CSCE 156 – Computer Science II

Lab 01 - Introduction, Conditionals & Loops

Dr. Chris Bourke

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`

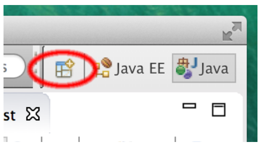
2.2 Checking Out Code From Github Using Eclipse

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.

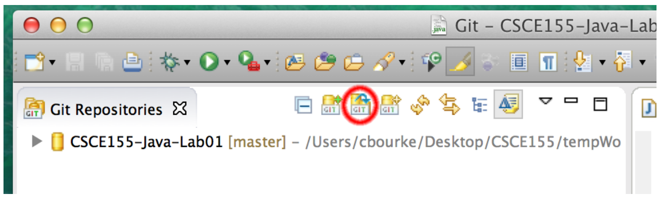
To check out the code for this lab, do the following. You may want to reference this step-by-step process in subsequent labs.

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:

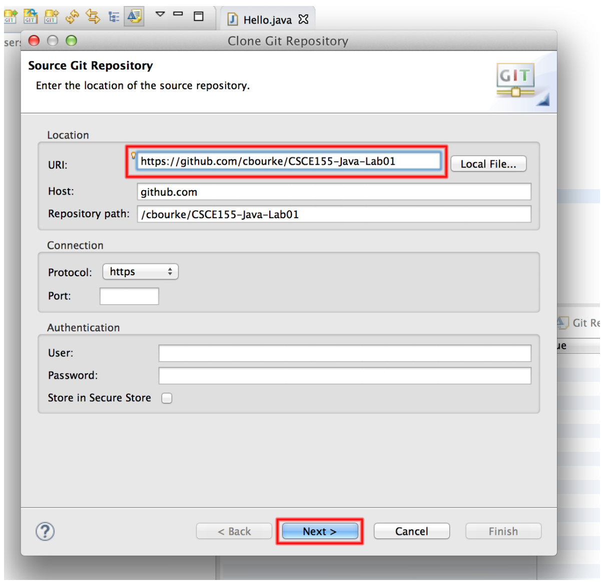
   

   Select "Git" from the menu and click OK

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

   
   Git – CSCE155-Java-Lab
   Git Repositories
   CSCE155-Java-Lab01 [master] – /Users/cbourke/Desktop/CSCE155/tempWo

3. Copy/past or type into the URI field, the URL:
   https://github.com/yi-xia/CSCE156-Lab01-IntroConditonalsLoops.git

   Clone Git Repository

   Source Git Repository
   Enter the location of the source repository.

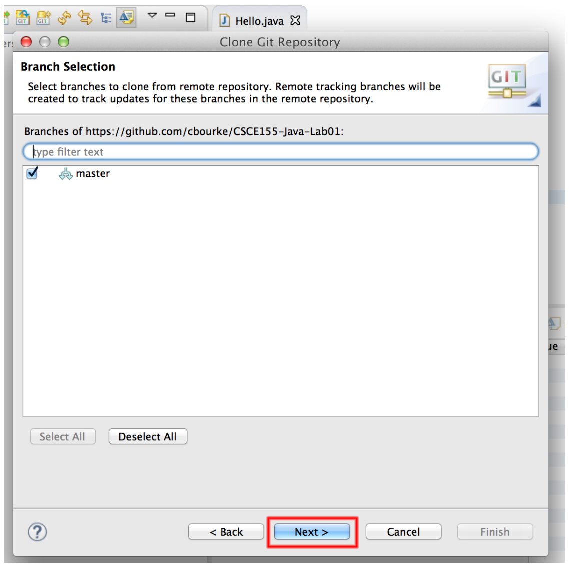
   Location
   URI: https://github.com/cbourke/CSCE155-Java-Lab01
   Host: github.com
   Repository path: /cbourke/CSCE155-Java-Lab01

   Connection
   Protocol: https
   Port:

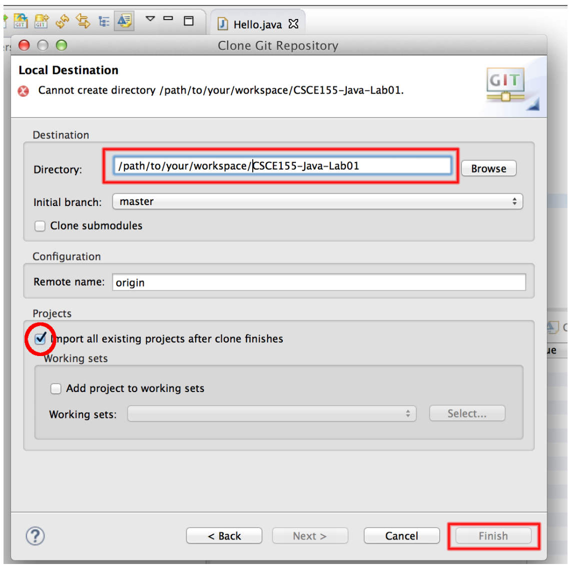
   Authentication
   User:
   Password:
   Store in Secure Store

   < Back | Next > | Cancel | Finish

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


Hello.java
Clone Git Repository
Branch Selection
Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.
Branches of https://github.com/cbourke/CSCE155-Java-Lab01:
type filter text
master
Select All
Deselect All
< Back
Next >
Cancel
Finish

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.

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

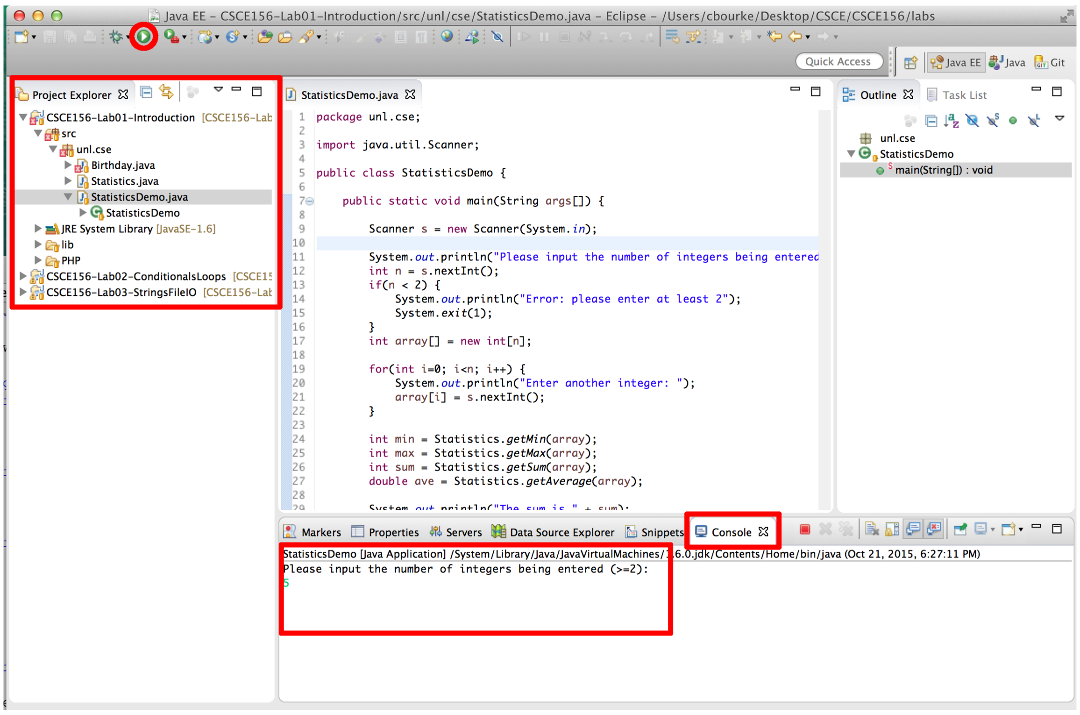
Though the program runs, it does not output correct answers. You will need to modify these classes to complete the program.

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.

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


Java EE – CSCE156-Lab01-Introduction/src/unl/cse/StatisticsDemo.java – Eclipse – /Users/cbourke/Desktop/CSCE/CSCE156/labs
Quick Access
Java EE
Java
Git
Project Explorer
CSCE156-Lab01-Introduction [CSCE156-Lab
src
unl.cse
Birthday.java
Statistics.java
StatisticsDemo.java
StatisticsDemo
JRE System Library [JavaSE-1.6]
lib
PHP
CSCE156-Lab02-ConditionalsLoops [CSCE15
CSCE156-Lab03-StringsFileIO [CSCE156-Lab
StatisticsDemo.java
Outline
Task List
unl.cse
StatisticsDemo
main(String[]) : void
Markers
Properties
Servers
Data Source Explorer
Snippets
Console

```
package unl.cse;

import java.util.Scanner;

public class StatisticsDemo {

    public static void main(String args[]) {

        Scanner s = new Scanner(System.in);

        System.out.println("Please input the number of integers being entered
        int n = s.nextInt();
        if(n < 2) {
            System.out.println("Error: please enter at least 2");
            System.exit(1);
        }
        int array[] = new int[n];

        for(int i=0; i<n; i++) {
            System.out.println("Enter another integer: ");
            array[i] = s.nextInt();
        }

        int min = Statistics.getMin(array);
        int max = Statistics.getMax(array);
        int sum = Statistics.getSum(array);
        double ave = Statistics.getAverage(array);

        System.out.println("The sum is " + sum);
```

```
StatisticsDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Oct 21, 2015, 6:27:11 PM)
Please input the number of integers being entered (>=2):
5
```

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

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:

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

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 **AR**chive) 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 shift-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 Conditionals & Loops

Java provides standard control structures for conditionals and repetition. Specifically, Java provides the usual if-then-else statements and while, for, and do-while loops. The syntax for these control structures should look familiar; some examples:

```java
if(condition1) {
  //DO SOMETHING
} else if(condition2) {
  //DO SOMETHING ELSE
} else {
  //OTHERWISE
}

for(int i=0; i<n; i++) {
  //DO SOMETHING
}

int i=0;
while(i<n) {
  //DO SOMETHING
  i++;
}
```

```java

int i=0;
do{
  //DO SOMETHING
  i++;
} while(i<n);
```

In addition, Java provides a foreach-loop, also referred to as an *enhanced for-loop*, for iterating over collections (classes that implement the `Iterable` interface) or elements in an array. This feature is mostly for convenience. The following examples illustrate this loop's usage.

```java
String arr[] = new String[10];
...
for(String s : arr) {
  System.out.println(s);
}
```

5.1 Sum of Natural Numbers

Natural numbers are the usual counting numbers; 1, 2, 3, ... In this exercise you will write several loops to compute the sum of natural numbers 1 thru n where n is read from the command line. You will also write an enhanced for-loop to iterate over an array and process data.

1. Open the `Natural.java` source file. The code to read in n has already been provided for you. An array mapping integer values 1 thru 10 to text values has also been created for you.
2. Write a for-loop and a while-loop to compute the sum of natural numbers 1 thru n and output the answer.
3. Write a foreach loop to iterate over the elements (key/value pairs) of the `zeroToTen` array. As you iterate over the elements,concatenate each string, delimited by a single space to a result string and print the result at the end of the loop. Your result should look something like the following:

   ```
   zero + one + two + three + four + five + six + seven + eight + nine + ten = 55
   ```
4. Hand in your program using the webhandin and use the webgrader to verify your program works correctly.

5.2 Child Tax Credit

When filing for federal taxes, a credit is given to tax payers with dependent children according to the following rules. The first dependent child younger than 18 is worth a $1000.00 credit. Each dependent child younger than 18 after the first child is worth a $500 tax credit each. You will complete a Java program to output a table of dependent children, how much each contributes to a tax credit, and a total child tax credit. Your table should look something like the following.

```
Child           Amount
Tommy (14)      $1000.00
Richard (12)    $500.00
Harold (21)     $0.00
Total Credit:   $1500.00
```

1. Open the `Child.java` and `ChildCredit.java` source files
2. The `Child` class has already been implemented for you. Note how the `Child` class is used; several instances of children have been created and placed into an array.
3. Write code to iterate over the array, compute the child tax credits and output a table similar to the one above. Note: to call a method on an instance of the `Child` class, use the following syntax: `kid.getAge()`
4. Answer the questions in your worksheet and demonstrate your working code to a lab instructor.

Advanced Activity (Optional)

Use the `String.format()` method to reformat the output of the Child Tax Credit program to print every piece of data in its own column.