

# CSCE 156 – Computer Science II

## Lab 1.0 - Introduction

### Prior to Lab

In each lab there may be pre-lab activities that you are *required* to complete prior to attending lab. Failure to do so may mean that you will not be given full credit for the lab. For the first lab, there are no pre-lab activities.

### 1 Lab Objectives & Topics

Following the lab, you should be able to:

- Receive and activate your CSE account and log into the network using a Windows machine and your CSE account.
- Clone projects from GitHub using Eclipse
- Open, compile, and execute a given Java program in Eclipse.
- Write a simple program in the selected IDE, compile, and execute that program.

### Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both

driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is “in charge.” Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

## 2 Getting Started

### 2.1 Login & Consent Form

If you do not already have a CSE login, you will receive one from the CSE System Administrators who will also give you an overview of the CSE system and its policies. Be sure to login and change your temporary password. Some departmental resources that you may find useful:

- CSE Website: <http://cse.unl.edu>
- UNL Computing Policy: <http://www.unl.edu/ucomm/compuse/>
- CSE Academic Integrity Policy: <http://cse.unl.edu/academic-integrity-policy>
- CSE System FAQ: <http://cse.unl.edu/faq>
- Account Management: <https://cse-apps.unl.edu/amu/>
- CSE Undergraduate Advising Page: <http://cse.unl.edu/advising>
- CSE Student Resource Center: <http://cse.unl.edu/src>

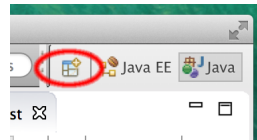
### 2.2 Checking Out Code From Github Using Eclipse

Each lab will have some starter code and other *artifacts* (data files, scripts, etc.) that will be provided for to you. The code is hosted on Github (<https://github.com>) and you must clone your own copy to work with it. You will not need to know the details of using git nor be a registered Github user to get access to the code necessary for your labs. However, you are *highly encouraged* to learn this essential tool. You may find it very useful to keep track of your own code and to share code if you work in pairs or groups.

Eclipse is the de facto industry-standard IDE for Java development. There are several other popular and emerging IDEs available and you are welcome (and encouraged) to try them out and use them. However, for this course, we will primarily use Eclipse.

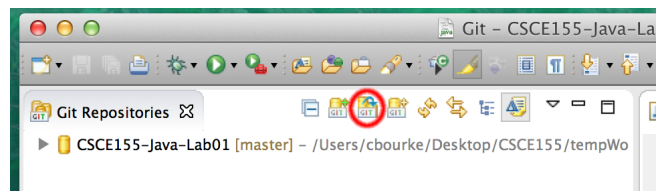
To check out the code for this lab, do the following. You may want to reference this step-by-step process in subsequent labs.

1. First we need a *Git perspective* (a context in the Eclipse User Interface that will allow us to work with Git). To open the Git perspective, click on the “Open Perspective” tab in the upper right:

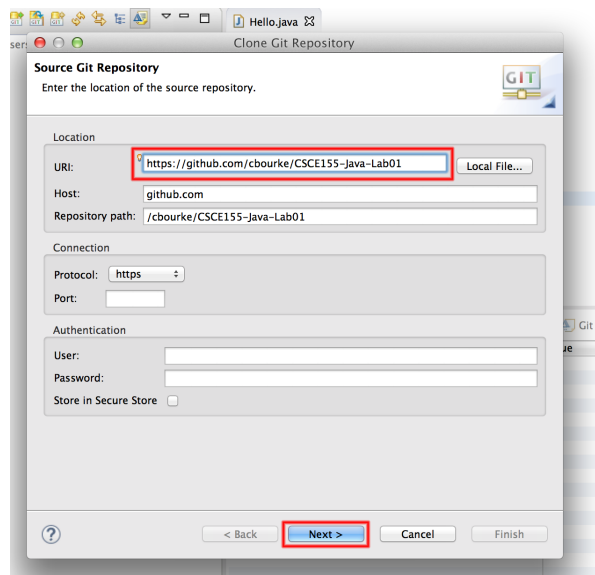


Select “Git” from the menu and click OK

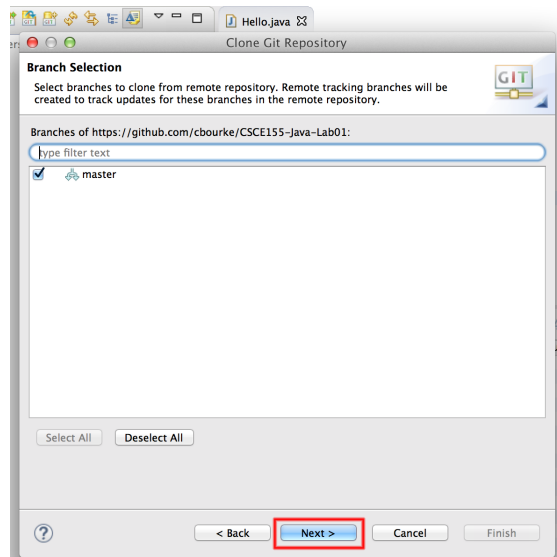
2. Click the “Clone a Git repository” in the Git Repositories navigation menu:



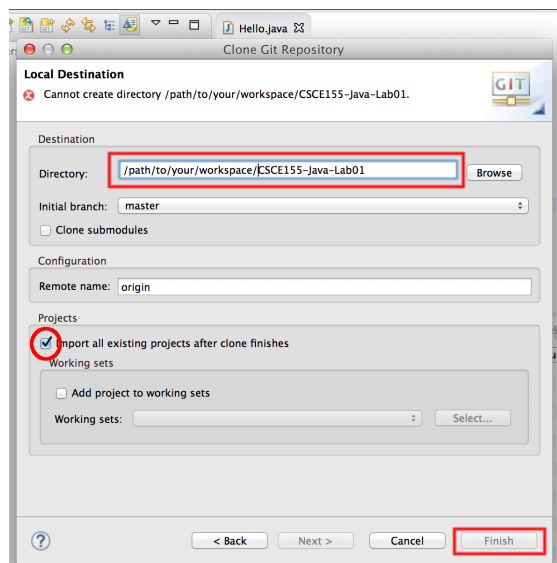
3. Copy/past or type into the URI field, the URL:  
<https://github.com/cbourke/CSCE156-Lab01>



4. Click “Next”; once Eclipse has grabbed the project, the “master” branch should be selected (checkbox); click “Next” again.



5. Select the directory where you want your project to be saved. Caution: the default option may not correspond to your default workspace. You may want to change it to your workspace, but the choice is yours. Also mark the “Import all existing projects after clone finishes” checkbox option or you will need to manually import the cloned project into Eclipse.



6. Switch back to your Java or JavaEE perspective and you can see your cloned project.

Note: this process assumes that the project you are cloning originated from an Eclipse project. Eclipse expects that files be organized in a particular way and that configuration files are present that describe how the project is setup. If the project was not an Eclipse project, you'll need to clone/setup the project in Eclipse manually.

**Note:** To complete the Java version of this lab proceed to section 3; to complete the

PHP version of this lab, proceed to section 5.

## 3 Running & Editing Programs

We will now familiarize you with Eclipse by editing a project's code.

1. Expand the `src` directory, under this we have a *package* named `unl.cse`. Java classes are organized in a hierarchy of packages. This is done for better logical organization and for package visibility reasons (to be discussed later).
2. Expand the package and you'll find several *classes*. All Java code must be contained in a class. This is in contrast to other languages that may allow global variables or allow functions to exist without an object or a class.
3. Double click on the `StatisticsDemo` class to open it in the Eclipse editor. This class contains a main method, `public static void main(String args[])`. In Java, classes are executable only if a main method is defined. Classes without a main method can be used by other classes, but they cannot be run by themselves as an entry point for the Java Virtual Machine.
4. Run the `StatisticsDemo` as follows.
  - a) Click on the “play” button as highlighted in Figure 1 (click “Proceed” if prompted).
  - b) The output for this program will appear in the “console” tab at the bottom.
  - c) Click on the console tab and enter the input as specified.

### 3.1 Completing the Statistics Program

Though the program runs, it does not output correct answers. You will need to modify these classes to complete the program.

1. Implement the `getMax()` method in the `Statistics` class. Use the `getMin()` method for directions on syntax.
2. Implement the `getSum()` method in the `Statistics` class. Use the other methods for direction on syntax.
3. Rerun the program to verify that it now works.

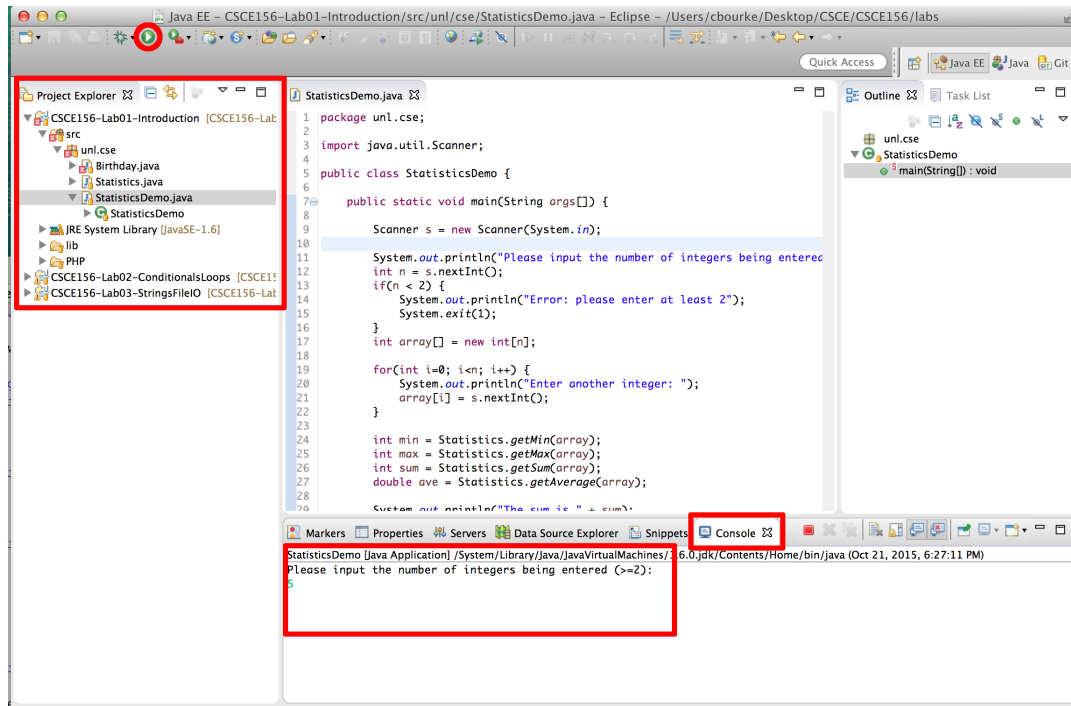


Figure 1: Eclipse Screenshot, package explorer, run button, and console input/output areas highlighted.

### 3.2 Modifying the Statistics Program

The program you've completed is interactive in that it prompts the user for input. You will now change the program to instead use command line arguments to read in the list of numbers directly from the command line.

Command line arguments are available to your main method through the `args` array of Strings. The size of this array can be obtained by using `args.length` which is an integer. Modify your code to iterate through this array and convert the arguments to integers using the following snippet of code:

```
1 for(int i=0; i<args.length; i++) {
2     array[i] = Integer.parseInt(args[i]);
3 }
```

The *command line* may not be apparent as you are using an IDE. However, it is still available to you. Instead of clicking the "Play" button to run your program, click the down arrow right next to it. Then select "Run Configurations". This brings up a dialog box with which you can run custom configurations. Click the Arguments tab and enter a space-delimited list of numbers under "Program Arguments" and click "Run".

### 3.3 Handing In and Grading Your Program

Many of your assignments will include a programming portion that will require you to hand in *soft-copy* source files for graders to compile and evaluate. To do this, you will use a web-based handin program. After handing your file(s) in, you can then grade them by using the web grader. To demonstrate, do the following.

1. Open a browser to <https://cse-apps.unl.edu/handin>
2. Login with your CSE credentials
3. Click on this course/lab 01 and hand in the `Statistics.java` and `StatisticsDemo.java` source files. You can either click the large “handin” area and select the file or you can drag-drop the files. You will be able to re-handin the same file as many times as you want up until the due date.
4. Now that the file has been handed in, you can “grade” yourself by using the webgrader; open a new tab/window and point your browser to <https://cse.unl.edu/~cse156/grade> (depending on your section, this URL may be different).
5. Fill the form with your CSE login and password, select the appropriate assignment (Lab 01) and click “Grade”

For future assignments and labs, you can compare the results of your program with the “Expected Results”. If there are problems or errors with your program(s), you should fix/debug them and repeat the handin/grading process. You can do this as many times as you like up until the due date. Some programs and assignments will run test cases and may provide expected output alongside your output. Others may have more sophisticated test cases and actually provide you a percentage of test cases passed. It is your responsibility to read, understand and *address* all of the errors and/or warnings that the grader produces.

## 4 IDE Orientation

In the next activities you’ll get more familiar with using Eclipse and the convenient functionality IDEs provide.

### 4.1 Using External Libraries

No man is an island. Good code depends on selecting and (re)using standard libraries whenever possible so that you are not continually reinventing the wheel. This activity will familiarize you with how to import and use an external Java library. Java libraries are usually packaged into JAR (**J**ava **A**Rchive) files which contain a collection of compiled class files and other resources necessary to use the library.

1. You'll notice that there are compilation errors in the `Birthday.java` file. This is because this class uses other classes that are not available in the standard JDK (Java Development Kit). It instead uses classes from the Joda-Time library; a library of useful classes and utilities for dealing with dates, times, intervals, durations, etc.
2. The JAR file, `joda-time-2.0.jar` has been included in the project in the `lib` folder. External libraries are usually kept in a hierarchy of folders like this (you can create your own folders by right-clicking the project and selecting "New" → "Folder")
3. Right-click the JAR file and select "Build Path" → "Add to Build Path." The library is now included in your project and the compiler errors should go away.

## 4.2 Cleaning Up

Though the syntax errors should now be resolved, the code isn't pretty making it difficult to read and understand. Eclipse provides a built-in code formatter functionality. Typically if you write good code to begin with it will automatically provide consistent indentation and other stylistic features. It is best practice to get in the habit of writing good, clean code automatically.

However, if you need to clean up a file in one shot you can do use the auto-formatter feature.

- On Windows: press control-alt-f to reformat the code
- On Mac: press shift-command-f to reformat the code

Another issue with the code is that it is using `lower_underscore_casing` for some of its variables. Change the variable names to the preferred `lowerCamelCasing` convention in Java. You could do this manually but a neat trick that most IDEs provide is as follows.

1. Highlight the variable name (any instance will do)
2. Right click and select `Refactor` → `Rename`
3. Type the new variable name and hit enter and it will automatically be changed for all instances!

## 4.3 Finishing The Program

Though the program should have no syntax errors, if you run it, no output will be displayed. You need to complete the program as follows.



1. For the variables, name, month, date, and year, enter your own information (your name and your birthday).
2. Add appropriate code (using `System.out.println()` ) which prints to the standard output a full line) a greeting similar to the following.

```
Greetings, NAME. Today you are XX years, XX months, and XX days old.
```

Of course, the placeholders should be replaced with variable values. In Java, variable values can be concatenated with strings using the `+` (plus) operator.

3. Add a conditional statement that, if today is the user's birthday will output `Happy Birthday` . If it is not the user's birthday, output

```
Your friends have XX shopping days until your next birthday
```

again with an appropriate variable value.

4. Complete your worksheet and demonstrate your working code to a lab instructor.

## Advanced Activity (Optional)

Explore the Joda-Time library (its API is available at: <http://joda-time.sourceforge.net/api-release/index.html>) and Java itself by implementing the following program. The program will read in (or hard code) two birthdays and compute the number of years, months, and days between them.

## 5 Running & Editing PHP

The project you checked out from GitHub contains a `PHP` folder in which there are several PHP scripts. We will now familiarize you with editing and running these scripts. These instructions assume that you are working on a lab computer. You may work on your own machine, but you will need to install a PHP interpreter or an SSH client such as Putty.

The PHP file, `statistics.php` contains a PHP program that prompts the user to enter a number of integers and computes several statistics (minimum, maximum, average, sum) and outputs the results. Most of the program's framework has been provided for you. In this activity, you will write the remainder of the program.

1. Review the `statistics.php` code in Eclipse or the text editor of your choice.
2. Login to CSE account using SSH or Putty. You can execute your script using PHP's Command Line Interface (CLI) as follows:

```
php statistics.php
```

3. Though the script runs, it does not output the correct values. To get it to work, you will need to implement the following two functions:
  - `getMax($array)` – This method should return the maximum of the elements in the array.
  - `getSum($array)` – This method should return the sum of the elements in the array.
4. Rerun your program to check that it works.

## 5.1 Modifying The Script

The program you've completed is interactive in that it prompts the user for input. You will now change the script to instead use command line arguments to read in the list of numbers directly from the command line.

PHP supports command line arguments by providing a built-in array, `$argv` which is 0-indexed. Note: the first element in this array is the script's name, so you will need to process starting from index 1. Once you have made the appropriate changes, you should be able to execute your script as follows:

```
php statistics.php 10 20 30 40
```

## 5.2 Submitting and Grading Your Program

Go to section 3.3 and follow the instructions to submit and grade your script.

# 6 PHP in a Dynamic Webpage

You will now utilize PHP in a dynamic, interactive web page. The webserver is responsible for handling requests from clients (browsers), executing the proper PHP script(s) and returning the output (HTML) to the client. This activity will have you finishing two PHP pages that reads a user's name and birthday and tells them how old they are.

1. The Apache webserver on CSE has been setup so that each user can place files into a `public_html` directory and these files will be available through a web browser. To create this directory, use the following command (in Putty):

```
mkdir public_html
```

2. Make sure your `public_html` directory world executable using this command:

```
chmod go+x public_html
```

3. Copy the `birthday.php` and `birthday_result.php` files into your `public_html` directory and make them world-readable using the command:

```
chmod o+r *.php
```

4. Open a web browser using the URL, <http://cse.unl.edu/~login/birthday.php> with `login` replaced with your login. You will need to modify the scripts to get them working properly.

The `birthday.php` page should render in your web browser without any problems. However, it is incomplete. Take these steps to get it in working order.

1. There are several months missing from the drop down menu. Include them by modifying the code that builds the `$month` array.
2. Modify the code as indicated to populate the Date drop down menu so that dates 1 thru 31 appear. Use the year drop down menu as a model.

The HyperText Transfer Protocol (HTTP) allows clients to `POST` data to another page for processing. You'll notice that your HTML form in the `birthday.php` is configured to post its results to `birthday_result.php`. Parameters posted to this page are specified by the input elements in the HTML form using the name property. In the result page, these parameters are available through the `$_POST` array. This is an associative array (indexed by strings). You can get the posted values using the syntax, `$name = $_POST['name'];` where `'name'` is replaced with the name of the form value.

1. Modify the result page to pull the values posted from the form page.
2. Also modify the script as indicated to print the appropriate message. If today is the user's birthday, print `Happy Birthday!`, otherwise print

```
Your friends have XX shopping days until your next birthday
```

where XX is the appropriate variable value. Note: strings can be concatenated with variable values in PHP by using the period operator. An example:

```
print "This is PHP, version " . $version . ", have fun!";
```

3. Answer the questions in your worksheet and show your completed programs to a lab instructor.

## Advanced Activity (Optional)

Enter an invalid birthdate into your web page and observe what happens. Is this behavior ideal? Think about some alternatives to handle invalid dates.

Most web pages utilize some JavaScript or JavaScript frameworks or libraries to make the web page dynamic. Dates are usually entered by having a calendar-like popup appear

and have the users click actual dates on the calendar. This provides a much better user experience and has the added benefit of preventing invalid dates.

Familiarize yourself with jQuery and in particular, the “Datepicker” jQuery UI plugin (see <http://jqueryui.com/demos/datepicker/#dropdown-month-year>) and integrate it into your PHP page. Note: changes will need to be made to both the input and the result page.