main difference is that you have more room for creativity and you will have to do more research and thinking on your own.

In CPSC150, you used BlueJ as an Integrated Development Environment (IDE). This semester, you will use Eclipse, which is a professional IDE. This means that Eclipse has more powerful features, although their sheer number can be intimidating sometimes. Like BlueJ, Eclipse is free and runs on Macintosh, UNIX, Linux and Windows. It can be downloaded from `www.eclipse.org`. If you want to familiarize yourself with Eclipse, please read subsection 4.1.

Furthermore, in CPSC250L, you will use Git, a professional version control tool, to manage and submit your lab exercises. We recommend that you read the Git tutorial at `https://gitlab.pcs.cnu.edu/pcse_tutorials/bash-and-git-wiki/wikis/home` to get started with Git. Your grade for each exercise is dependent on the number of jUnit tests that your solution passes, as well as other criteria your instructor may impose (such as code style guidelines and Javadoc). The tests are run against the latest code submitted to the master branch of your personal fork of the lab project; no other branches or commits are evaluated.

If you have any questions or issues please talk to your instructor.

2 Exercises

Before beginning, ensure that you have forked and cloned the repository at `https://gitlab.pcs.cnu.edu/cpsc250-students-s2018/cpsc250l-lab01`. This repository includes the lab manual, all jUnit tests, and a Java file which you will need to debug. After cloning, do not close Git Bash/terminal! You will need it throughout the lab.

2.1 "Hello World" with Eclipse and Git

This first exercise will introduce you to Eclipse.

Exercise 1

---

1. Open Eclipse.
2. When it asks you for the workspace, accept the default choice. (Make sure it is **not** your Git directory!)
3. Close the Welcome Screen.
4. Click "File → New → Java Project". In the window that pops up, uncheck the "use default location" checkbox and point it towards your cloned Git repository. More explicit instructions are in the Git tutorial found at `https://gitlab.pcs.cnu.edu/cpsc-labs/bash-and-git-wiki/wikis/home`.
5. Create a class named `Hello World`, by right-clicking your new project and then clicking "New → Class".
6. Create a `main` method that prints `Hello World` to the console.
7. Right click `HelloWorldTest.java` under the `src` directory in the package explorer (the leftmost pane) and click "Run as → jUnit Test".

8. If all of the jUnit tests passed (as indicated by a solid green bar in the jUnit panel), then continue on with the lab. Otherwise, if something failed or the tests will not run then ensure `jUnit 4` appears in the package explorer under your lab 1 project. If you continue to encounter problems, talk to your instructor.

Exercise 1 Complete

Run:

```
git add .
git commit -m "Completed exercise 1"
git push origin master
```

2.2 Program Arguments and the `main` Method

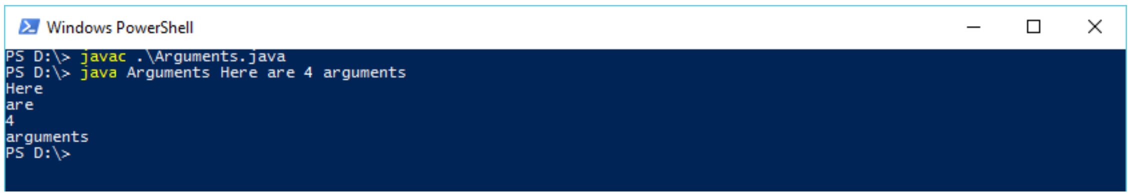
This exercise will introduce command line arguments in Eclipse.

Exercise 2

---

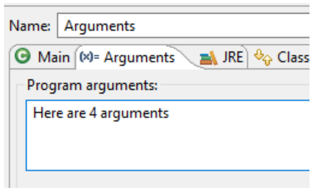
1. Create a new class called `Arguments`.
2. In this class, create a program that prints each command-line argument. You must use the `main` method for this. See an example in Figure 1.

   Hint: `args` is just an array of `String`s, and we can traverse it with a `for` loop.
3. In the package explorer, right-click `Arguments.java` and click on "Run as → Run configurations". In the window that pops up, click on arguments and then type some words in the "Program Arguments" text box. See Figure 2 for an example.



```
PS D:\> javac .\Arguments.java
PS D:\> java Arguments Here are 4 arguments
Here
are
4
arguments
PS D:\>
```

Figure 1: An example execution of `Arguments.java`



Name: Arguments

Program arguments: Here are 4 arguments

Figure 2: Example arguments added to Arguments.java

4. Once you have played with this for a little bit, run `ArgumentsTest.java`. If all tests pass, move on to the next exercise.

Exercise 2 Complete

Run:

```
git add .
git commit -m "Completed exercise 2"
git push origin master
```

2.3 Debugging

This exercise will center on debugging code.

Exercise 3

---

1. In Eclipse, open the `Debugging` class. In this file, you have been given code that is meant to find the maximum value in an array of integers.
2. Using your preferred style of debugging (`println` statements or Eclipse's debugger), fix the code so that it returns the maximum value in the array. You may assume that the array is not null and nonempty.
3. Run the test file `DebuggingTest.java`. If all tests pass, make sure you have everything pushed to Gitlab.

Exercise 3 Complete

Run:

```
git add .
git commit -m "Completed exercise 3"
git push origin master
```

---

3 Common Mistakes

Some common mistakes for this lab are as follows.

1. jUnit is not in the build path. If jUnit is not in the build path, jUnit tests cannot be run. To correct this, right click your project and click "Build path → Libraries". In the resulting window, select jUnit and click next. In the next screen, select jUnit 4 and click finish.
2. `HelloWorld` fails tests. Ensure that the class name is exactly as the lab specifies. Additionally, ensure that `main` is `public`, `static`, returns `void`, and takes a single `String[]` as an argument. Lastly, make sure it prints `Hello World` exactly.

3. Out of bounds errors. The `Arguments` and `Debugging` programs both use arrays, make sure that you do not go out of bounds. The maximum index in any array is `array.length - 1` since Java indexes arrays from 0.

Well-styled code is crucial to avoiding errors. Eclipse can autoformat your code to Java standards. Press `ctrl+shift+f` (Windows) or `cmd+shift+f` (Mac) to automatically reformat of your code. Do this as often as required to keep your code well formatted.

4 Tutorial

4.1 Eclipse

If you wish to further explore Eclipse, check out the "Explore Windows", "Explore Menus", and "Explore Views" tutorials at `http://www.tutorialspoint.com/eclipse/eclipse_explore_windows.htm`.

If you wish to learn how to use Eclipse's debugger see `http://www.tutorialspoint.com/eclipse/eclipse_debugging_program.htm`.