University of Virginia, Department of Computer Science

Lab 01 - Introduction to Java and Eclipse

Dr. Mark R. Floryan

August 4, 2018

1 Pre-Lab

The goal of this prelab is to setup your coding environment in eclipse, and to write a very simple program. You will do the following:

1. Download and install eclipse, setup a new project.
2. Implement a simple program that computes the power function.
3. **FILES TO DOWNLOAD:** None
4. **FILE TO SUBMIT:** Power.java

1.1 Download and Install Eclipse

Eclipse is an Integrated Development Environment (IDE) for Java developers. An IDE allows programmers to code in an environment containing more advanced features than a text-editor. Eclipse also provides easy tools for managing projects, compiling code, incorporating outside libraries, etc. Your first step is to download Eclipse. You should be able to find it here.

When you first open Eclipse, you might (depending on your version) see a welcome / tutorial screen (see figure 1.1). If you see this screen, click on **Workbench** in the upper right corner to get to your primary work area. Your workbench, once you find it, should look like figure 1.2.

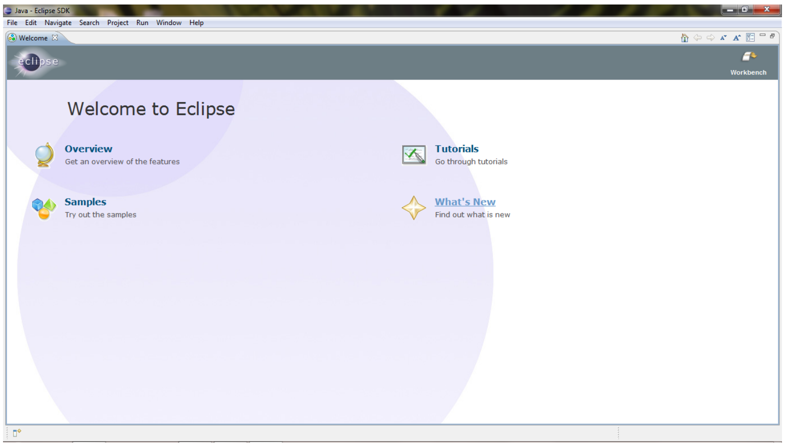
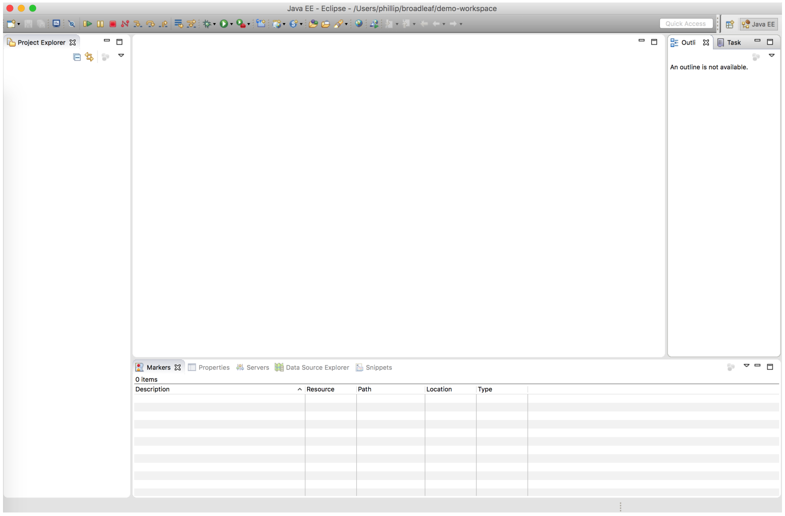


Figure 1.1: Eclipse welcome screen



An outline is not available.

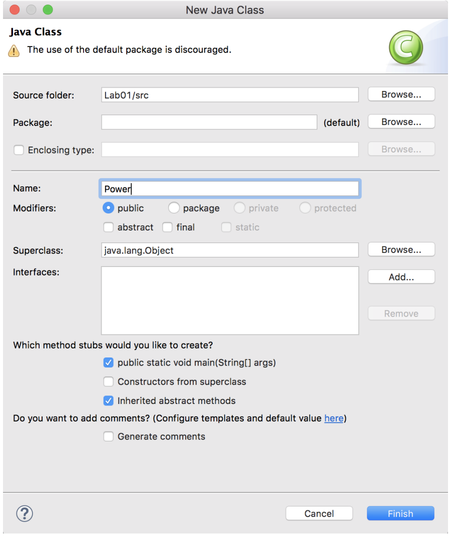
0 items

Description	Resource	Path	Location	Type

Figure 1.2: Eclipse Workspace / Workbench

To start a new project, click on **File –> New –> Java Project**. A dialog will appear. You should give your project a name, select a folder for it (optional), and click finish. You should see your project appear in the left hand dialog of the screen. If you expand the project, you should see a reference to your Java system library, as well as a folder called **src**.

Your next task is create your first Java class. Right click on your project on the left and select **New –> Class**. A dialog will appear (see figure 1.3), name your class **Power**, and select the check box that says **public static void main(String[] args)**. this option will create a main method automatically (though if you forget to do this, you can just type in the main method code manually).



New Java Class

Java Class

⚠ The use of the default package is discouraged.

Source folder: Lab01/src

Package: (default)

□ Enclosing type:

Name: Power

Modifiers: ● public ○ package ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

Figure 1.3: New Class Dialog Screen

You are ready to begin coding. For this lab, you will only need the one class, and you can write all of your code in this area. To run your program, there is a small green bug icon at the top of the screen.

1.2 POWER.JAVA

Now that your environment is setup, you will write one very small Java method as a warm-up. The method signature is:

```
public static long power(int base, int exp);
```

Your program should take in two numbers for the user. First, the base and second, the exponent. Then, your program should invoke the power method above and present the user (via the console) the value of base to the exponent power. You can assume that when we test your code, all inputs will fit inside of int types, and all correct answers will fit inside of long types. This method may be iterative or recursive, but **may not use any library functions such as Math.pow()**.

2 IN-LAB

The goal of this in-lab is to continue practicing writing some simple java code. You will do the following:

1. Get into small groups of two

2. The TAs will present you with various programming challenges. These will not be graded, but you are required to attend and to participate.

3. You may think about the challenges before lab below if you'd like, but we highly recommend that you not solve them ahead of time.

4. The TAs will give you time to solve each problem and lead you in sharing solutions with one another.

2.1 PROGRAMMING CHALLENGES

The TAs will lead you in attempting to write the following methods:

1. Given a list, print out the even numbered values only

2. Given a list, print out every other element

3. Given a list, print out only numbers that increase as you go

4. Given two lowercase strings (a-z characters only), determine if they are anagrams of one another.

5. Given the cost of a product c, and the amount a customer has paid p (note $p >= c$), determine the exact coins (using quarters, dimes, nickels, and pennies) to give to produce the correct amount of change.

   - If time remains, try to do each of these recursively instead of iteratively.

6. **Bonus Challenge**: Given the four corner points (x and y coordinates, given in clockwise order) of a polygon in two-dimensional space, and another fifth point, determine if the fifth point is inside the polygon created by the first four points.

3 Post-Lab

The goal of this post-lab is to continue practicing writing some simple java methods. You will do the following:

1. Implement five methods shown below.
2. Implement a small program to allow a user to choose one of those methods to invoke.
3. **FILES TO DOWNLOAD:** None
4. **FILE TO SUBMIT:** PostLabOne.java

3.1 PostLabOne.java

For this post-lab, you will implement five functions, as well as a small program that allows a user to select which method they'd like to test. You will start by implementing the following methods:

1. **max**: Accepts an array of integers as a parameter, and returns the largest integer among them.
2. **min**: Accepts an array of integers as a parameter, and return the lowest integer among them.
3. **average**: Accepts an array of integers as a parameter, and returns the average of the numbers as a double.
4. **median**: Accepts an array of integers as a parameter, and returns the median of the numbers. *Note that you can invoke your max and min function here to make this easier. You don't necessarily need to sort the data.*
5. **stddev**: Accepts an array of integers as a parameter, and return the standard deviation of the numbers. *Note that you can invoke your average method here.*
6. **mode**: Accepts and array of integers as a parameter, and returns the most commonly occurring value among them. *If there is more than one mode, you may return any of them.*

Once these methods are complete, you should write a program that asks the user to input a number one through five. If the user doesn't enter a valid number, the program exits. If the user enters a valid number one through five, then they are prompted to enter seven integers. Those seven integers are the given to the corresponding method above (1 = max, 2 = min, 3 = average, 4 = stddev, 5 = mode) and the answer (a single value) is printed to the console. The program then exits.

You should submit one file for this lab, **PostLabOne.java**.