

# Lab 0: “Hello, World!” and Beyond

## Welcome TO COMP 1510 PROGRAMMING METHODS

Welcome to your first COMP 1510 lab.

The goal of this lab is to introduce you to the tools you will use in COMP 1510 (and possibly COMP 2526).

The tools described should already be installed on the computers in the BCIT labs.

The first 3 sections of this lab show you how to set up your laptop and/or home computer. The rest of the lab introduces you to the Eclipse development environment and the command line.

If you have any trouble setting up your work environment, contact one of your instructors as soon as possible so we can help you.

## What will you DO in this lab?

In this lab, you will:

1. **(Windows and macOS)** Install the Java Software Development Kit on your personal computer.
2. **(Windows only)** Set environment variables to add Java to the Windows Path.
3. **(Everyone)** Install the following software:
  - a. Eclipse IDE (Integrated Development Environment)
  - b. e(fx)clipse plugin for Eclipse
  - c. Checkstyle plugin for Eclipse
  - d. SceneBuilder
4. **(Everyone)** Use Eclipse to:
  - a. create a Java project
  - b. compile a Java project
  - c. execute a Java program.
5. **(Everyone)** Use the command line to:
  - a. navigate to your source code
  - b. compile and execute your source code.

There is a helpful Table of Contents on the next page of this document. This is a long lab. Take your time. Read every page. Don't skip any steps. And have fun!

## Table of Contents

<b>Welcome TO COMP 1510 PROGRAMMING METHODS</b>	<b>1</b>
<b>What will you DO in this lab?</b>	<b>1</b>
<b>1. INSTALL JAVA</b>	<b>3</b>
<b>2. WINDOWS ONLY: Define an Environment Variable for Java</b>	<b>4</b>
<b>3. WINDOWS ONLY: add JAVA_HOME to the Windows Path</b>	<b>9</b>
<i>What did we just do, and how can we see if we did it right?</i>	<i>11</i>
<b>4. EVERYONE: Install the Eclipse IDE</b>	<b>13</b>
<b>5. WINDOWS AND MacOS: Launch Eclipse</b>	<b>13</b>
<b>6. WINDOWS AND MacOS: Install the e(fx)eclipse Plugin</b>	<b>15</b>
<b>7. WINDOWS AND MacOS: Install the Checkstyle Plugin</b>	<b>16</b>
<b>8. WINDOWS AND MacOS: Install SceneBuilder</b>	<b>17</b>
<b>9. Configure Eclipse</b>	<b>18</b>
<b>10. Create an Eclipse Project</b>	<b>20</b>
<b>11. Add a Java Source File to the Project</b>	<b>21</b>
<b>12. Compile and Execute a Java Project</b>	<b>24</b>
<b>13. Create a Project Archive (JAR) File</b>	<b>25</b>
<b>14. Run Examples from the Text (Import a Project)</b>	<b>27</b>
<b>15. Configure Checkstyle</b>	<b>28</b>
<i>Working With Checkstyle Violations</i>	<i>30</i>
<b>16. Javadocs</b>	<b>31</b>
<b>17. Working from the Windows Command Line or macOS Terminal</b>	<b>33</b>
<i>Command Line Commands: CD and DIR (Windows) / LS (macOS)</i>	<i>34</i>
<i>Compiling from the Command Line</i>	<i>36</i>
<i>Additional Things to Know about the Command Prompt</i>	<i>37</i>
<i>Redirection</i>	<i>38</i>
<b>18. You Have Completed Lab 0</b>	<b>38</b>

## 1. INSTALL JAVA

To compile and run Java programs, we need to install a Java SE Development Kit (SDK). The JDK includes a Java “runtime environment” and tools for developing, debugging, and monitoring Java applications:

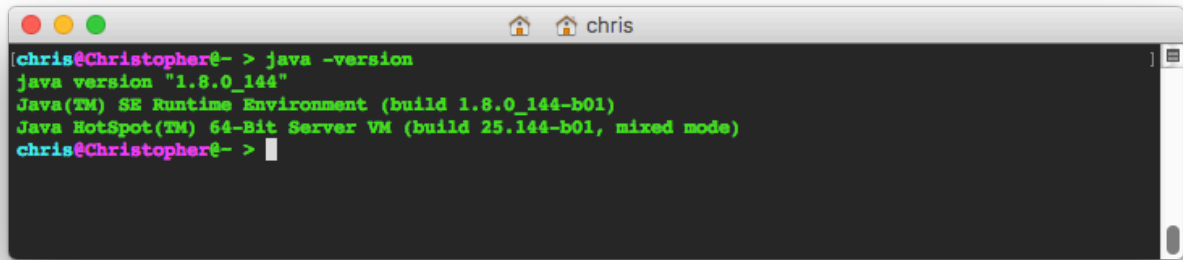
1. Open a browser window and navigate to <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
2. We want to install the most up to date version of the Java SE Development. At the time this document was written, the current version is 8u144 (this means Java version 8 update 144) but updates are frequent so you may see 8u145 or 8u150 or beyond!

Java SE Development Kit 8u144		
You must accept the <a href="#">Oracle Binary Code License Agreement for Java SE</a> to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.89 MB	<a href="#">jdk-8u144-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.83 MB	<a href="#">jdk-8u144-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	164.65 MB	<a href="#">jdk-8u144-linux-i586.rpm</a>
Linux x86	179.44 MB	<a href="#">jdk-8u144-linux-i586.tar.gz</a>
Linux x64	162.1 MB	<a href="#">jdk-8u144-linux-x64.rpm</a>
Linux x64	176.92 MB	<a href="#">jdk-8u144-linux-x64.tar.gz</a>
Mac OS X	226.6 MB	<a href="#">jdk-8u144-macosx-x64.dmg</a>
Solaris SPARC 64-bit	139.87 MB	<a href="#">jdk-8u144-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.18 MB	<a href="#">jdk-8u144-solaris-sparcv9.tar.gz</a>
Solaris x64	140.51 MB	<a href="#">jdk-8u144-solaris-x64.tar.Z</a>
Solaris x64	96.99 MB	<a href="#">jdk-8u144-solaris-x64.tar.gz</a>
Windows x86	190.94 MB	<a href="#">jdk-8u144-windows-i586.exe</a>
Windows x64	197.78 MB	<a href="#">jdk-8u144-windows-x64.exe</a>

Figure 1: The Java SE SDK download page should contain a section that looks like this.

3. Accept the license agreement for the latest version of Java.
4. Click the download link for your operating system (Windows users should download the 64 bit version).
5. Download the JDK installation file. It’s okay to save it in your Downloads folder. When we are done installing Java, we will delete this file.
6. Double click the JDK installation file to execute the JDK installer. You may accept the default installation location for the SDK and the JRE (Java runtime environment).
7. Wait until the installation finishes. This may take a few minutes, especially on a Windows computer.
8. **Windows and macOS:** You can delete the JDK installation file now. You don’t need it anymore.
9. **macOS only:** You’re done installing Java. When the installer finishes, you should verify your installation. Open a Terminal window and enter the command `java -version`. The output should look like the following. What happens if you enter the command

java, or the command `javac`? After you verify your installation, Mac users may skip ahead to section 4.



```
chris@Christopher ~ % java -version
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)
chris@Christopher ~ %
```

Figure 2: macOS Terminal: enter `java -version` and see that Java is installed

10. **Windows only:** You still need to define an environment variable for the folder that contains the Java compiler, and add it to the path. Let's do that next.

## 2. WINDOWS ONLY: Define an Environment Variable for Java

**Remember: this step is only for Windows users. macOS users may skip ahead to step 4.**

After installing the JDK, we can execute Java programs. In order for us to compile Java programs in Windows, we need to add the Java compiler to the Windows **Path**.

The cleanest way to do this is to create an environment variable for the Java installation folder, and then add the bin folder it contains to the Path.

These instructions are for Windows 10 x64, but can be adapted to other versions of Windows

1. Open the Control Panel and Choose System and Security > System > Advanced system settings > Environment Variables:

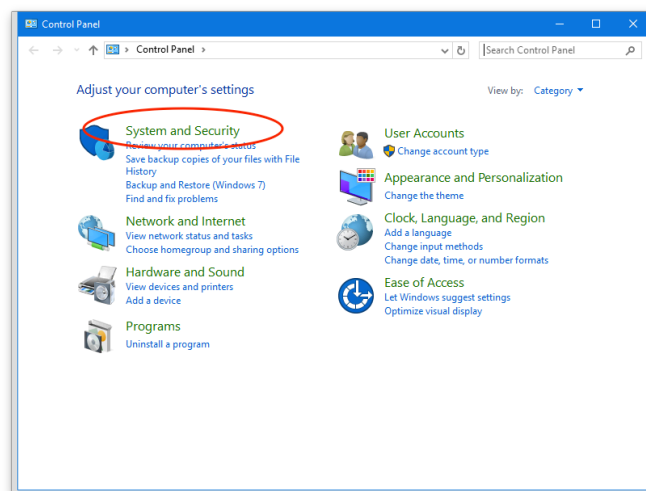


Figure 3: Control Panel

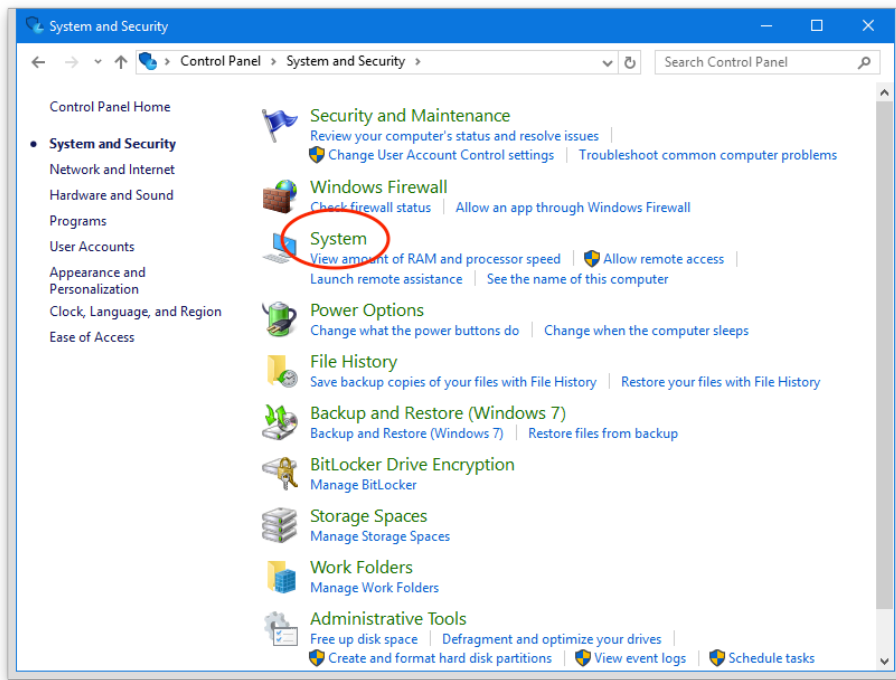


Figure 4: Control Panel/System and Security

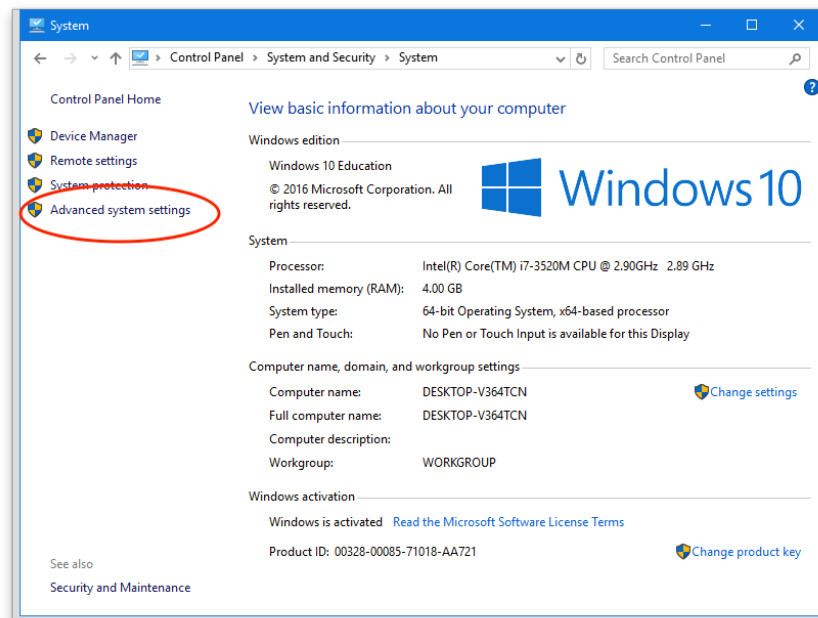


Figure 5: Control Panel/System and Security/System

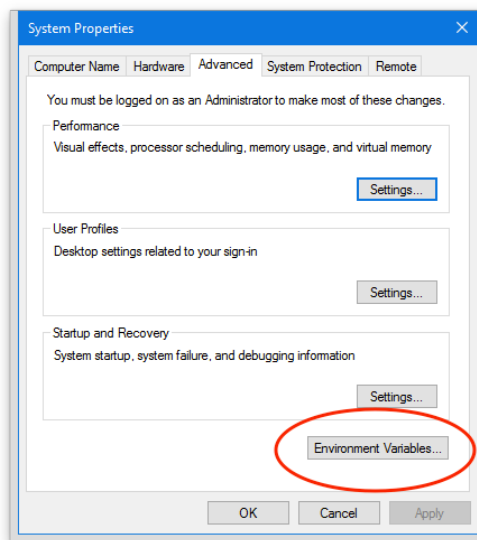


Figure 6: Control Panel/System and Security/System/Advanced system settings

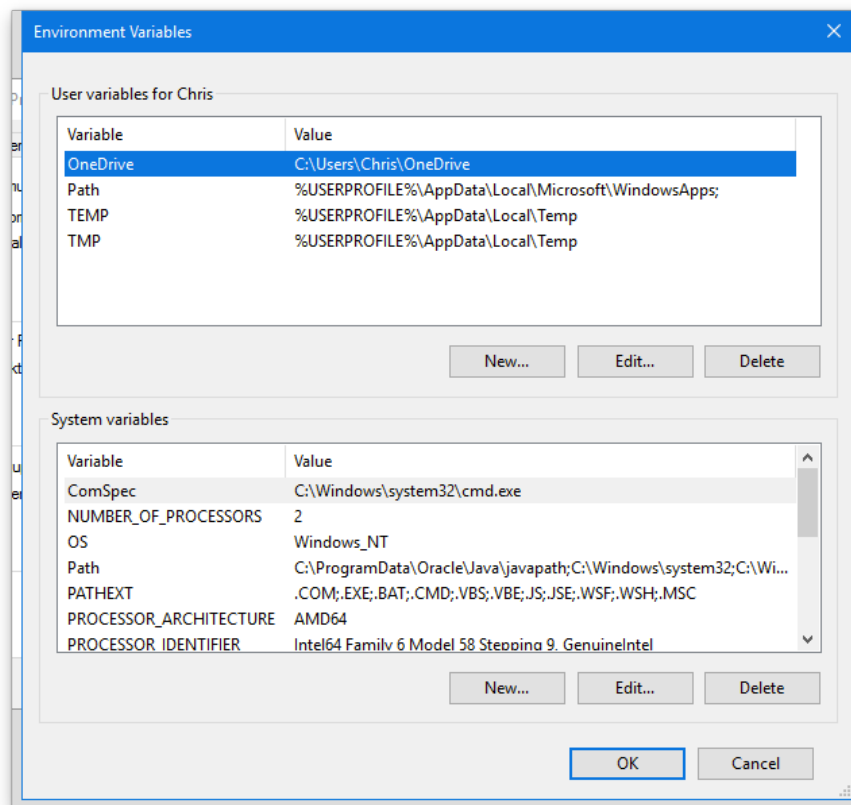


Figure 7: Windows Environment Variables.

2. Note that every variable has a name and a value.
3. There are two types of Windows environment variables:
  - a. User variables for you, the current user of the computer
  - b. System variables for the system, i.e., all of the users on the computer.
 In the BCIT labs you only have permission to change the User variables. On your personal computer, you probably want to change the User variables.
4. Before you proceed, check your User and System variables. If you see a variable named JAVA\_HOME, check to make sure that the variable value is the location of the correct installation folder, i.e., jdk1.8.0\_144.

5. If there is no variable that defines JAVA\_HOME, click the New button under User variables:

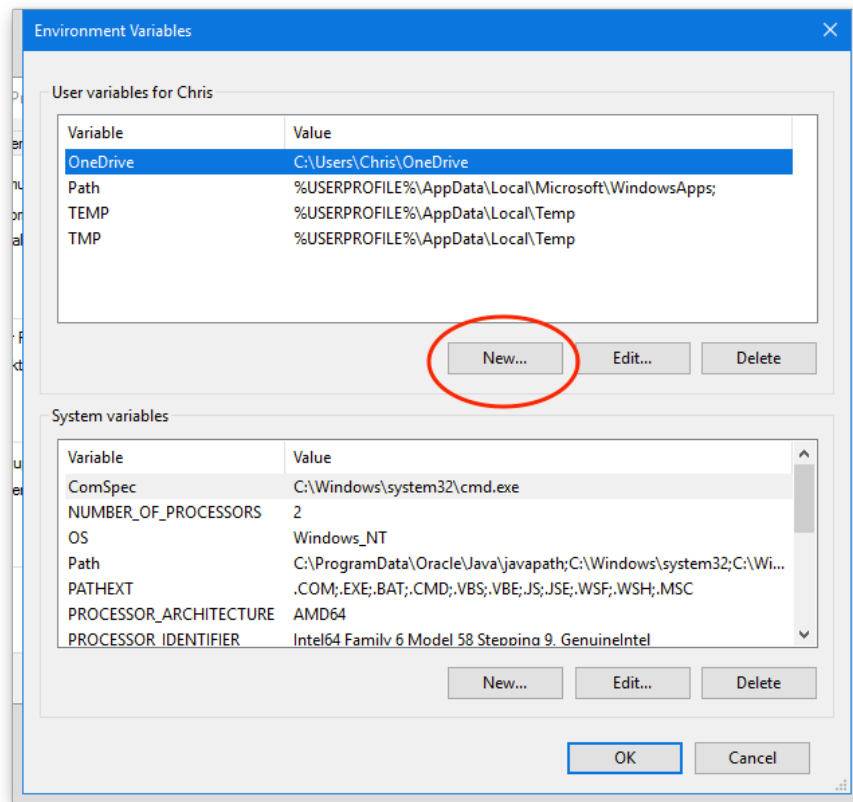


Figure 8: Add a new user variable by selecting New...

6. Create a new environment variable called JAVA\_HOME, and choose Browse Directory...

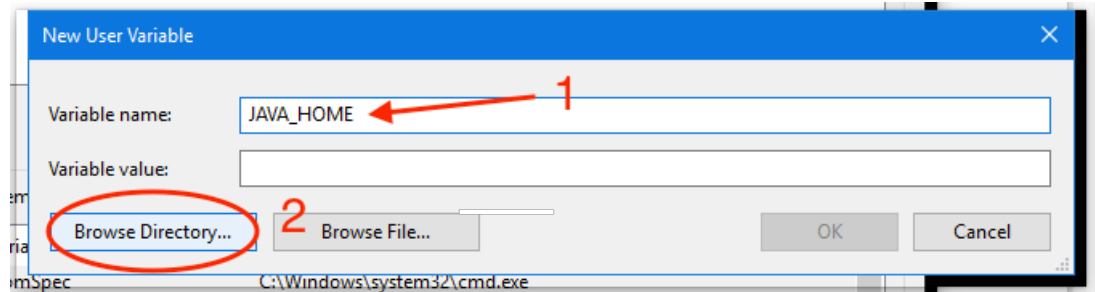


Figure 9: Create a new system variable called JAVA\_HOME

7. Navigate to the location of the jdk1.8.0\_144 installation folder (mine, for example, was installed here):

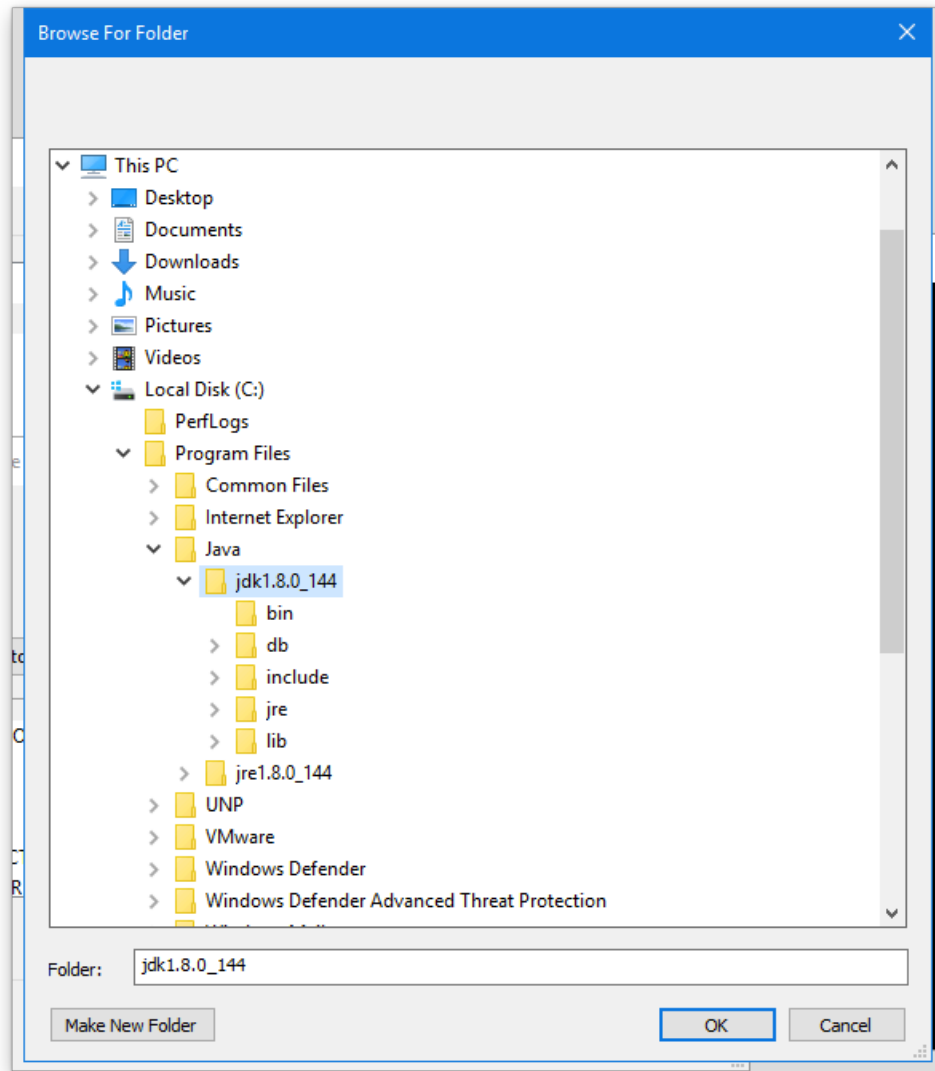


Figure 10: Select the JDK installation folder

8. Click OK. You should return to the New User Variable window, which looks like this:

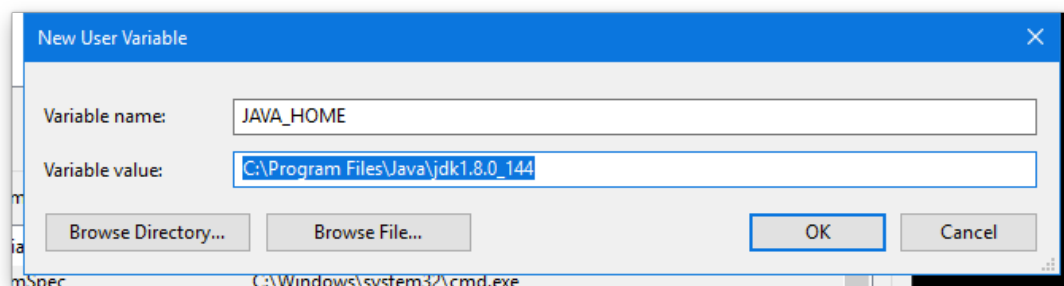


Figure 11: New user variable and value

9. Click OK to create the new user environment variable.
10. Your User variables should contain a new Variable called JAVA\_HOME with the path to the correct installation folder as its Value:



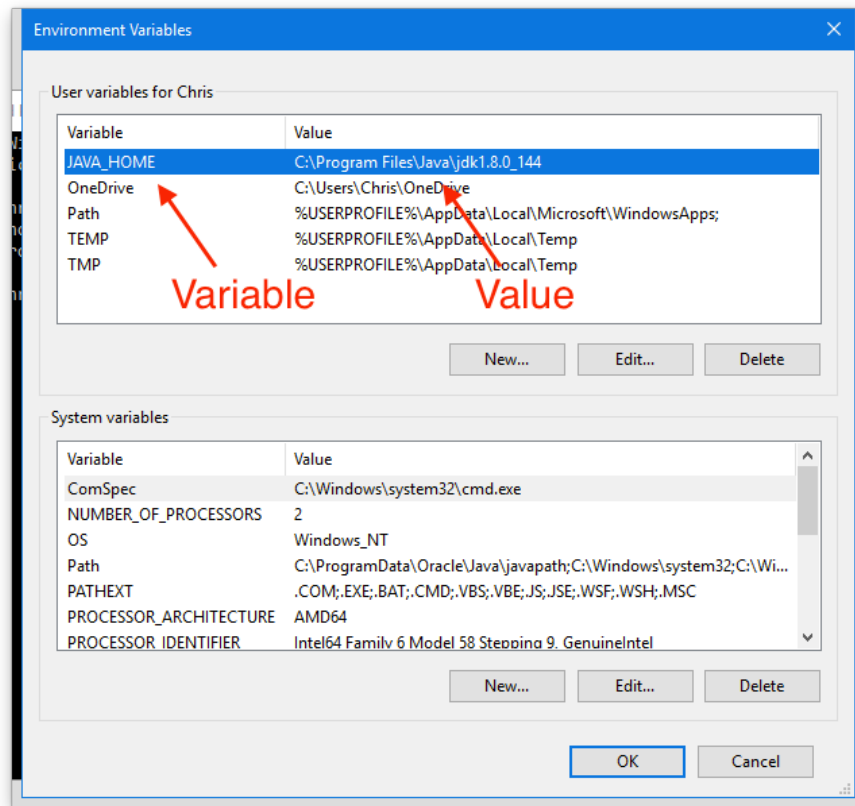


Figure 12: JAVA\_HOME created!

### 3. WINDOWS ONLY: add JAVA\_HOME to the Windows Path

**Remember: this step is only for Windows users. macOS users may skip ahead to step 4.**

Once we have defined an environmental variable, we can refer to its value by enclosing it in % characters. For example, after the preceding definition, %JAVA\_HOME%\bin is equal to C:\Program Files\Java\jdk1.8.0\_144\bin.

We now want to change the Windows Path environment variable so that it points to the directory that contains javac (the Java compiler) and the other command line tools that were installed with the JDK. We will do this by adding the %JAVA\_HOME%\bin folder to the Windows Path environment variable:

1. The first rule of editing the Path: **do not delete anything already in it**. If there are existing values in the Path, do not erase them. We will prepend (add) our environment variable to the beginning of the Path.
2. Select the Path variable in the User variables, and click Edit:

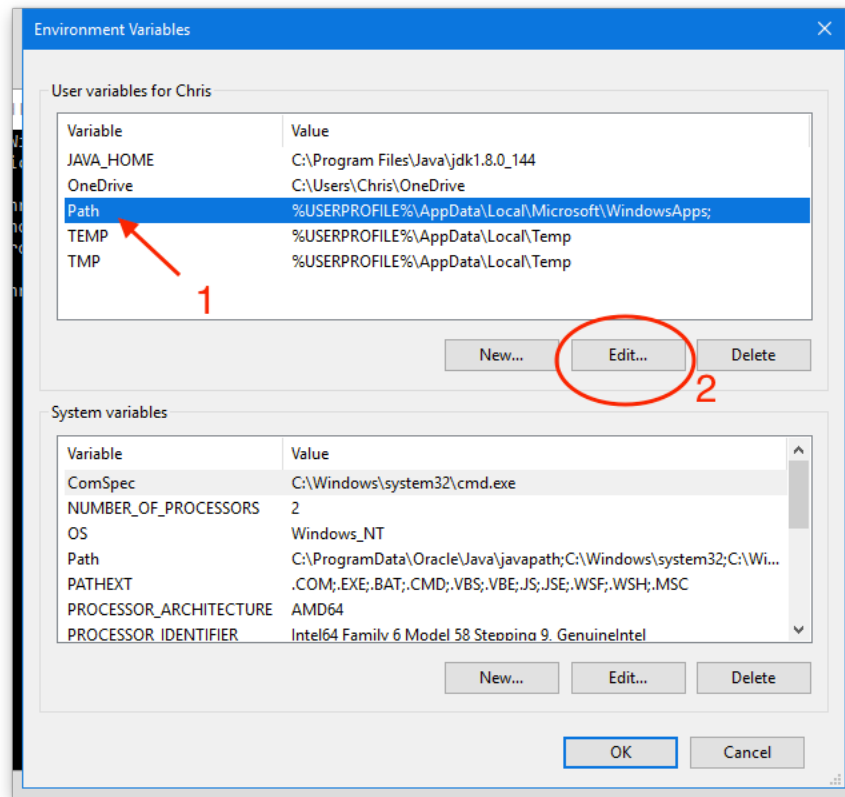


Figure 13: Select the User Variable Path and then click Edit...

3. Move your cursor to the beginning of the Path value. Do not delete any of it.
4. Now add the directory where the javac command is (hint: it is in the bin directory in the JAVA\_HOME location). If there is already something in the Path, we add the bin directory to the beginning of the Path and separate it from what is already there with a semi-colon (;). Note that a semi-colon ; is used to separate the directory names:

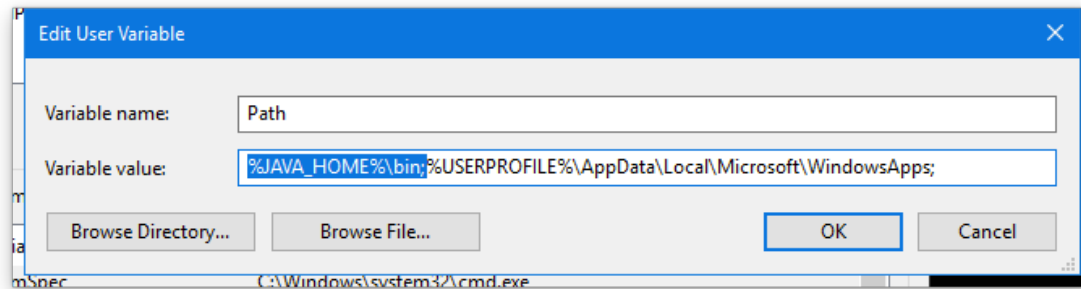


Figure 14: Prepend %JAVA\_HOME%\bin to the Path User Variable

5. Verify that the Path in your User variables now look like this:

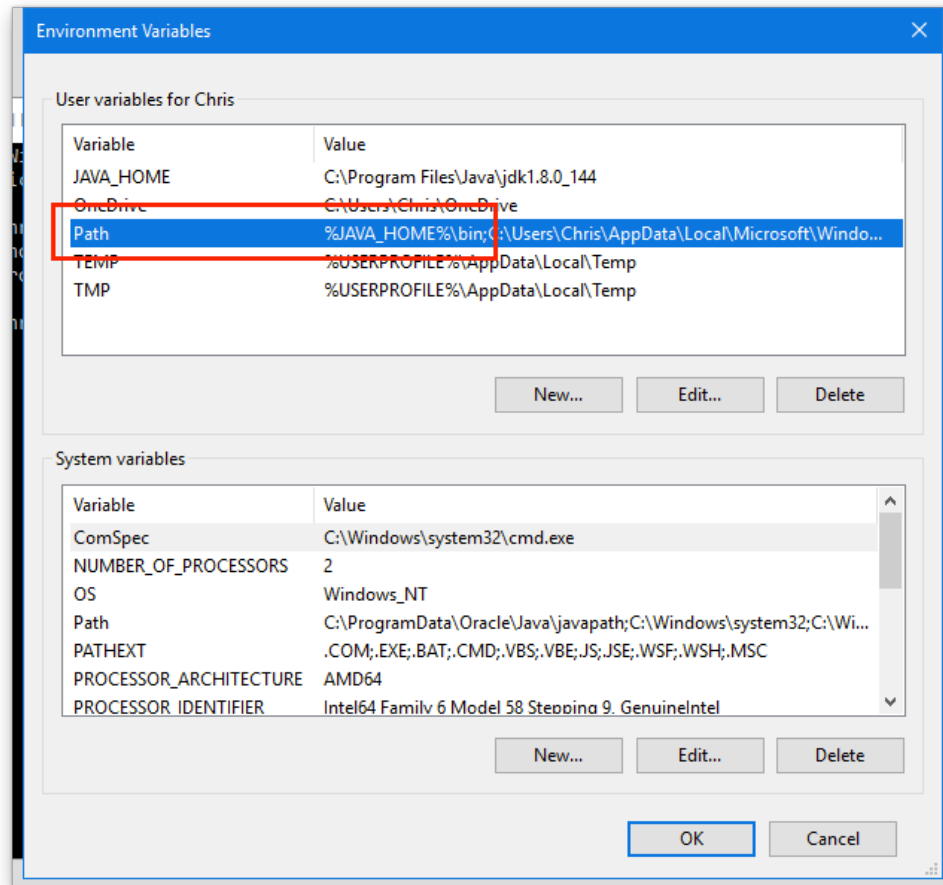


Figure 15: JAVA\_HOME added to PATH

6. Once you have set the variables click the OK button in the Environment Variables dialog box, and click OK in the System Properties dialog box. This saves the changes.

What did we just do, and how can we see if we did it right?

The Windows Command Prompt uses the Path variable to search for programs that are not built into it (such as CD or DIR). Note that there is a Path variable in the System variables list, too. The Windows Path is equal to the sum of whatever is in the System Path *plus* whatever is in the User Path (which we just updated).

For example:

If the System Path is set to C:\Windows\System32;C:\temp

If the User Path is set to C:\ProgramFiles\Java\jdk1.8.0\_144\bin

The Windows Path is C:\Windows\System32;C:\temp;C:\ProgramFiles\Java\jdk1.8.0\_144\bin (remember we separate directories in the path using the semi-colon).

When we enter the command `javac` on the command line, essentially each directory in the Path is searched for a file named `javac.exe`. The first file found with that name is executed. If a program with that name is not found, we will get an error message like this:

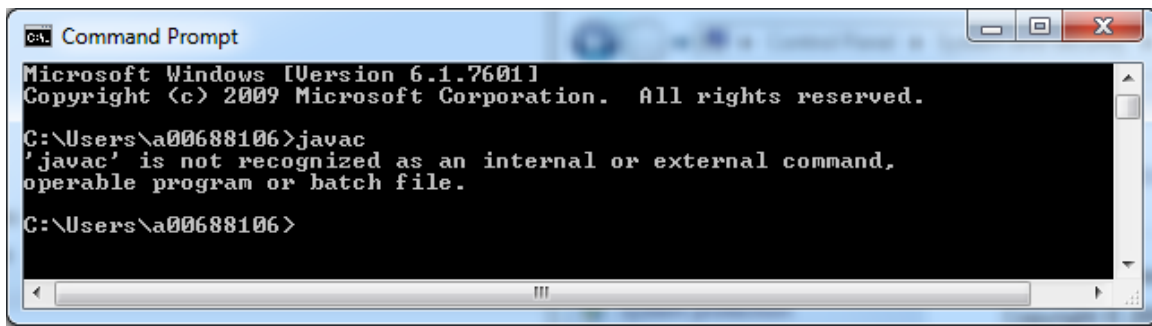


Figure 16: javac is not recognized...

When we type javac the Command Prompt in our example looks for:

- a) C:\Windows\System32\javac.exe (does not find it)
- b) C:\temp\javac.exe (does not find it)
- c) C:\ProgramFiles\Java\jdk1.8.0\_144\bin\javac.exe (finds it and executes it).

This is how you can add new commands to your system without resorting to contortions such as putting the command program into the C:\Windows\System32 folder (don't do this) or putting your source files into the same directory as the javac.exe (don't do this).

7. Open a Command Prompt window and enter the javac command.
8. If you already have a Command Prompt window open, you will have to close it and re-open it. Why? Because the Command Prompt does not update automatically – we need to close it and open a new instance in order to include the new variable. You should see something like this. This is the javac program telling you how to use itself:

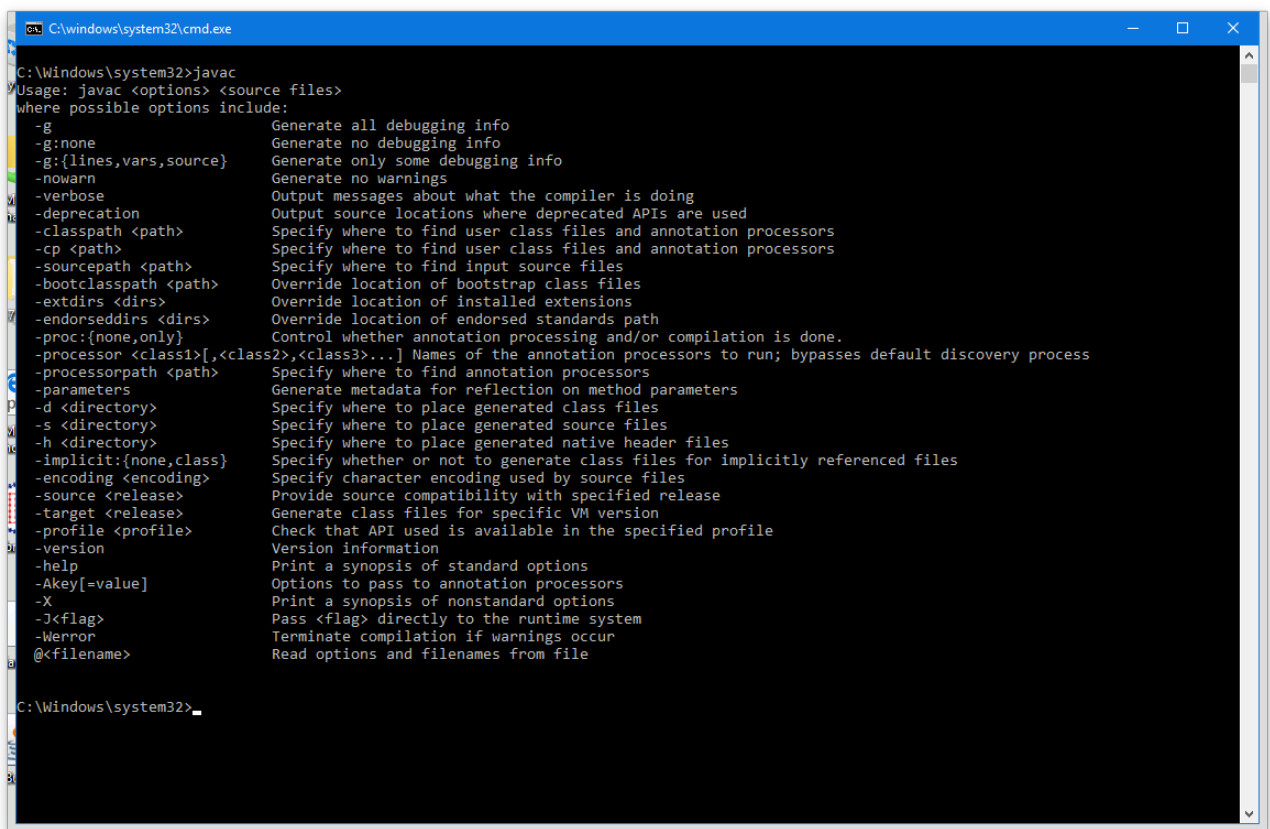


Figure 17: javac works hooray

We're going to return to the Command Prompt later in this lab. Right now, let's install the rest of our tools.

#### 4. EVERYONE: Install the Eclipse IDE

We can write Java programs using a text editor and compile them using the Java compiler on the Command Prompt, but why do that when there are great tools out there for software development that make development easy and fun? You're going to start using one of the most popular tools right away.

The tool you will use the most in this course is Eclipse. Eclipse is an Integrated Development Environment (IDE). It makes programming easier, more efficient, and more fun.

We are going to download and install the latest version of Eclipse – Eclipse OXYGEN (4.7):

1. Navigate to <https://www.eclipse.org/downloads/eclipse-packages/>
2. Download the Eclipse IDE for Java EE Developers (64 bit version to match the JDK)
3. When the download has completed, double click the downloaded file. The installation will take a few moments.

Eclipse is one of the most popular Integrated Development Environments (IDEs) used in industry. Other common IDEs are IntelliJ IDEA, NetBeans, Android Studio, and Microsoft Visual Studio (which is for C# development). Your subsequent courses may want you to use a particular IDE, or let you choose. Eclipse is great because it gives you continual feedback on correctness and style, and has the advantage that it is widely used in the Java industry.

#### 5. WINDOWS AND MacOS: Launch Eclipse

1. Once you have installed Eclipse (do not do this on the BCIT lab computer), go ahead and launch it.
2. When Eclipse starts, it begins by asking for a directory to use as a workspace. **You only need one workspace.** It is the folder where you will create and store all your Java projects, labs, and assignments. In the following figure, the workspace is at C:\EclipseWorkspace. You may want to check the checkbox so you always open this workspace by default:

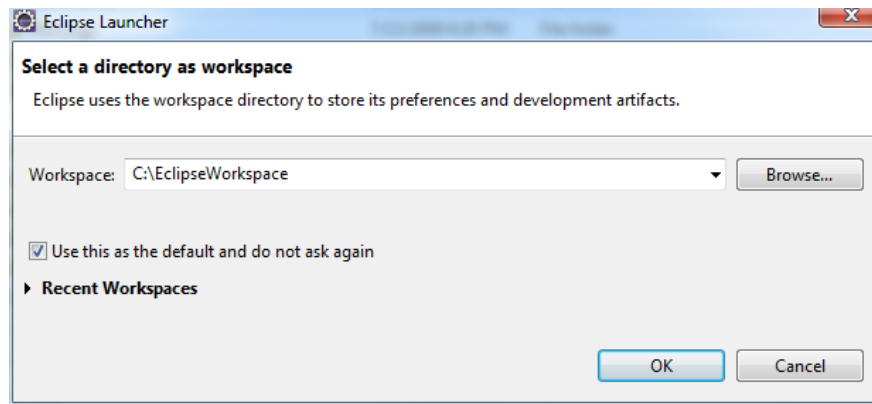


Figure 18: Set the Eclipse workspace directory

3. Once you have selected your Workspace, Eclipse finishes loading and displays the Welcome window. It should look like this. You can uncheck the option in the lower left corner if you do not want to see this each time you open Eclipse:

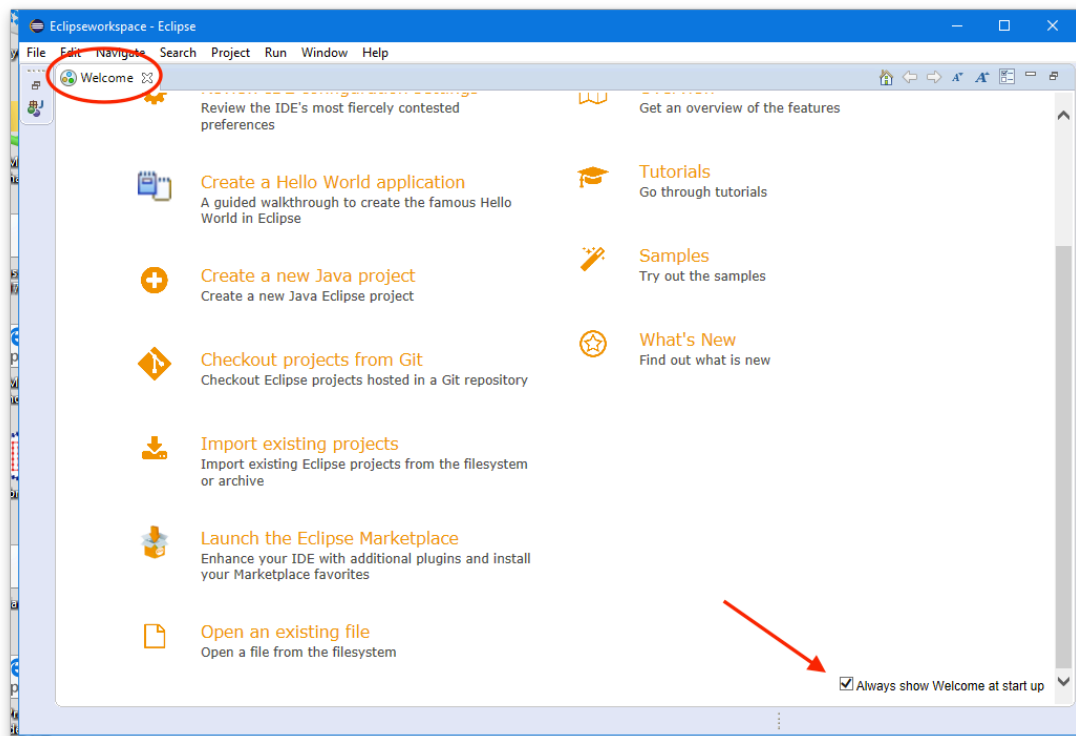


Figure 19: Eclipse welcome screen

4. Close the Welcome window.
5. Eclipse has perspectives. The workspace is organized differently for different perspectives. Different perspectives are better for different tasks. We will always use the **Java perspective** in COMP 1510. You can ensure you are in the Java perspective in one of two ways:
  - a. From the main menu select Window > Open Perspective > Java
  - b. In the upper right corner of the workspace, click the Open Perspective window and select Java.

You will see a workspace similar to this (there is a lot of white space):

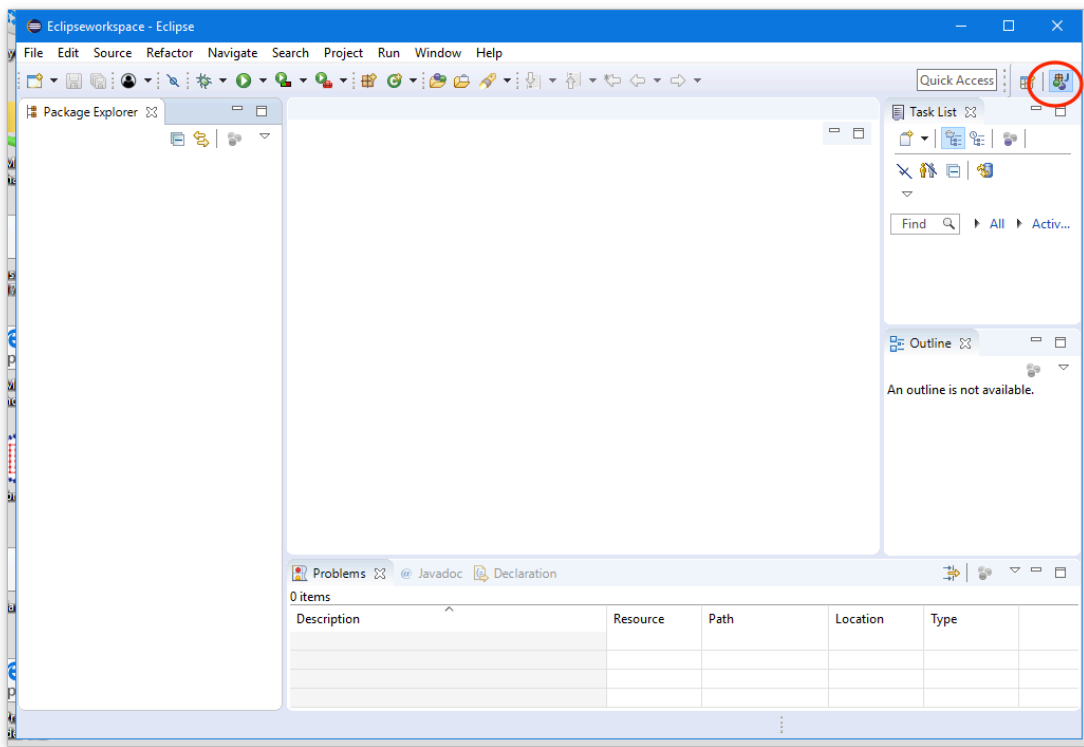


Figure 20: Fresh Eclipse installation

## 6. WINDOWS AND MacOS: Install the e(fx)eclipse Plugin

One of the great things about Eclipse is that there is an entire ecosystem of add-ons that we can download and install. In the Eclipse world we call these plugins.

We need to install the e(fx)clipse plugin so we can use JavaFX later in the term. JavaFX lets us build graphical user systems in Java.

There are various sources for Eclipse plugins. The Eclipse Marketplace contains the plugins we will install today.

1. From the menu in Eclipse, select Help > Eclipse Marketplace:

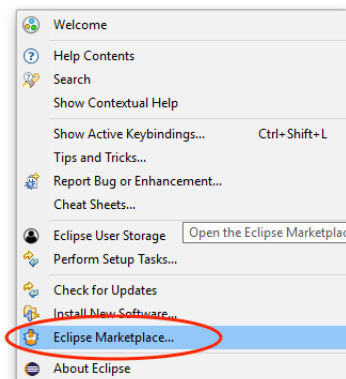


Figure 21: Select Help > Eclipse Marketplace

2. In the Search tab, enter e(fx)clipse into the Find window, and press enter. Eclipse will search online for the plugin called e(fx)clipse:

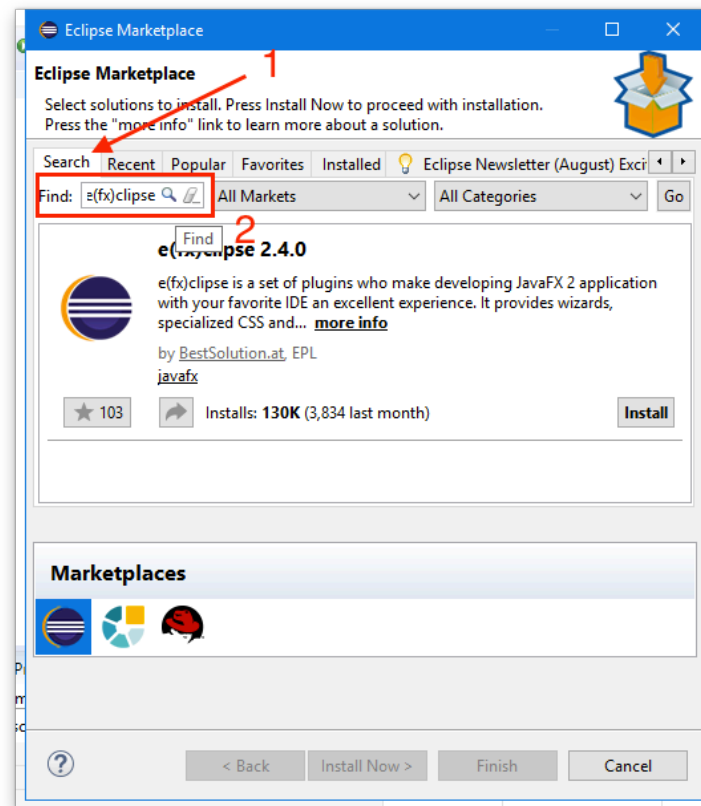


Figure 22: Search for e(fx)clipse

3. Click the Install button to install e(fx)clipse. Accept the terms of installation. After installing e(fx)clipse, you may need to restart Eclipse. Go ahead.
4. That was easy, wasn't it. Now let's install a code style helper called Checkstyle.

## 7. WINDOWS AND MacOS: Install the Checkstyle Plugin

Java programmers are quite particular about the format of our code. For COMP 1510, we will use an Eclipse plugin called Checkstyle that will give us continuous feedback on our code style and format.

Checkstyle is described in great detail at <http://checkstyle.sourceforge.net/>. Checkstyle is a development tool to help programmers adhere to a coding standard. By default, it supports the Google Java Style Guide and Sun Code Conventions, but is highly configurable. It can be invoked with an ANT task and a command line program.

1. In Eclipse, click on the Help menu tab and select Eclipse Marketplace. Enter checkstyle in the Find box and click Go. The following window will appear. You will need to scroll down a bit to find the Checkstyle plugin:



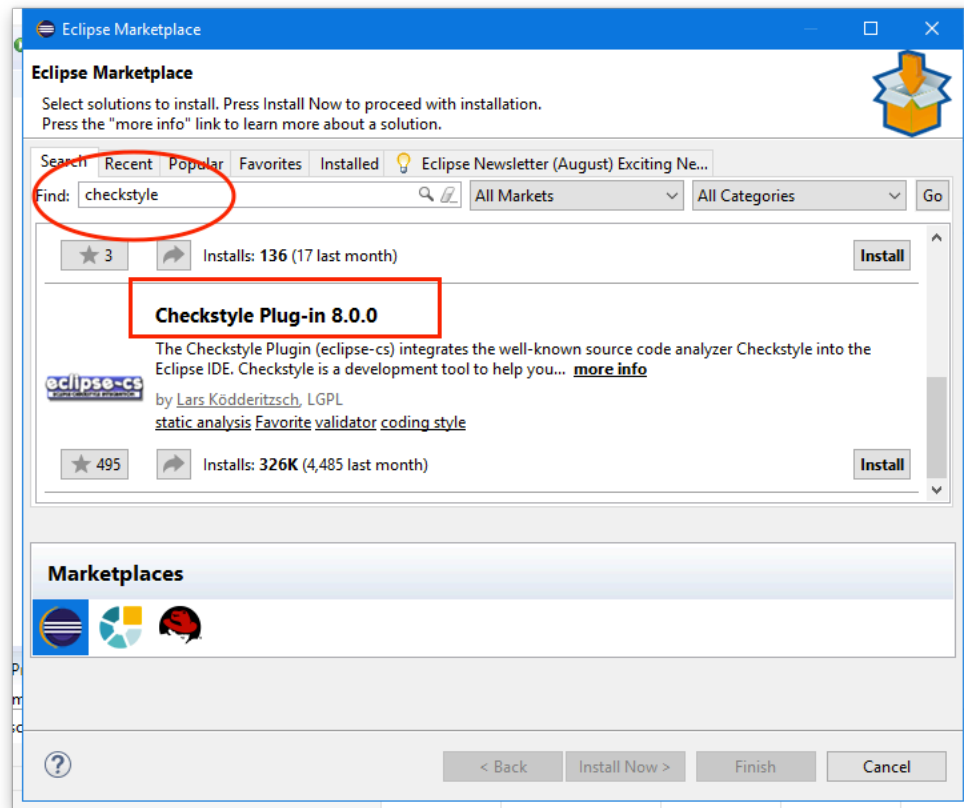


Figure 23: Eclipse marketplace (note the plugin is now version 8.0..0)

2. Click on the Install button for the **Checkstyle Plug-in 8.0.0** to install it.
3. After a (perhaps lengthy) pause, accept the license and select Finish. Eclipse will download and install the plugin.
4. Click OK if you get a warning about unsigned content and click Restart Now when Eclipse indicates you need to.

## 8. WINDOWS AND MacOS: Install SceneBuilder

There is one more piece of software we need to install. **SceneBuilder** works with the JavaFX ecosystem and may be helpful later in the term when we learn about graphical user interfaces (GUI). The current version is 8.3.0, released in December 2016.

1. Open your browser and navigate to <http://gluonhq.com/products/scene-builder/>
2. Scroll to the bottom of the page and select your installation package:
  - a. **Windows:** Windows Installer (x64)
  - b. **macOS:** Mac OS X dmg.
3. Install SceneBuilder by double-clicking the downloaded file.
4. When the installation is complete, SceneBuilder will automatically open. It's perfectly fine to close it. We won't use it for a few weeks.
5. Go ahead and delete the installation file. You won't need it, and it's just taking up space.



## 9. Configure Eclipse

Installations are complete. Now it's time to configure our tools. Let's take a look at the Preferences page that configures how Eclipse works.

1. From the Menu, select Window > Preferences and expand Java/Code Style/Formatter:

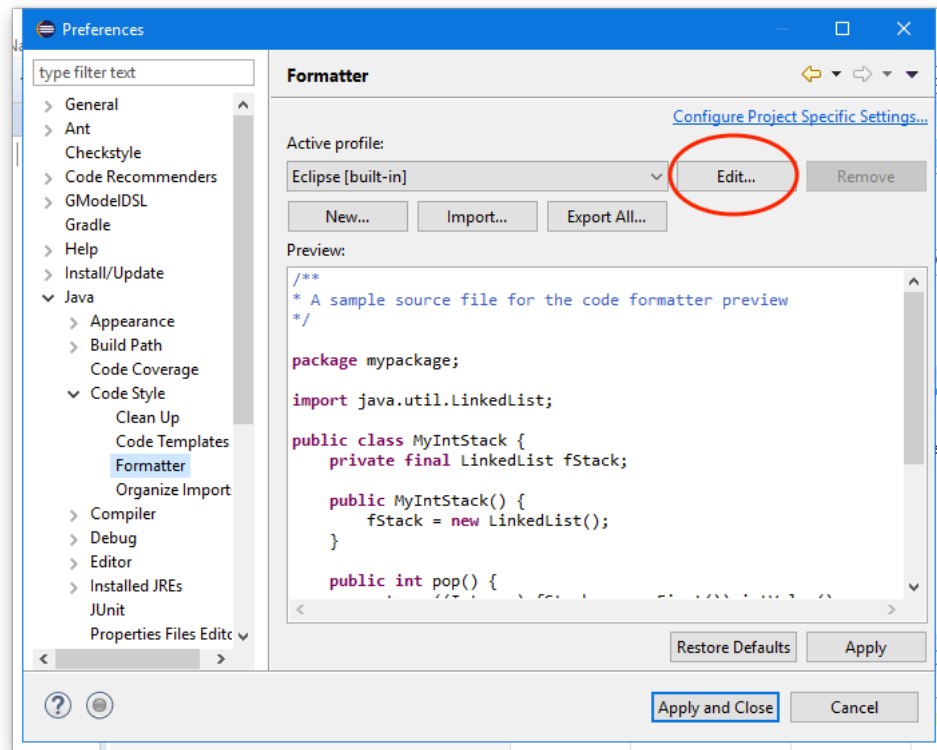


Figure 24: Window > Preferences > Java > Code Style > Formatter > Edit

2. We will create a standard profile so 4 spaces are used for indentation instead of tabs. Click the Edit... button.
3. Change the profile name to Standard.
4. In the Indentation tab, change Tab policy to Spaces Only, and ensure both Indentation size and Tab size are set to 4. Your result should look like this:

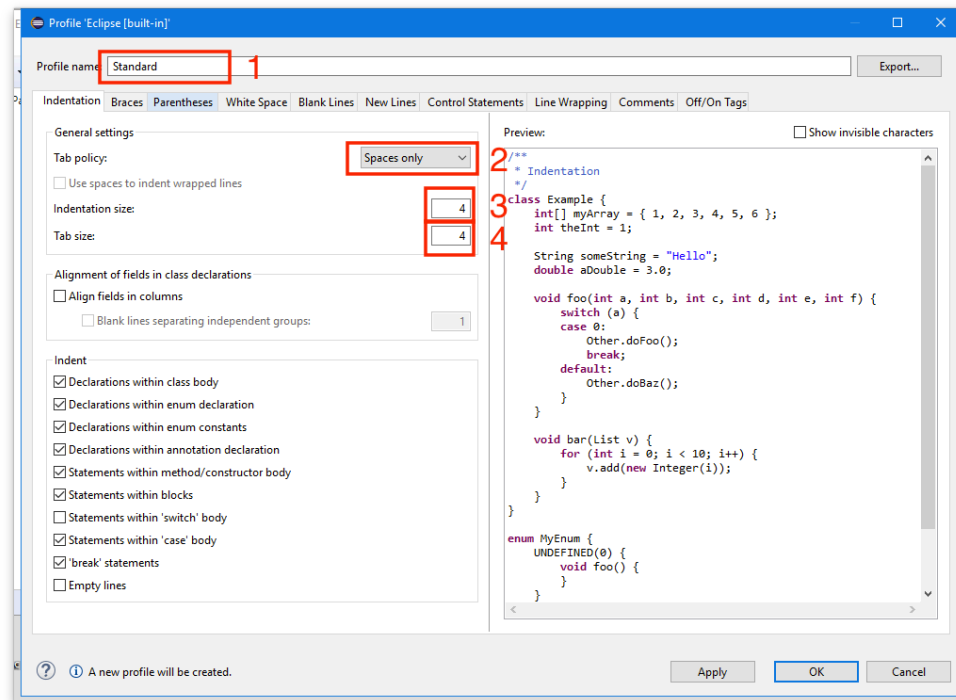


Figure 25: Set Tab policy to 4 spaces

- Click OK to accept the new profile.
- Now in the Preferences window navigate to Java > Editor > Content Assist. This is the dialogue which controls the suggestions Eclipse's editor makes will you are programming. This may be useful after you have some experience, but right now it will just get in your way. Turn off auto-activation by deselecting the Enable auto activation box in that section:

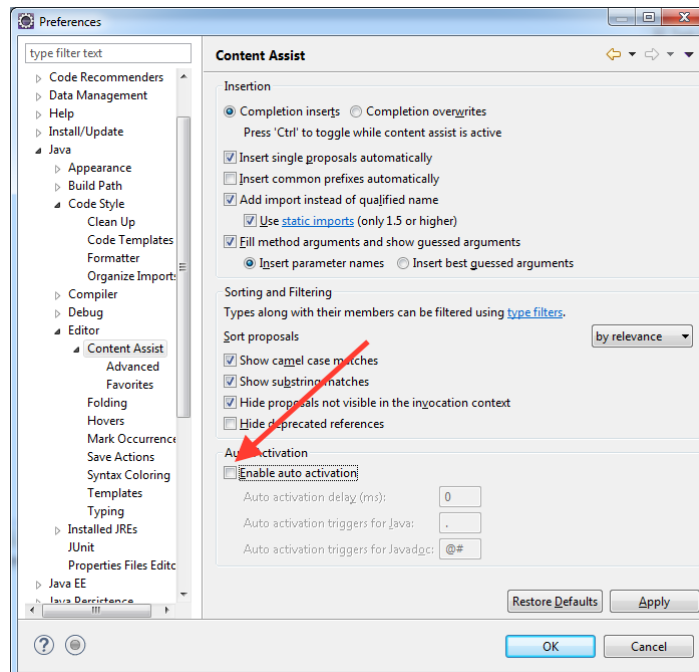


Figure 26: Disable code assist auto activation

- Click OK in the Preferences window to accept the changed preferences and close the window. **Content assist is still available by hitting control-space**, but will not clutter up the edit screen as we type.

8. You will find that if you type quickly, you might enter code faster with content assist turned off. Of course you need to know what you are doing, but that's why you're here!

## 10. Create an Eclipse Project

Eclipse requires that you create a project for your source files. A project is simply a collection of files and settings for a program or, in our case, an assignment or lab.

1. To create a new project, click on the New button (left-most button under the menu bar) and select Java Project, or select File > New > Java Project from the menu. This opens the New Java Project dialog box.
2. There are a number of default settings that a new Java project has. We will start by providing a project name. Traditionally the first program in any new language is called Hello World. Use meaningful names for your projects:

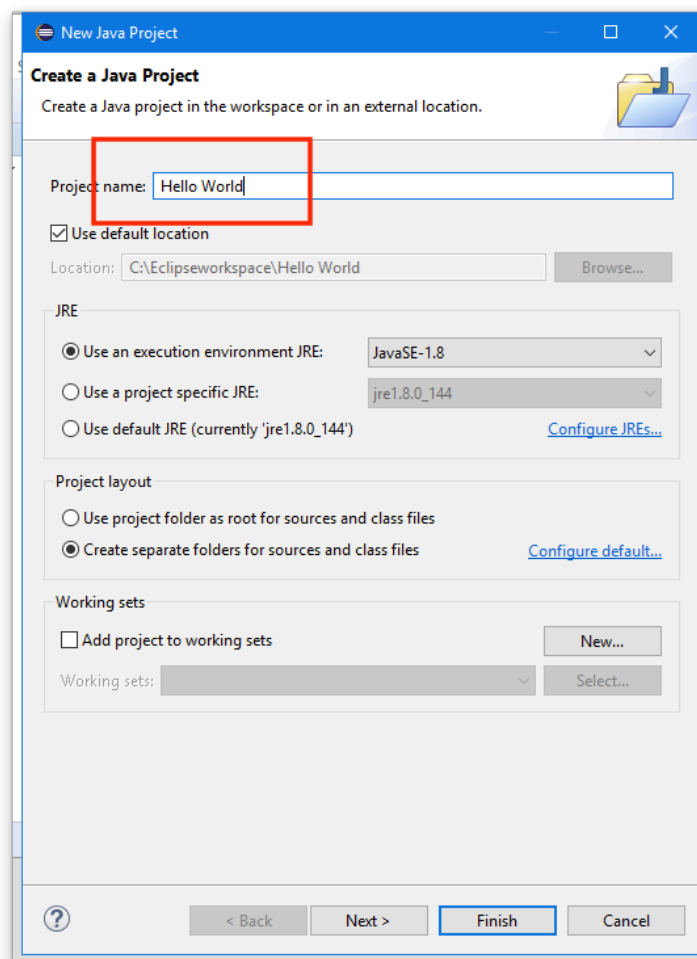


Figure 27: New Java project needs a good name

3. Go ahead and click Finish. In the leftmost pane of the workspace you should see your new Java project appear. It should look like this:

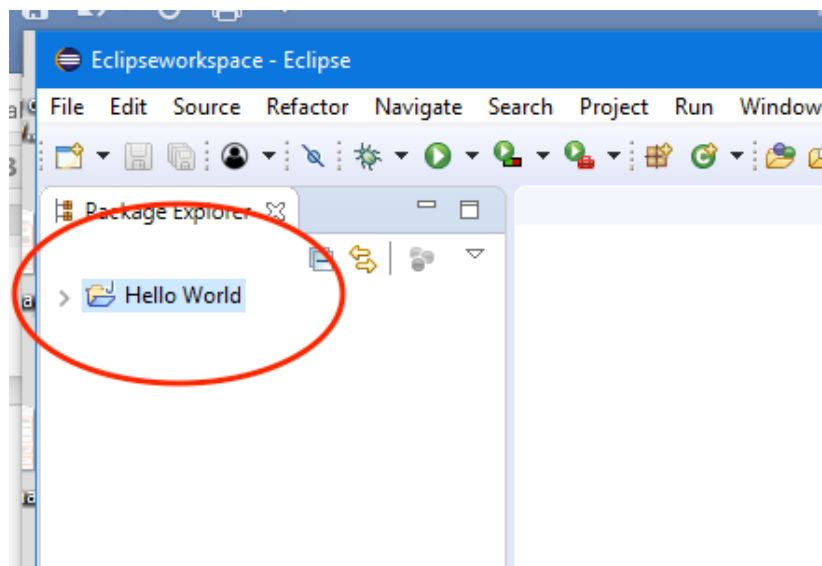


Figure 28: New Java project in the Package Explorer

## 11. Add a Java Source File to the Project

The leftmost pane of the workspace should be called the Package Explorer. If it's not called the Package Explorer, you can find it by choosing Window > Show > Package Explorer. As you create new Java projects, they will populate this list.

1. If you click on the little triangle beside the new project folder, it will expand the project and you will see something like this:

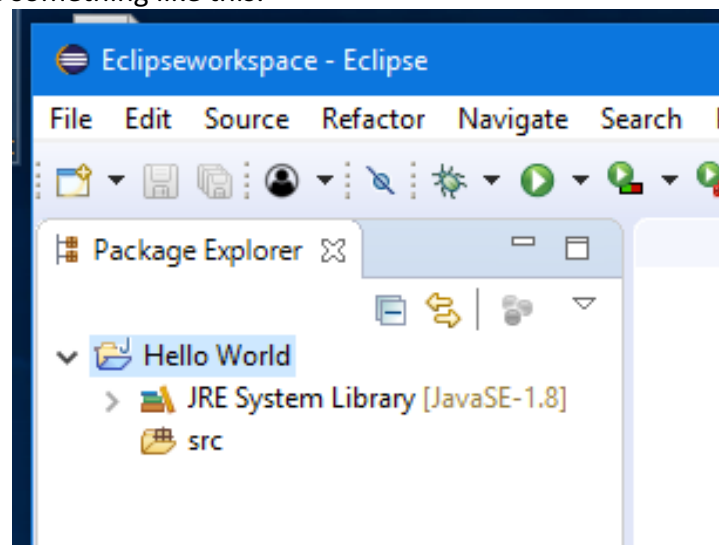


Figure 29: Contents of a new, empty project

2. The **JRE System Library** folder links your project to the Java 8 library. You can expand this folder and look inside it, but we won't use this folder in COMP 1510.
3. The **src folder** is the folder that contains our source files. We want to add a source file to our project, so right-click the src folder and choose New > Class. This opens the New Java Class dialog:

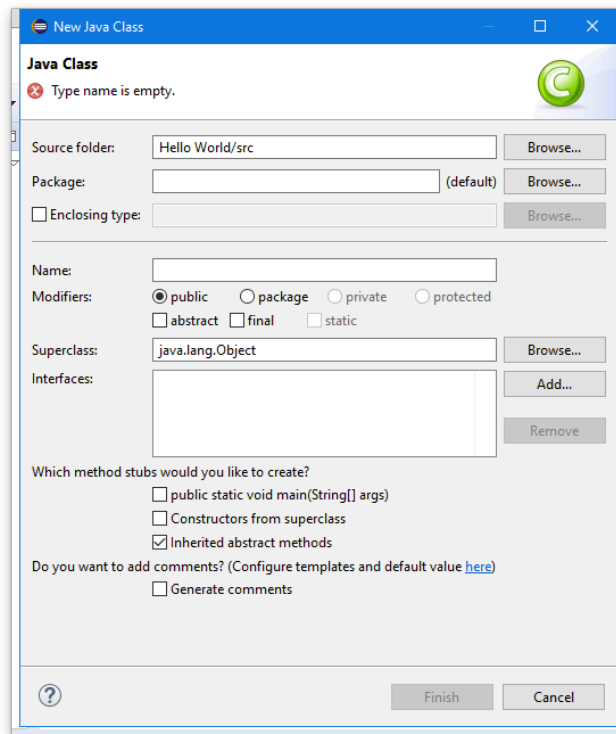


Figure 30: New Java Class dialog

4. The name of the initial class is often called “Main” (enter this into the Name field). Later in this course when you have several files in a project, you will name the classes according to their function, such as Driver, or Student, etc.
5. Put your file into a package p1 (enter this in the package field).
6. Check that the source folder entry field corresponds to the src folder under your project. If it is not, click on the Browse button and locate it.
7. Click in the check box to create a public static void main stub.
8. Also check the box “Generate Comments”
9. Make sure your dialog box looks like the following figure. If it does, go ahead and click Finish:

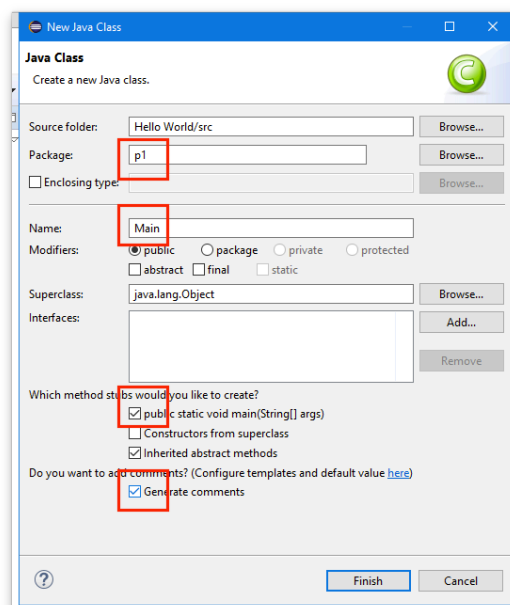


Figure 31: New class dialog box (completed)

10. Eclipse creates an empty source file called Main.java which contains the source code for a class called Main. Inside the Main class, there is a main method with a TODO comment which we will replace with code:

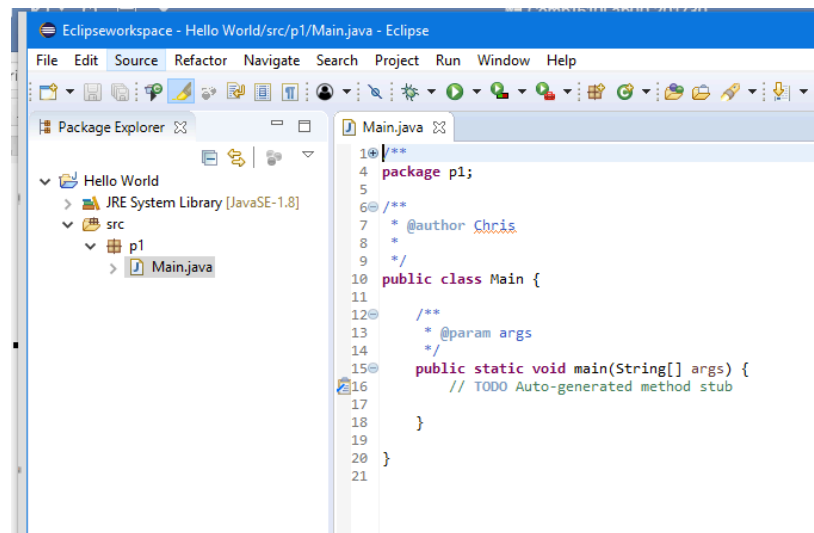


Figure 32: Our Main.java source file

11. The name of the class MUST match the name of the file. Since we named our class Main, Eclipse will create a file called Main.java. Remember though that in Java class names always begin with a capital letter.
12. **NOTE: Never modify source files or move them around unless you are inside Eclipse. If you try to modify project files outside of Eclipse, you may corrupt the project.**
13. Let's look at the source code for the file. If your line numbers aren't showing in the left margin, you can activate them via Window > Preferences > General > Editors > Text Editors > Show line numbers:

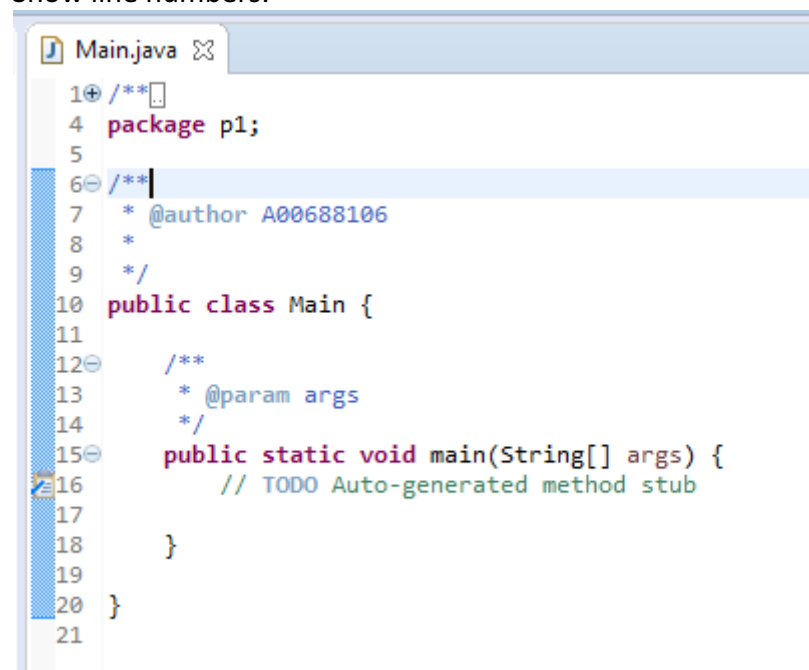
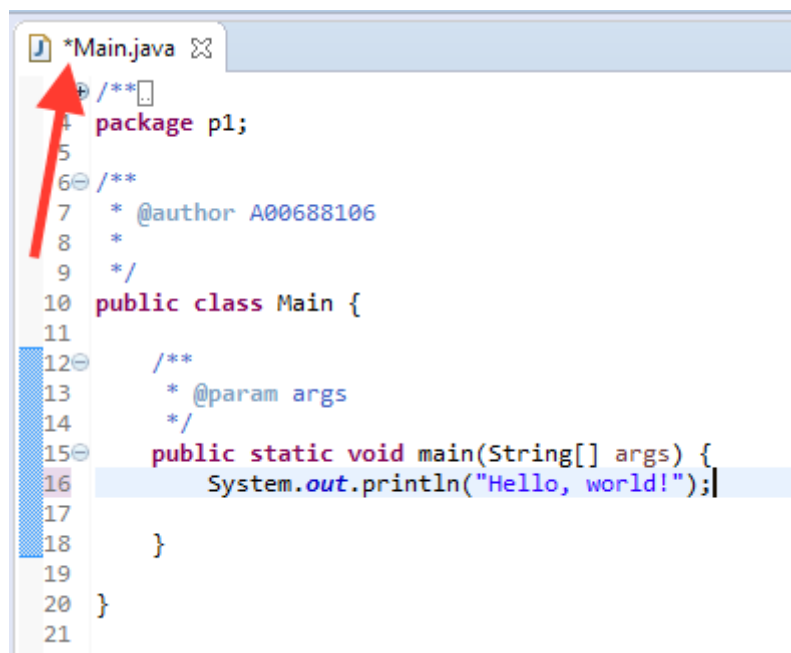


Figure 33: Main.java source file

- a. The class and the main method are both “public” so that the Java Virtual Machine can access them (you’ll learn more about this later in the term).

- b. The “static” keyword permits the main method to be invoked (called) without first creating an instance of the Main class (don’t worry, you are not supposed to understand what that means yet either).
  - c. The “void” is the return type (void means return nothing).
  - d. The String[] is an array of String objects (the arguments passed in from the command line – again, do not worry, we will cover strings and arrays in future labs).
  - e. args is the name of the String[] variable.
  - f. For now, you can treat public static void main(String[] args) as a magic incantation that is necessary in Java programs.
14. Note the style that Eclipse uses for curly braces { }. The opening brace is at the end of a line, and the corresponding closing curly brace lines up underneath the line containing the opening brace. This is one of the popular styles used in Java programming (the other being shown in the text, where the braces are each on their own line). The preferences can be changed to use either style, but the default seems to be the more popular one.
15. The comments before the class and main method are called Javadoc comments. Javadoc comments will be required in the assignments to properly document your programs, as we will see later.
16. In the source code delete the TODO comment and replace it with `System.out.println("Hello, world");`
17. Save the program (File > Save in the menu, or Control-S, or click on the save button). You can always tell that a file needs to be saved because the tab with the file name will have an asterisk (\*) before the file name. When you save your changes, the asterisk (\*) disappears (see Figure 26).



```
1  package p1;
2
3  /**
4   *
5   * @author A00688106
6   */
7
8  public class Main {
9
10     /**
11      * @param args
12      */
13     public static void main(String[] args) {
14         System.out.println("Hello, world!");
15     }
16 }
17
18
19
20
21
```

Figure 34: Hello, world (saved)

## 12. Compile and Execute a Java Project



1. Each Java program must first be compiled from source code into bytecode. The bytecode is then executed by the Java Virtual Machine (and probably further compiled to run faster on the particular platform that it is running on).
2. Eclipse automatically compiles (builds) your program when you save it. If you want to recompile (rebuild) your project from scratch, select Project > Clean from the menu.
3. There are various ways to run (execute) your program in Eclipse:
  - a. Click on the project folder in the Package Explorer to select the project and then click on the green arrow in the menu
  - b. Select Run > Run in the menu
  - c. Right-click the source file and choose Run As > Java Application
  - d. If the "Run As" pop-up window appears, select Java Application and click OK.
4. The output "Hello, World!" appear in the Console window. Hooray! You just wrote, compiled, and executed your first Java program!

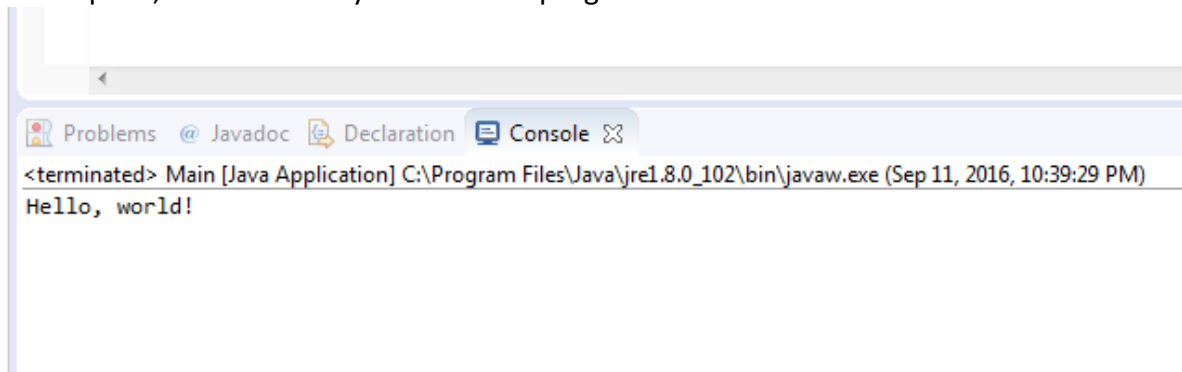


Figure 35: Hello, world! (console output)

5. Now that Eclipse knows how to run your project, when you click on the Run button, it will run the project without asking.
6. If you have several main methods in your project, you can right click on one of the classes and select "Run As" to run it.

## 13. Create a Project Archive (JAR) File

Usually a Java project will have several files. It is convenient for a user to package these up into one file, called a Java Archive (JAR) file. This is very similar to a ZIP file, but it stores Java class files (the files that contain the bytecode). This is a simple way of packaging up a project for people to run:

1. To create a JAR file, right click on the project and select Export.
2. In the pop up window, select Java > Runnable JAR file:

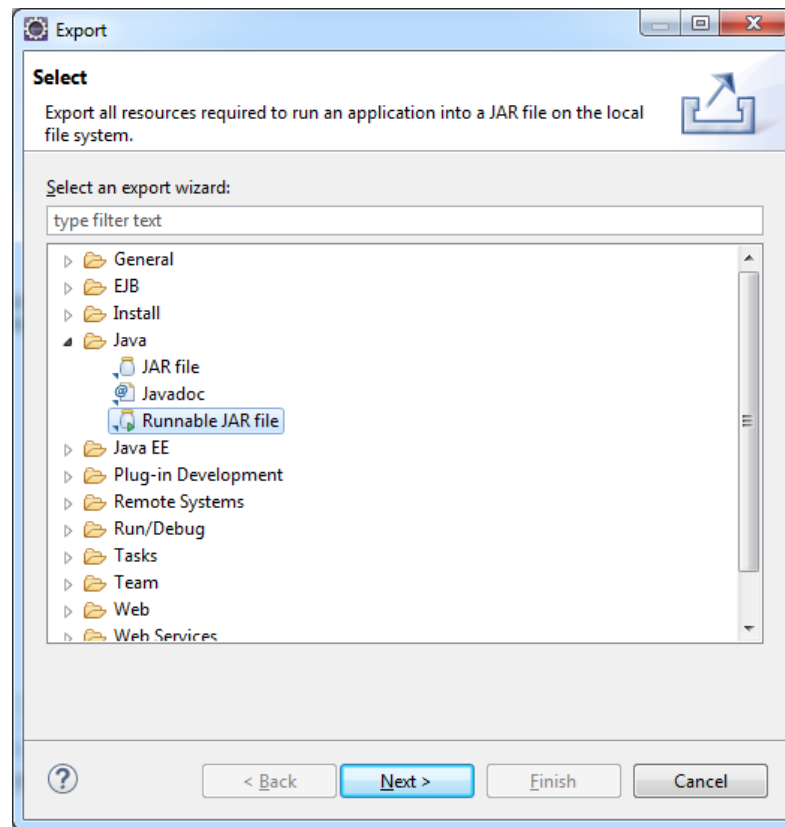


Figure 36: Export a runnable JAR

3. In the Runnable JAR File Export dialog, select the Main file from the dropdown under Launch configuration.
4. Select an Export destination (try the desktop) and call the JAR file Lab0.jar.
5. Before you click Finish, make sure your choices look similar to this (your export destination will be different, of course):

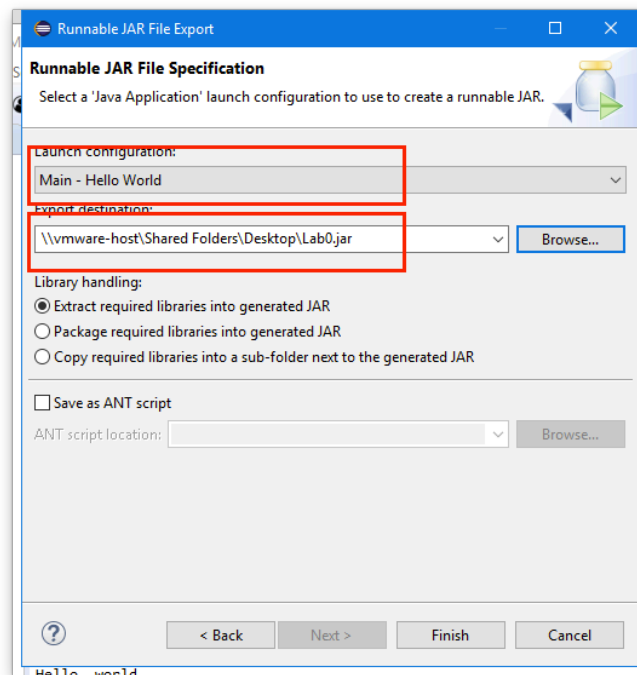


Figure 37: Runnable JAR settings

- When you click Finish, a JAR file will be created. This JAR file is a runnable JAR, i.e., an executable program. You can run your project from the command line by navigating to the folder containing the jar file in Command Prompt and entering the command `java -jar Lab0.jar` (try it!).

## 14. Run Examples from the Text (Import a Project)

Another thing you should do is follow along with the examples in the text. To do this, download the file `Comp1510Examples.zip` from D2L, and import the project into Eclipse as described in the following steps:

- Select `File > Import > General > Existing Projects into Workspace > Next`.
- Choose “Select archive file” and browse to the `Comp1510Examples.zip` file. The `COMP-1510Examples` checkbox should be selected under Projects:

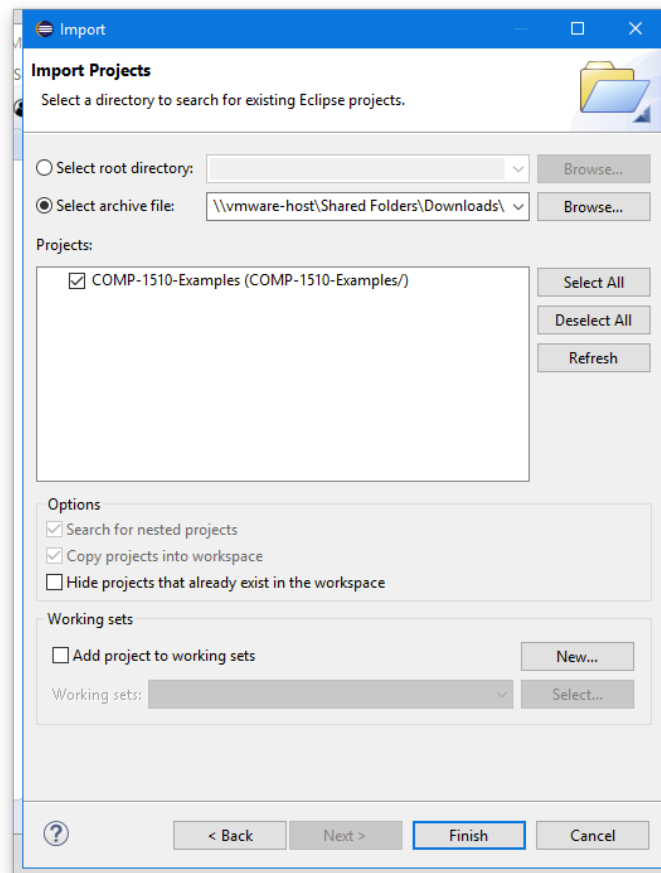


Figure 38: Importing the text project

- Click on Finish.
- Click on the triangle to expand the `Comp1510Examples` project in the Package Explorer, and click on the triangle beside the `src` folder to reveal the project folders (they are in packages):

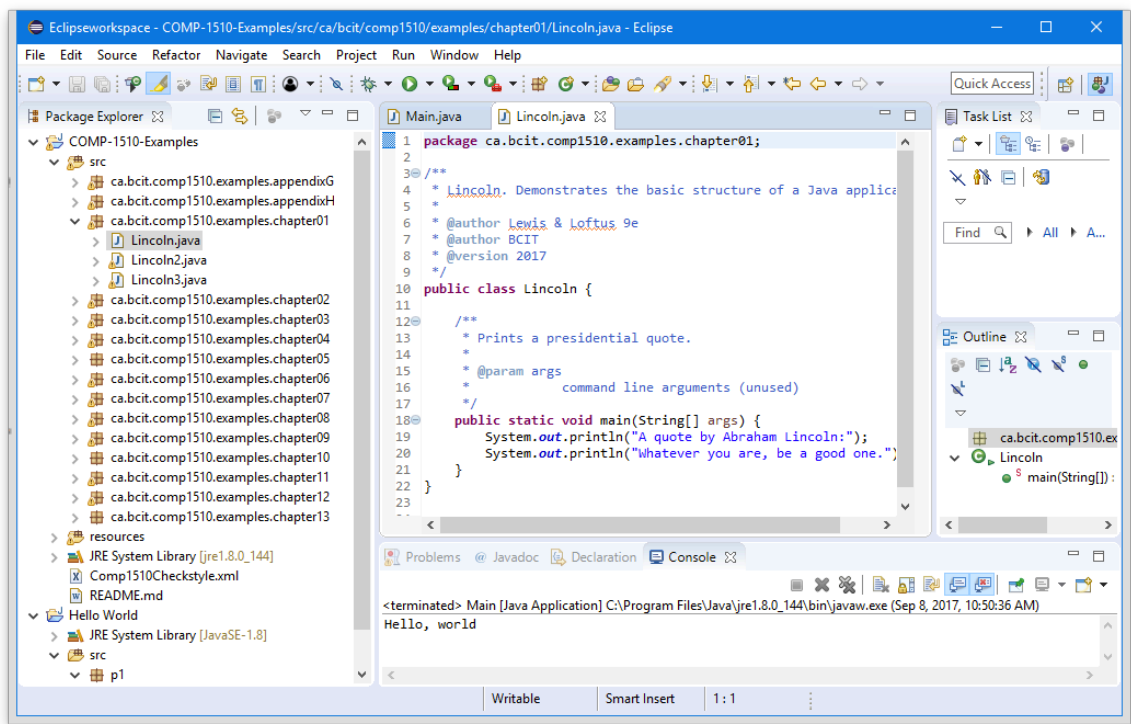


Figure 39: Source file for a Chapter 1 program

5. Open the source examples for any chapter by clicking on the triangle beside the corresponding package name.
6. Open any of the programs for reading/editing/running by double clicking on it.
7. Run any of the programs:
  - a. Select the source file in the Package Explorer and then select Run > Run in the menu
  - b. Right-click the source file and choose Run As > Java Application
  - c. If the “Run As” pop-up window appears, select Java Application and click OK.

## 15. Configure Checkstyle

1. We need to add the Comp1510Checkstyle parameters to define our initial style (as described next). You may need to repeat this for subsequent assignments to get the latest style definition. Here’s how:
2. Note that in the Comp1510Examples project there is a file called Comp1510Checkstyle.xml. We have developed this style guide for Checkstyle and COMP 1510.
3. On the Eclipse menu bar navigate to Window > Preferences > Checkstyle.
4. We want to add a new default configuration. Click on the New button:

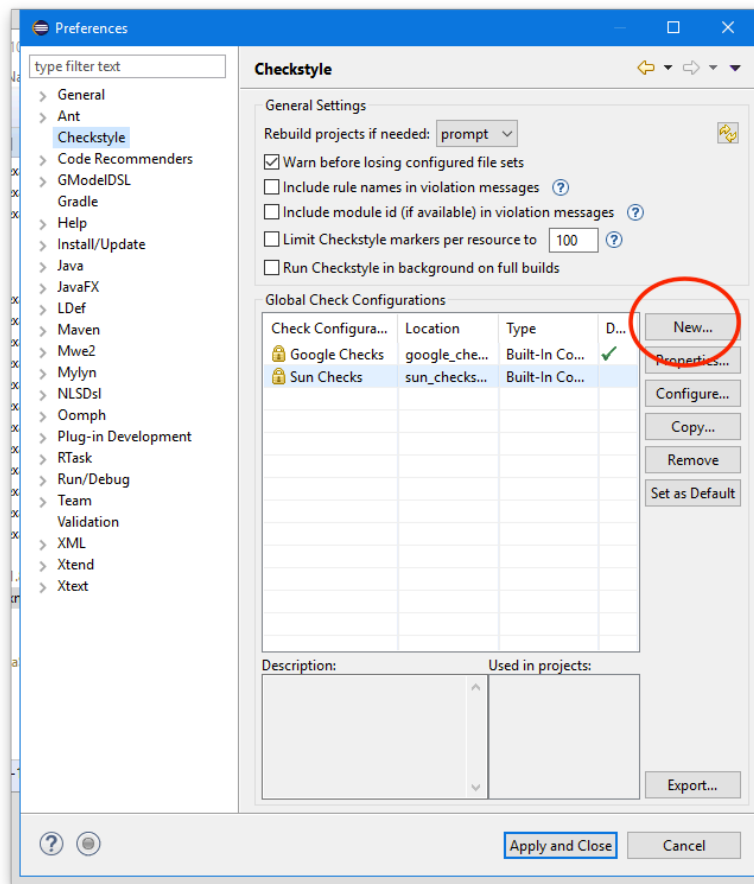


Figure 40: Create a new Checkstyle configuration

6. Select Type: Project Relative Configuration.
7. Enter a Name (a good name is 1510 Checkstyle)
8. Browse to the location of the Comp1510Checkstyle.xml in the Comp1510Examples project:

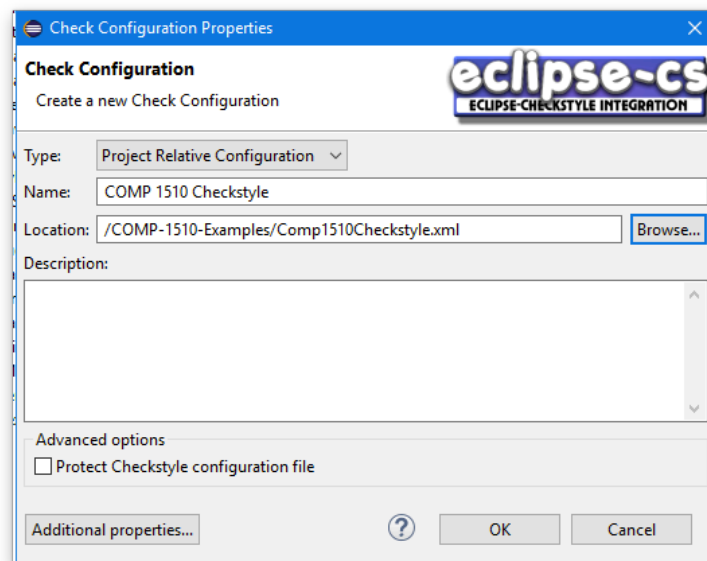


Figure 41: New check configuration dialog

9. Click OK.
10. Select the configuration we just created, and click the Set as Default button to make that configuration file the default:

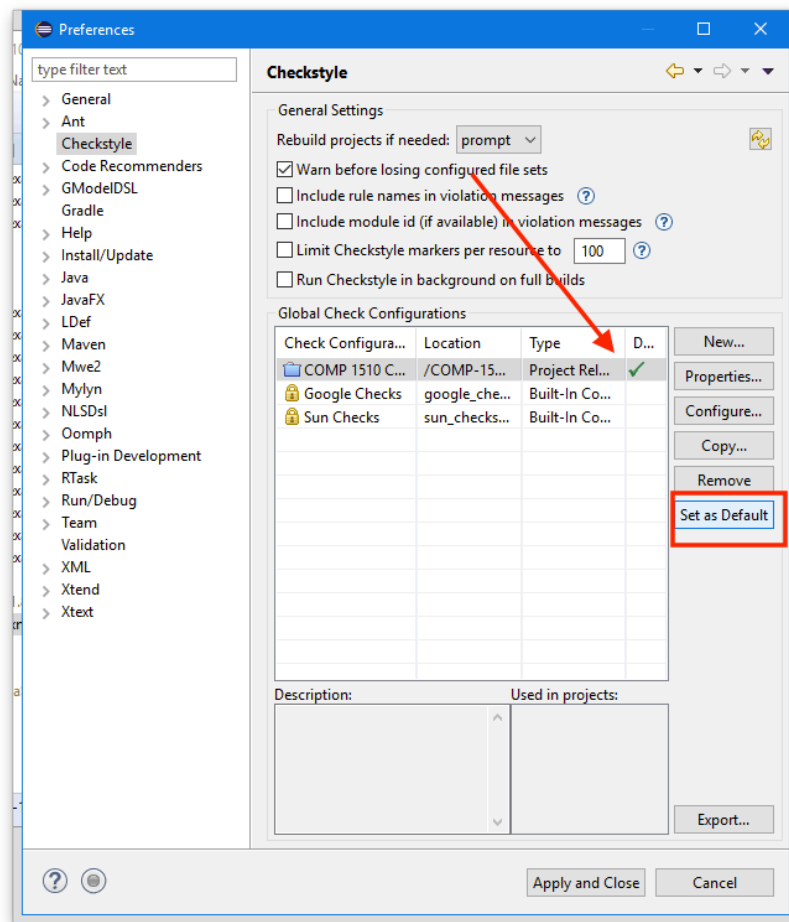


Figure 42: Set as default configuration

11. Click Apply and Close to close the Preferences window and Yes if Eclipse suggests that some projects need to be rebuilt.
12. To apply Checkstyle to a project, right click on the project and select Checkstyle > Activate Checkstyle in the popup menu. Do this for the Hello World project.

### Working With Checkstyle Violations

13. The Checkstyle plugin will flag lines that contain style violations by placing a little yellow magnifying glass symbol in the left margin. We saw in the lecture that the Lincoln2 and Lincoln3 examples had poor use of white space. If you open them now you will see Checkstyle does not like them either:

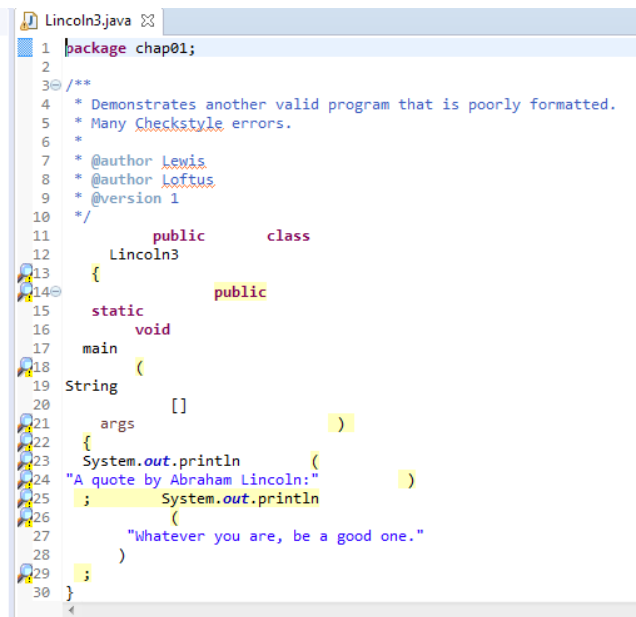


Figure 43: Lincoln3.java contains Checkstyle violations

14. Move the mouse cursor over the magnifying glasses to see the errors.
15. When you are writing code, it is best to eliminate all of the Checkstyle errors as you go. We will need to complete the material from chapter 2 before we can do this, so don't worry about it too much right now. When the first assignment is released after the end of chapter 2 you will need to correct the Checkstyle violations.
16. When you edit a source file and save it, Checkstyle will automatically reassess the file and update the violation list.

## 16. Javadocs

You may recall that in the discussion of comments, the comment that begins with `/**` is called a Javadoc comment. Javadocs are the standard industry method used to document Java code.

You will need to comment your code with Javadoc comments for the labs and assignments.

Eclipse can process the Javadoc comments and generate a corresponding document about your class to share with other developers so they know how to use your code.

After you have done this, Eclipse can use the Javadoc to provide users with helpful popups in your code. Let's create the Javadoc for the Main class we wrote:

1. Click the Project menu and select Generate Javadoc.
2. The Generate Javadoc dialog box opens. Press the Configure button, and navigate to the `%JAVA_HOME%\bin` folder we created at the very beginning of the lab (**macOS** users navigate to `/Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin`).
3. Select the `javadoc.exe` file and press Open.
4. Click the project you want to process.
5. Select your desired Destination folder (the preset doc folder in the project is good):

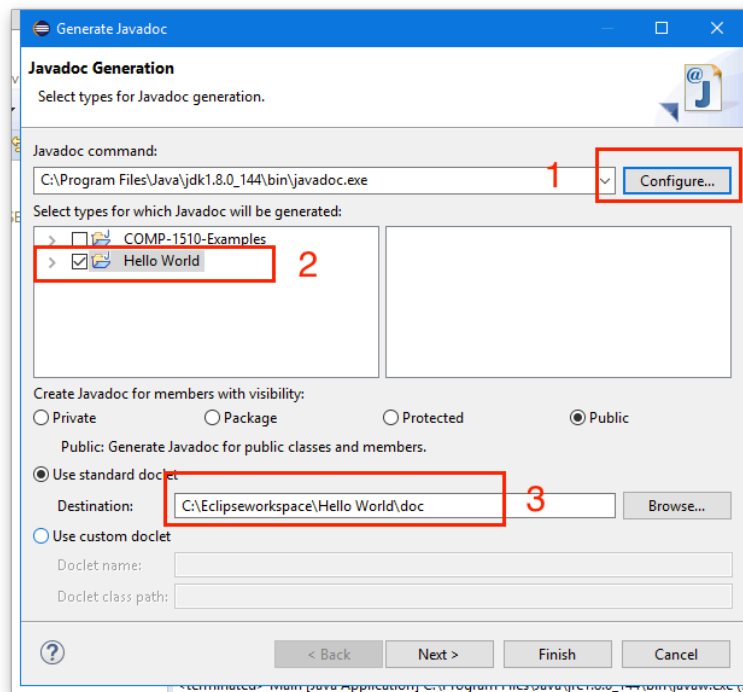


Figure 44: Generate Javadoc

6. Click Finish. Answer Yes to the question about updating the Javadoc location for the project. Once Eclipse is done, you will see a Doc folder appear in your project. Double click the index.html file inside it. Our file is quite simple because we haven't added any comments yet, but we will see examples of fully populated Javadocs later in the term:

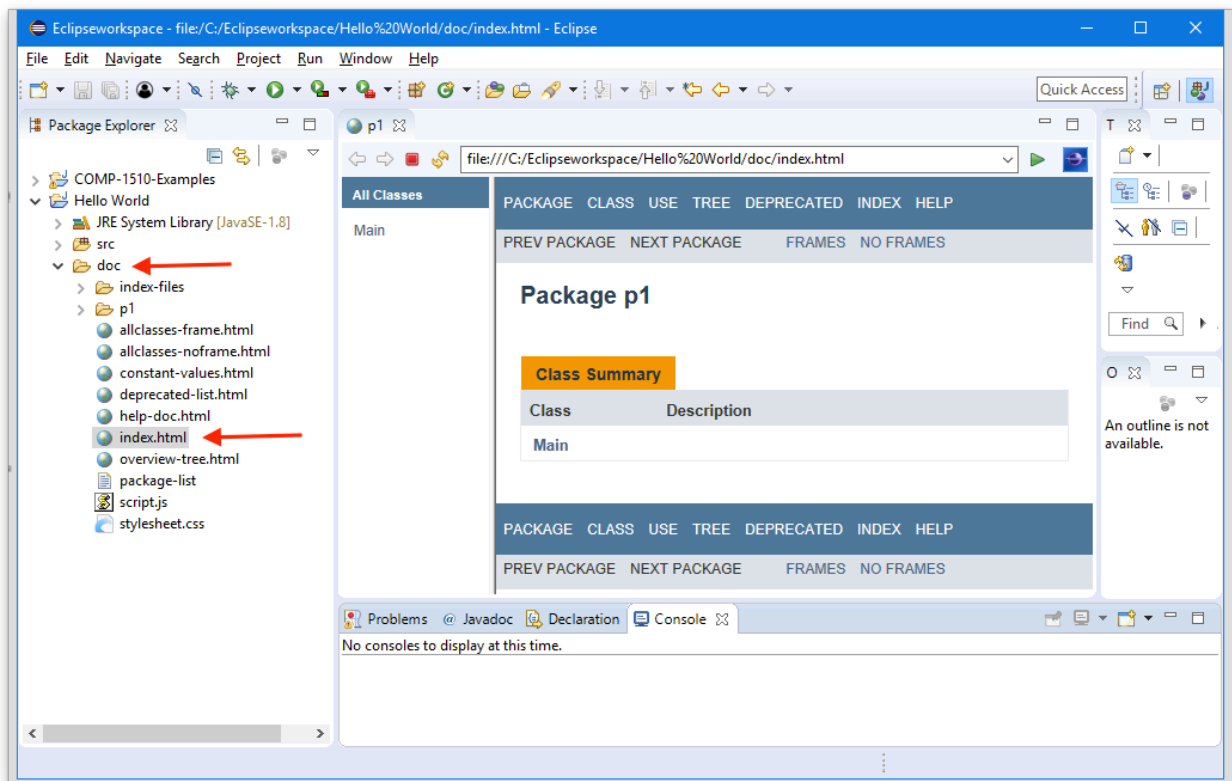


Figure 45: Javadoc for the Hello World project



## 17. Working from the Windows Command Line or macOS Terminal

Most people are not going to download Eclipse to run your program, so we need to learn how we can run Java programs without the IDE (Integrated Development Environment). This just means we need to learn about the Command Prompt (Terminal in macOS):

1. To open a Command Prompt in **Windows**, open the Start Menu and type Command in the search field. Windows will find Command Prompt. Click it to open it.
2. To open a **Terminal** in macOS, choose Applications > Utilities > Terminal.
3. The Command Prompt looks like this in Windows:

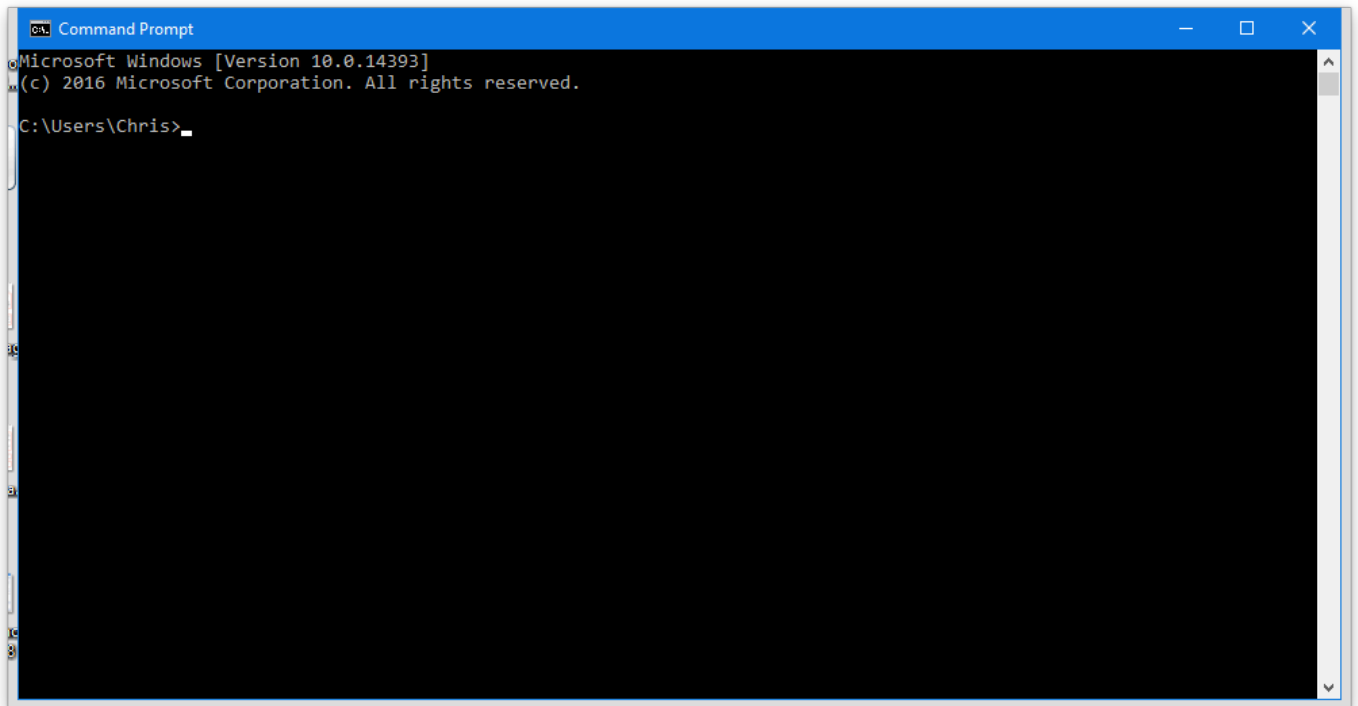


Figure 46: Windows Command Prompt

4. The Terminal in macOS looks like this:

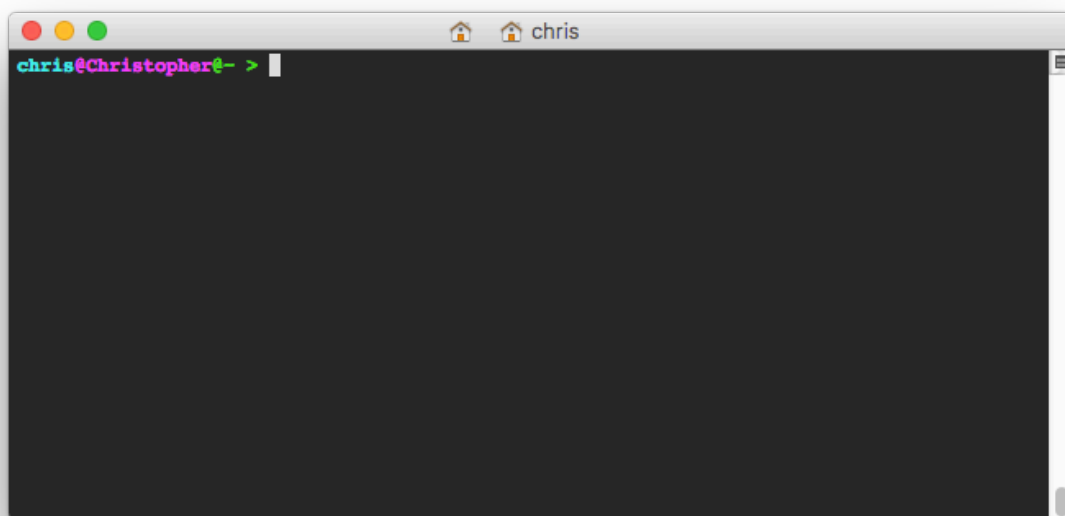


Figure 47: macOS Terminal window

5. It is a good idea to drag the icon for the Command Prompt (Windows) or Terminal (macOS) to your desktop or taskbar so you can access it quickly.
6. Compare the differences between using the Windows File Explorer (not to be confused with Internet Explorer) and the Command Prompt when looking at files. Windows 7, 8, and 10 each have slightly different looks for the File Explorer. You can open File Explorer by entering File Explorer in the search field beside the Start button:

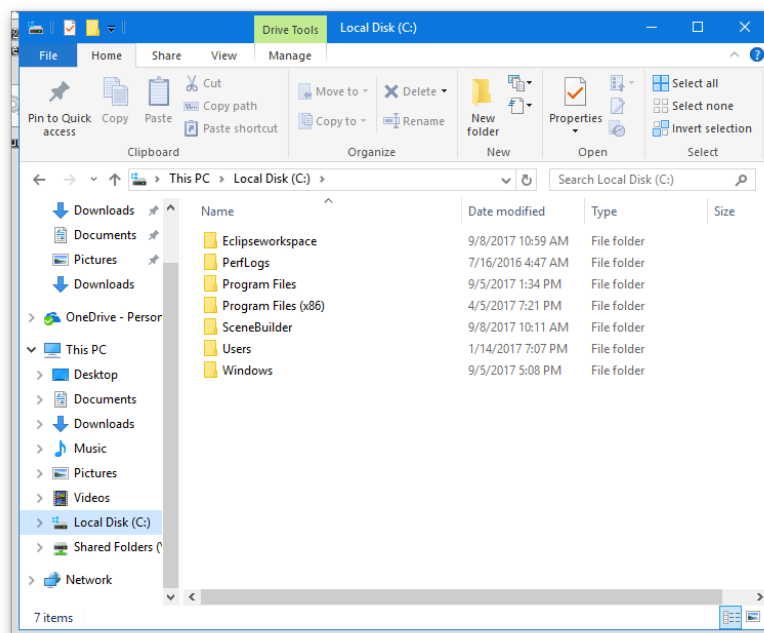


Figure 48: The C drive viewed in Explorer

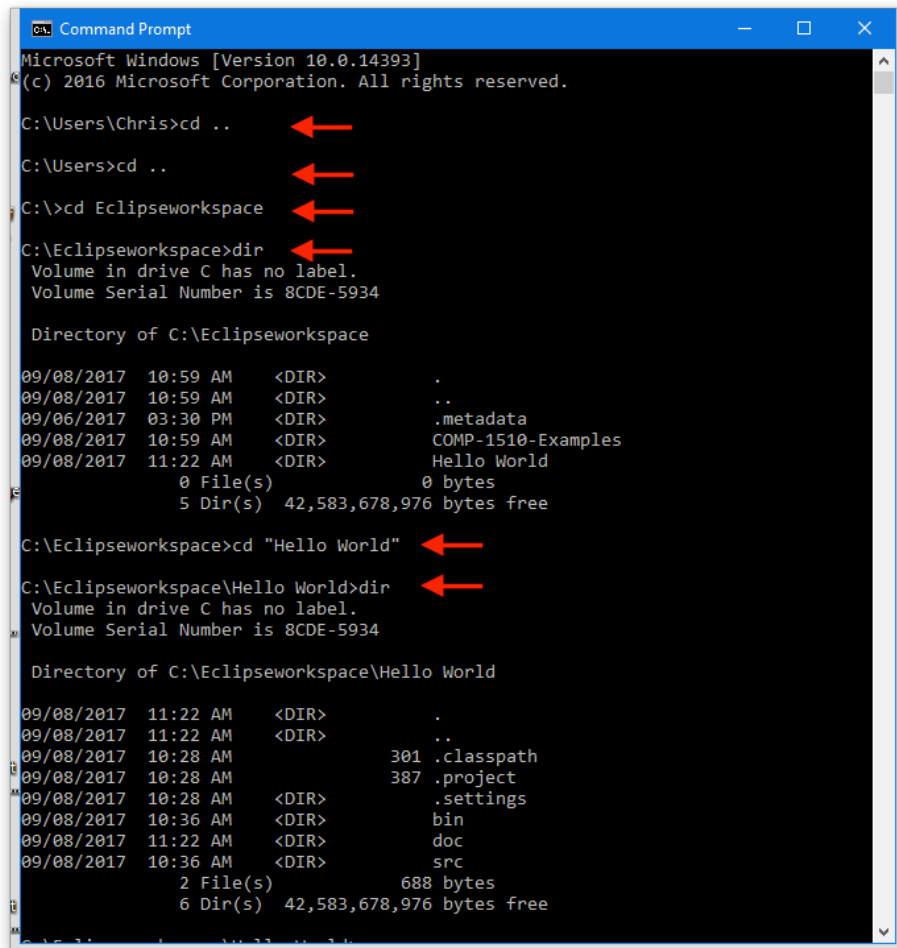
## Command Line Commands: CD and DIR (Windows) / LS (macOS)

The first command to learn is “CD” for Change Directory. (cd, cD, Cd, CD are all the same in the Command Prompt and Terminal).

To move into a directory simply type: `cd <directory path>` where <directory path> is where you want to go relative to the directory currently displayed in the Command Line/Terminal.

In our case we want to move to where we created the project in Eclipse. The directory we want is the Eclipse workspace directory (shown at Eclipse startup) followed by the project name.

7. Let’s navigate to our project folder. In my case it is in `C:\Eclipseworkspace\Hello World`. Do you remember where your workspace is?
8. Note the following when using the CD command:
  - a. To go “Up” or “back” a level, use `cd ..` (that’s cd followed by two periods).
  - b. Separate directories in a hierarchy using the `\` character in Windows or the `/` character in Linux/macOS.
  - c. Put quotes `""` around a directory name if a directory has spaces in the name.
  - d. Press the tab key after typing part of a directory or file name and Windows and macOS will attempt to complete the name for you and even put it in quotes if there is a space (this is so nice).
9. When you have fully entered a command, hit the enter key to execute the command:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Chris>cd ..
C:\Users>cd ..
C:\>cd Eclipseworkspace
C:\Eclipseworkspace>dir
Volume in drive C has no label.
Volume Serial Number is 8CDE-5934

Directory of C:\Eclipseworkspace

09/08/2017  10:59 AM  <DIR>          .
09/08/2017  10:59 AM  <DIR>          ..
09/06/2017  03:30 PM  <DIR>          .metadata
09/08/2017  10:59 AM  <DIR>          COMP-1510-Examples
09/08/2017  11:22 AM  <DIR>          Hello World
               0 File(s)              0 bytes
               5 Dir(s) 42,583,678,976 bytes free

C:\Eclipseworkspace>cd "Hello World"
C:\Eclipseworkspace\Hello World>dir
Volume in drive C has no label.
Volume Serial Number is 8CDE-5934

Directory of C:\Eclipseworkspace\Hello World

09/08/2017  11:22 AM  <DIR>          .
09/08/2017  11:22 AM  <DIR>          ..
09/08/2017  10:28 AM             301 .classpath
09/08/2017  10:28 AM             387 .project
09/08/2017  10:28 AM  <DIR>          .settings
09/08/2017  10:36 AM  <DIR>          bin
09/08/2017  11:22 AM  <DIR>          doc
09/08/2017  10:36 AM  <DIR>          src
               2 File(s)              688 bytes
               6 Dir(s) 42,583,678,976 bytes free
```

Figure 49: Use CD to navigate to the src folder

**Windows:** Another useful command is **DIR** (again, dir, diR, dIR, DIR, Dir, etc... are all the same in the Command Prompt). DIR gives you a “DIRectory listing” of files and directories are located in a given directory.

**macOS:** The macOS system uses **LS** (LS, IS, Ls, ls, etc...) instead of DIR to list the contents of a directory.

Use the command prompt to navigate the project folder and files that Eclipse created. Observe that Eclipse creates the following folders (directories) for each project:

1. bin: where compiled class files are stored
2. build: where the executable JAR file is created
3. src: where the source code files that we create are located
4. .settings: Eclipse project preferences

Eclipse also creates files for:

1. .classpath: controls where Eclipse looks for libraries and puts compiled files.
2. .project: the project configuration file.

## Compiling from the Command Line

Now it is time to compile the source code we wrote into Java bytecode. Eclipse does this for us automatically of course when we save our source files, but it's a good idea to learn how to do this manually.

To do this we have to run `javac` (the java compiler). Remember that this command is not part of Windows, it is part of the Java Software Developer Kit (Java SDK). The BCIT labs should be set up correctly to run `javac` on the Command Line, and after completing the first part of this lab your laptop, etc., should be set up too.

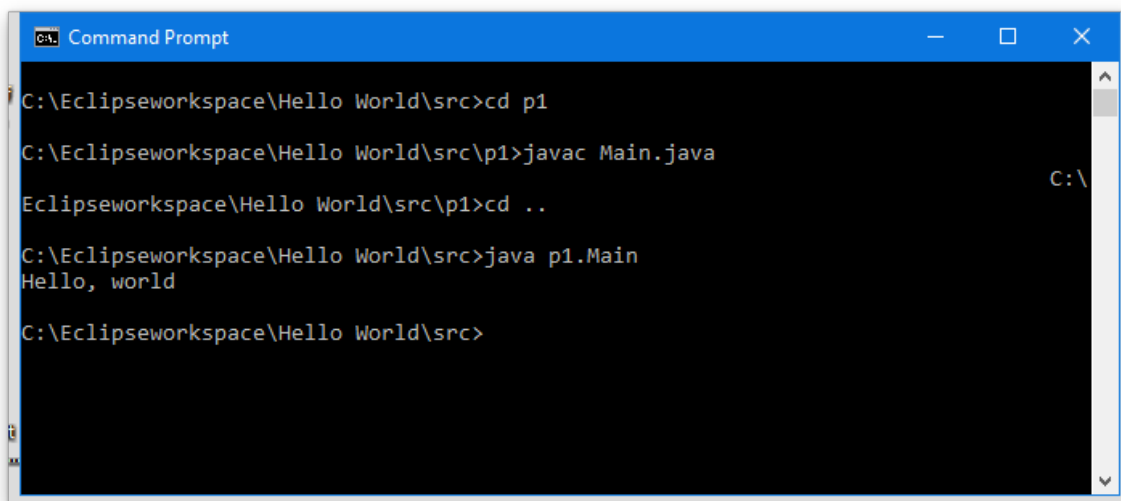
If the machine is not configured to run `javac` you will be presented with a message that tells you that `javac` could not be found.

---

*Aside: type `path` into the Command Prompt and press enter. This is your computer's PATH. Can you identify what's there? Window users: What happens if you type `%JAVA_HOME%` and press enter? What if you type `echo %PATH%`? macOS users: what happens if you type this into the Terminal: `/usr/libexec/java_home` and press enter?*

---

10. Ensure you are in the `src` folder. Remember we put our source file in a package called `p1`? Navigate to the `p1` package. It's a folder called `p1` inside `src`.
11. Compile the source code by entering `javac *.java` or `javac Main.java`. `*` is a wildcard, so `*.java` means compile all the `.java` files while `Main.java` means compile only the `Main.java`. Does it work? What is the output? If nothing is printed, that's okay! That means it compiled without any errors.
12. When a Java class in a package is compiled, it is placed in a directory with the package name. The full name of the compiled class includes the package name. In this case the full name of the class we just compiled is `p1.Main`.
13. To execute the program, we must return to the `src` folder and enter **`java p1.Main`** (no extension):



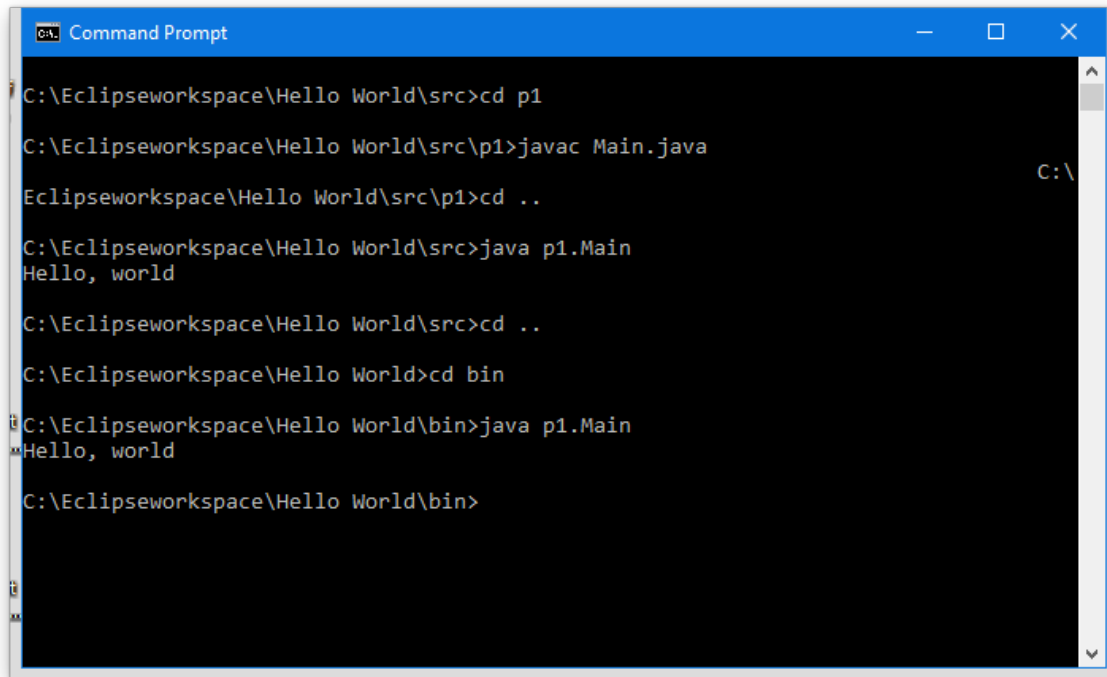
```
Command Prompt
C:\Eclipseworkspace\Hello World\src>cd p1
C:\Eclipseworkspace\Hello World\src\p1>javac Main.java
C:\Eclipseworkspace\Hello World\src\p1>cd ..
C:\Eclipseworkspace\Hello World\src>java p1.Main
Hello, world
C:\Eclipseworkspace\Hello World\src>
```

Figure 50: Compiling from the Command Prompt

Some important notes:

14. When we compile a java file, we need to tell the javac command the name of the file.  
**We use the entire file name including the extension .java with the javac command.**
15. When we run a java program, we need to give the java command the name of the *class*, not the file name. For this reason **we do not use a file extension with the java command.**

Note that we did not need to compile the Java program in this case because we wrote it using Eclipse. When we create a Java source file in Eclipse, Eclipse automatically compiles the class for us when we save it. We can cd to the bin directory and run the program directly:



```
C:\Eclipseworkspace\Hello World\src>cd p1
C:\Eclipseworkspace\Hello World\src\p1>javac Main.java
C:\Eclipseworkspace\Hello World\src\p1>cd ..
C:\Eclipseworkspace\Hello World\src>java p1.Main
Hello, world
C:\Eclipseworkspace\Hello World\src>cd ..
C:\Eclipseworkspace\Hello World>cd bin
C:\Eclipseworkspace\Hello World\bin>java p1.Main
Hello, world
C:\Eclipseworkspace\Hello World\bin>
```

Figure 51: Execute the class file Eclipse auto-compiled

## Additional Things to Know about the Command Prompt

When **changing directories** (CD) we can use .. (two periods) to go up one directory (above we were in the src directory and wanted to go to a bin directory at the same level).

Windows inherits the old DOS system of **drive letters** for mounted disk partitions. All of the above examples were on the C: partition. If you examine the computer in the lab (for example by using the Windows File Explorer) you will see several different partitions with different drive letters.

Each drive letter has a working directory. Using the CD command with a drive letter changes the working directory for that drive (with no drive letter it changes the working directory in the drive you are in). **To change to a different drive, type the drive letter, followed by a colon (:)** **at the Command Prompt.** Try to change to the D: drive, just for fun, and then change back to the C: drive.

The Command Prompt always tells you what drive letter you are currently using. Pay attention to this to avoid confusion, as several drives are used in the lab computers.

**New directories** can be created using the **md** (Make Directory) command in Windows and the **mkdir** command in macOS..

**Remove directories** using the **rd** (Remove Directory) command in Windows and the **rm -r** command in macOS. As with the CD command, directories can be specified in either an absolute or relative manner.

**If you need to delete files, use the DEL command in Windows and the rm command in macOS.**

## Redirection

The output of commands sometimes scrolls by, leaving us wondering what happened. The command prompt offers several solutions to help us deal with this problem.

The first of these is the pipe (|) operator. Using the pipe operator, we can take the output from one command and send it as input into another command. For example, the command:

```
C:\work> java Lincoln | more
```

takes the output produced and feeds it into the more command. **The more command simply displays the input to the screen, pausing after each screen.** This provides us a way to view output one page at a time (not critical in this case). This works for Windows and macOS.

In addition to the pipe operator, we also have the **output redirection** operators > and >>. **The output redirection operator takes our output and stores it in the specified file.** For example, the command:

```
C:\work> java Lincoln > out.txt
```

takes the output of the command and saves it in the out.txt file (in the current directory). In this case, if the out.txt file already exists, it is overwritten. The >> operator differs in that it will append to an already existing file, rather than replace it.

## 18. You Have Completed Lab 0

That's it! Take a break. When you've recovered, open the COMP 1510 Lab 1 document and complete the lab exercises. Ask your lab instructor any questions. Good luck, and have fun!