# CS 225L: Lab 01 Hello, World!

# Learning outcomes:

* Use the ***javac*** command to compile a *.java* file into a *.class* file
* Use the ***java*** command to run a compiled Java (*.class*) file and generate output
* Know what a workspace is and how to change its location
* Be familiar with the various tabs, panes, and windows in Eclipse
* Create a new Java project in Eclipse
* Run the new project and observe output in Eclipse
* Know some of the different parts of a Java program
* Write a complete Java program
* Execute a program in Eclipse
* Import an existing project into Eclipse

# Pre-Lab

Welcome to CS 225L! Before we can jump into Java development, we'll need to download and install some software.

**How does Java work?**

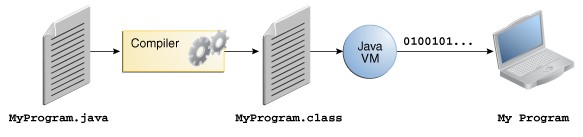
Typical programming languages, like C, work in the following way:

1. A programmer writes source code (instructions that can be read by a human).
2. A *compiler* translates the source code into *machine code* (also known as *native code*, which are instructions that can be read by a computer). 3. The computer executes the program.

The code must be compiled in a language that the executing machine can read, which means that different computers will need different translations of the source code.

Java is portable, meaning that the compiled code can be executed on many different computing platforms. This is accomplished in the following way: 1. A programmer writes Java source code and stores it in a file with a *.java* extension.

1. A Java compiler translates the source code into *Java bytecode*, which is stored in a *.class* file.
2. When the code is executed, a program called the Java Virtual Machine (JVM) interprets the bytecode, compiles it into machine code, and runs it.



An overview of how Java code is compiled and executed. Source: [docs.oracle.com](http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html)

**What am I downloading, and why?**

Java Development Kit (JDK): This contains everything you need to develop Java programs, including...

* The Java Runtime Environment (JRE), which contains: o The JVM to interpret Java bytecode (.class) files

o Standard Java libraries that are referenced in the bytecode

* Source code for Java libraries
* Compilation tools

Eclipse IDE (Integrated Development Environment): Instead of just writing code in Notepad, programs like Eclipse are designed specifically for software development. IDEs have the following advantages:

* Auto-completion of common code snippets
* Syntax highlighting (different colors and fonts for different parts of code, such as comments, variable names, [literals,](http://en.wikipedia.org/wiki/Literal_%28computer_programming%29) [java keywords,](http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html) etc.)
* Error-checking as code is written
* Easy compilation and execution of code  [Debugging](http://en.wikipedia.org/wiki/Debugging) tools
* And much, much more!

In the next couple of sections, we’ll go over how to install the JDK and Eclipse. *Note that these steps are for your personal computers. All computers on campus, including our lab computers, have Java and Eclipse already installed; however, the lab computers may need to have PATH updated for javac to work correctly.*

# Setting Up Java

## Step 0: Install Java

1. Download the most recent version of the Java JDK by clicking [here.](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)
2. Install the JDK according to the on-screen instructions. Installing to the default directories is recommended.

Once Java is installed, you may proceed to Step 1. If you are **not** a Windows user, you may proceed to Step 2.

## Step 1: Modify the PATH variable

In this step, we will open the system properties to configure an [environment variable](http://en.wikipedia.org/wiki/Environment_variable#Windows) called PATH, which will allow us to use Java from the command line. There are several ways to access the environment variables via system properties and control panel and whatnot, but the easiest way to modify PATH is directly from the command line itself.

*Opening the Command Line*

There are a few ways to open the command line, including the following:

* Press ***Windows Key + R*** to open the ***Run*** dialog, type **cmd**, then press **Enter** or click **OK**.
* Open the***Start Menu*** (or on Windows 8, the search thing), search for **Command Prompt**, then open it.

*Use the Command Line*

On the command line, type the following:

**setx path "%PATH%;**<your Java path here>**;"**

...where your Java path is the folder that contains the Java binaries. On the school computers, this path is C:\Program Files\Java\jdk1.8.0\_20\bin

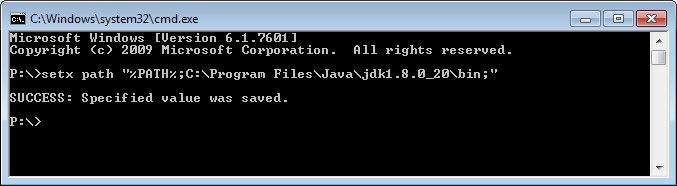
On your computer, it will either be this exact folder, or one very similar to it (Program Files (x86) vs Program Files, or a different jdk version number).

The full command looks like this:

**setx path "%PATH%;C:\Program Files\Java\jdk1.8.0\_20\bin;"**

(If you're using a school computer, you may copy-and-paste the above line directly).

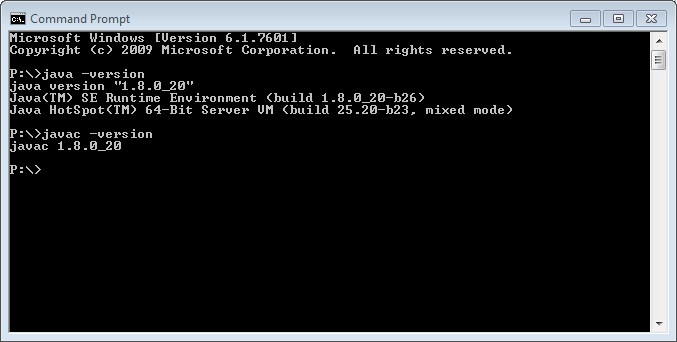
If everything went well, your command prompt should look like this:



If you see a message saying, “*WARNING: The data being saved is truncated to 1024 characters.*”, your lab instructor will help set up PATH graphically (System Properties -> Advanced system settings -> Environment Variables…).

## Step 2: Confirm Proper Configuration

1. Close and re-open command prompt (Mac and Linux users: open Terminal)
2. Type **java -version** and press Enter. This will confirm that the JRE is installed properly.
3. Type **javac -version** and press Enter. This will confirm that the JDK is installed properly.
4. If you see something like the following, the installation was successful:



# Setting Up Eclipse

1. Download the most recent version of Eclipse from [this link.](http://www.eclipse.org/downloads/) Select “Eclipse IDE for Java Developers”.

1. Install Eclipse according to the onscreen installation instructions. Installing to the default directories is recommended.

1. When you open Eclipse, you will be asked to select a workspace. The workspace is the folder that will contain all of your Java projects. If you’re using a school computer, we recommend putting your workspace on your P: drive (e.g. “P:\CS 225 workspace”).

# Introduction to the Command Line

Modern computers use a *graphical user interface* (GUI) as a means of communicating with a user. The user can click on buttons, select items from menus, see pictures, etc. By contrast, a *command-line interface* (CLI) is a text-based way of communicating with a system. CLIs are used both on the operating system level (***Command Prompt*** in Windows, ***Terminal*** in Mac/Linux), and sometimes for applications (MATLAB, GCC, Python, Maple, cowsay, etc.).

The Command Prompt/Terminal will open to a directory as indicated at the start of the line. On the lab computers, the default directory is P:\. Here are a few useful commands for exploration:

|  |  |  |
| --- | --- | --- |
| Command | Windows syntax | Mac/Linux syntax |
| Change directory | cd *C:\Full\Path\Here* | cd /full/path/here |
| Change drive | cd /D *C:* or, simply, *C:* | n/a (Unix-based file systems do not have a drive structure) |
| Go into a subdirectory | cd .\*folder-name* | cd ./*folder-name* |
| Go up one level | cd .. | cd .. |
| View directory contents | dir | ls (for **l**i**s**t)  ls –l to show details |
| Display the contents of a file | type *myfile.txt* | cat *myfile.txt* (for con**cat**enate) |
| Delete everything | del /S C:\ | rm –rf / |

# Using Java from the Command Line

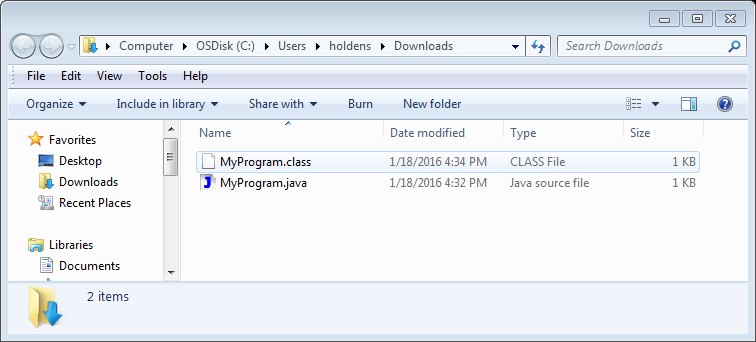
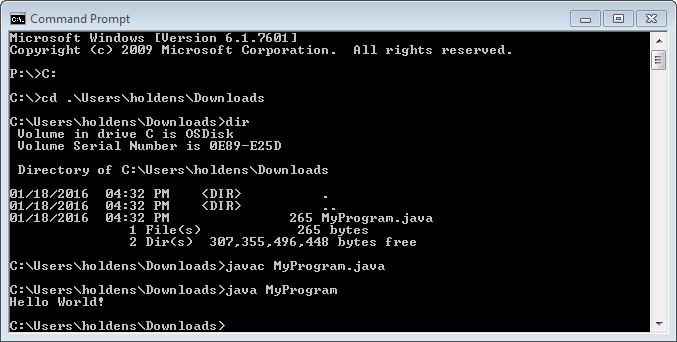
1. Download MyProgram.java and save it somewhere memorable (P: drive, Desktop, etc.).
2. Open the Command Prompt/Terminal (On Windows, search for ***Command Prompt***on the start menu).
3. Navigate to the folder to which you just downloaded the java file using cd.  Alternatively, your explorer may have an “Open Terminal here” when you right-click on a folder. On Windows, holding the *shift* key while right-clicking will add an “Open command window here” option.
4. Recall that source code must be compiled and then executed. To compile code from the command line, use the ***javac*** (***java*** ***c***ompile) command, followed by the name of the file (including the *.java* extension). What new file appears in your directory?

 If you are using a school computer and it doesn't recognize the command, we’ll need to add the javac location to the PATH environment variable. We can do that using the following command in the Command Prompt: **setx path "%PATH%;C:\Program Files\Java\jdk1.8.0\_20\bin;**

After the command is successfully executed, close and re-open command prompt. javac should now work.

1. Finally, to execute our compiled code, use the ***java*** command, followed by the name of the file without the *.java* extension.

If everything went well, your results should look similar to the pictures below.



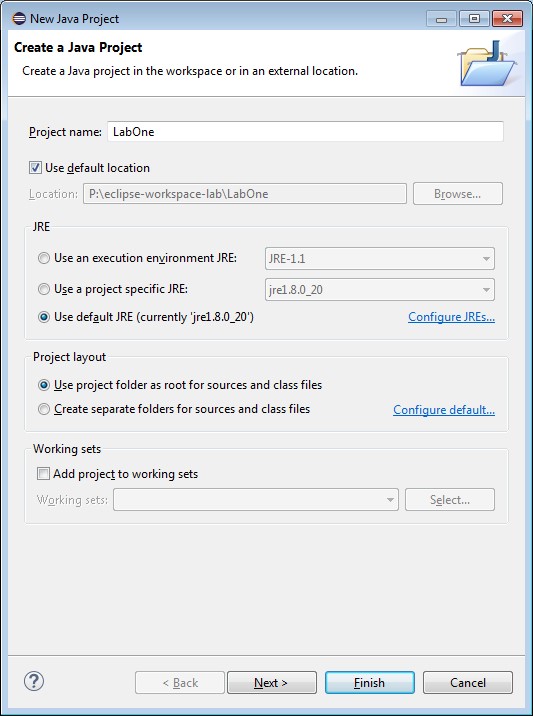
|  |
| --- |
| Please **stop** at this point and inform the instructor that you have completed the previous steps. |

At this point in the lab, your lab instructors will now give a brief introduction to Eclipse.

# Eclipse – Creating New Projects

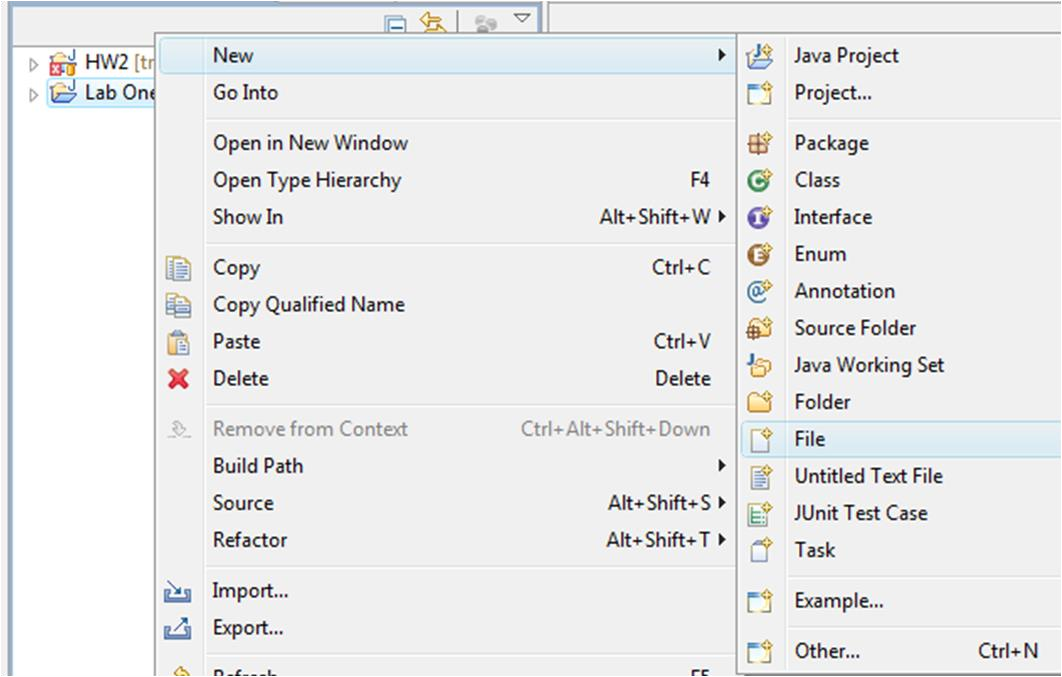
1. Open Eclipse and make sure your workspace is set in a location that you'd like it (your P: drive, a flash drive, or a location on your personal computer).
2. Create a new Java project (File -> New -> Project). Be sure to choose "Java Project".
3. Be sure to set the project variables as seen below\* (i.e. *Use default JRE*; *Use project folder as root…*).

\*I mean, you don't *have* to, and there are cases in which the other options are better suited for your project, but not for any particular reason that you’ll encounter during this course.



Be sure to use these options for all new Java projects.

1. Choose Finish.
2. Add a new file to the project by right click on the project -> New -> File (as seen below). Be sure add ".java" to the name the file ("**LabOne.java**"). Create a new file in the project.



Create a new file in the project.

The editor should open up the new file to be edited. If not, please ask your lab instructor for assistance. Otherwise, continue to the next part of the lab.

# Eclipse – My First Program

Now that we have our LabOne.java file open in the editor, let's give it some content!

First write a *block comment* to declare the Java file and the author.

NOTE: comments are not executable lines.

/\*

* LabOne.java
* Written By: Alex King
* CS 225 Spring 2010

\*/

Then declare the file as a Java class by writing the following code.

public class LabOne {

NOTE: The class name MUST MATCH EXACTLY the file name without the '.java'. Also be sure to add the opening brace '{'

Next, use a one line comment to mark the beginning of the class variables section.

//class variables

Now declare a class level variable.

public static int year;

Next, declare the 'main method' of the program, i.e. where to start executing the program. This is an important line.

public static void main(String[] args) {

Each project will contain ONLY ONE main method. The main method is where the program begins executing when run. The syntax for the main method is ALWAYS as seen above!

Now add content to the main method (i.e. lines to be executed).

String firstName="Bob"; String lastName="Loblaw"; year=2016;

System.out.println("Hello My Name is: ");

System.out.println(firstName +" " + lastName);

System.out.println("This is my first program of Spring " + year);

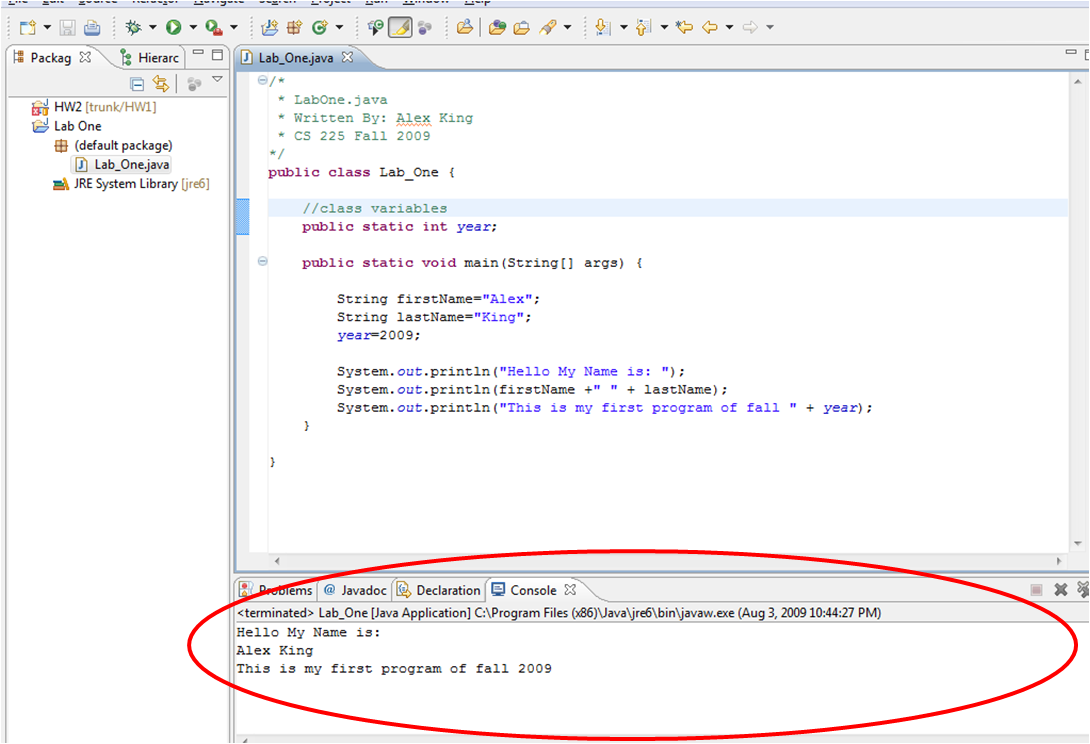
*System.out.println(…)* will print the content in the parenthesis to the output screen. Be sure to always end each executable line of code with a semicolon.

Finally, if you haven't done so already, add a couple of closing braces '}' to complete the main method and class declaration.

This file is now ready to run. Run the file using one of the following methods:

* Right Click on the file -> Run As -> Java Application
* Right Click in the editor -> Run As -> Java Application
* Run -> Run As -> Java Application

Once you run the application you should see the output console display the printed messages at the bottom of the Eclipse screen.



The Eclipse console will display any output from an executed Java program.

|  |
| --- |
| Please **stop** at this point and inform the instructor that you have completed the previous steps. |

.

**Reading Console Input**

We can add a little more sophistication to our program by having it ask for our name at the command line. We’ll use a Scanner object to read from the console (*System.in*), and retrieve the next line of text using the *nextLine()* method. Note that the Scanner can ask for other types of data (e.g.

*nextDouble()* for double values). Our class would then look something like this:

// Reference to the Scanner class in the java.util package **import** java.util.Scanner;

**public** **class** LabOneWithInput {

**public** **static** **void** main(String[] args) {

// Create a scanner for command line input

Scanner in = **new** Scanner(System.***in***);

// Prompt for, and store, the name

System.***out***.println("What is your name?");

String name = in.nextLine();

// Print the name back

System.***out***.println("Hello, " + name + "!\n");

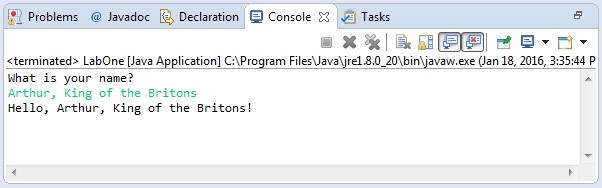
// Close the scanner

in.close();

}

}

And the console would look like this:



# Eclipse – Importing Existing Projects

1. Download the provided *.zip* file and save it somewhere memorable.
2. Outside of Eclipse, navigate to, and extract (optional), the *.zip* file.
3. Inside of Eclipse, create a new Java project just as you did before.
4. Right-Click the new project and choose **'Import...'**.

**5a.** Choose **'General' -> 'File System'** if you extracted the contents of the .zip into another folder, or...

**5b.** Choose **'General' -> 'Archive File'** to import it directly from the .zip file.

**6a.** **'Browse'** to the folder **into which you extracted the files**, or...

**6b.** **'Browse'** and **select the LabOne.zip file**, and then click **'Open'**.

1. Select the checkboxes next to all of the *.java* files you want and choose **Finish**.
2. Open the files and briefly examine the code.
3. Run the '**CreateLineArt.java**' file the same way you ran your first program.
4. Observe the output and experiment with the program.

|  |
| --- |
| Please **stop** at this point and show the instructor the output. |

# Post-Lab:

Download the lab quiz from Canvas, complete, and submit via Canvas.