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) by setting up your laptop properly.

The first part of this lab shows you how to set up your laptop (and home computer). Doing this correctly is critical for your success in this and other courses.

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.

If you are using a PC with Microsoft Windows 10, you can get software via http://appsanywhere.bcit.ca as well. These instructions cover the traditional way of installing the software (you will need to know how to do this anyway).

TL; DR – Oracle Java JDK 17.0.5 JDK + Hotspot JVM, Eclipse 2022-12 with Checkstyle plugin 10.4.0, Apache ant 1.10.12

What will you DO in this lab?

In this lab, you will:

1. Understand basic but critical aspects of working with a computer and software development tools.
2. Install the Java Software Development Kit on your personal computer.
3. Set environment variables to add Java to the PATH.
4. Install the following software on your *personal* computer:
    a. Eclipse IDE (Integrated Development Environment)
    b. add the Checkstyle plugin for Eclipse
5. Install Ant and add it to the PATH.
6. Download the Java Examples for the course from the learning hub and load it into Eclipse.
7. Use Eclipse to:
    a. create a Java project using the sample software
    b. test that Eclipse is set up properly by running a basic Java program
    c. create a simple Java project and execute it,
8. Use the command line to:
    a. navigate to your source code
    b. compile and execute Java source code
    c. run Ant

Table of Contents

1. Overview (READ THIS FIRST)

To ensure success in the CST program you must master the use of your personal laptop computer. This includes knowing how to use the operating system, command line, and general maintenance practices as well as using the tools required for the courses. If your computer gets corrupted, you will have difficulties completing the assignments on time and add unnecessary stress to your life (we will add enough stress for you).

If you bought a used computer *be sure to clean out the previous, possibly harmful, software*. It is best to install a clean version of the operating system, then install all of your software. If you cannot install something because you do not own it, get rid of it. Throw it totally away. You will find out that some things that seem “free” have a serious cost. At least delete all of the applications that you do not use, any games you did not install. Anything you do not know about could hurt you and make programs that you need fail in mysterious ways.

Learn the critical folders/directories that are being used by the operating system, or the tools you will use. *Do not unzip any files into these locations*! For example, on Windows do not unzip anything into the “Program Files” directory. Do not unzip anything into the Eclipse workspace.

Know which folders are automatically backed up into the Cloud and *avoid these for software development*. Cloud backup schemes (such as Apple iCloud and MS OneDrive) are *not* designed for the rapid changes involved in software development (you will learn better tools to handle this, such as git), and the cloud backup can cause failures with development tools.

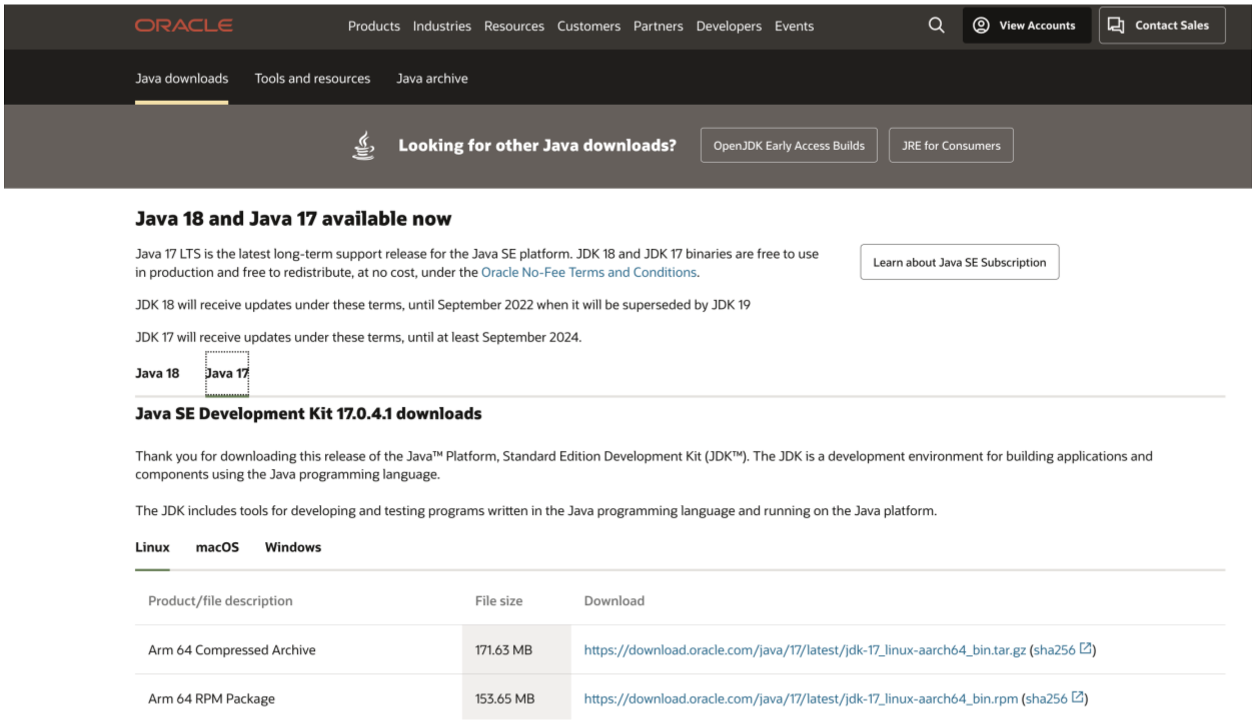
Remember the steps you take to solve problems. If something does not work, retrace your steps to find what went wrong. Retreat to a place that works and try again. If you cannot recall what you are doing, take notes. Often to help solve a difficulty, you will be asked what you did to get to the problematic situation.

Know all of the software installed on your computer. Until you are more experienced, ***have only one version of Java installed***. Know which tools use ports (examples would be web servers and databases) and what ports they use.

2. Install Java

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

1. Open a browser window and navigate to https://www.oracle.com/java/technologies/downloads/ Click on the Java 17 tab.
2. We want to install a stable version of the Java SE Development. Since September, 2021, the stable version was version 17 and we will use 17.0.5.
3. Note: Java has moved to a new development cycle with frequent updates. The stable supported updates change less often. The current version 17 is a stable, long term support version. This course is using Java 17 from the Oracle site.
4. Click the download link for the Oracle 17 JDK *for your operating system* from the download panel:


ORACLE
Products Industries Resources Customers Partners Developers Events
View Accounts
Contact Sales
Java downloads
Tools and resources
Java archive
Looking for other Java downloads?
OpenJDK Early Access Builds
JRE for Consumers
Learn about Java SE Subscription
Java 18
Java 17
Linux
macOS
Windows

Java 18 and Java 17 available now

Java 17 LTS is the latest long-term support release for the Java SE platform. JDK 18 and JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions.

JDK 18 will receive updates under these terms, until September 2022 when it will be superseded by JDK 19

JDK 17 will receive updates under these terms, until at least September 2024.

Java SE Development Kit 17.0.4.1 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

Product/file description	File size	Download
Arm 64 Compressed Archive	171.63 MB	https://download.oracle.com/java/17/latest/jdk-17_linux-aarch64_bin.tar.gz (sha256)
Arm 64 RPM Package	153.65 MB	https://download.oracle.com/java/17/latest/jdk-17_linux-aarch64_bin.rpm (sha256)

.

5. Download the JDK installation file (I used the MSI installer on Windows). It's okay to save it in your Downloads folder. When you are done installing Java, you can 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.
7. Wait until the installation finishes. This may take a few minutes.
8. You can delete the SDK installation file now. You don't need it anymore.
9. **macOS only**:
    a. These instructions all assume you are using `bash`. If you have switched to `zsh`, replace `.bash_profile` with `.zshrc`
    b. When the installer finishes, you should verify your installation. Open a Terminal window and enter the command `java -version`. The output should start as follows:

    ```
    java version "17.0.5" 2022-10-18 LTS
    ```

    What happens if you enter the command `java`, or the command `javac`?

    c. After you verify your installation, you need to define a JAVA_HOME environment variable. There are many ways to do this, one way is to create/edit the .bash_profile file in your home directory with the following contents added (if you do not have a preferred command-line editor, use the command `nano ~/.bash_profile`):

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-17.0.5.jdk/Contents/Home

export PATH=$JAVA_HOME/bin:$PATH
```

    d. To test this worked, open a new terminal window and enter the command `echo $JAVA_HOME`. You should see what is on the right side of the assignment in the box. A second test is that the command `ls $JAVA_HOME/bin` should show a bunch of files.

e. Whenever something needs to be defined as an environment variable or added to the path, follow a similar procedure.

f. Mac users may skip ahead to section 5.

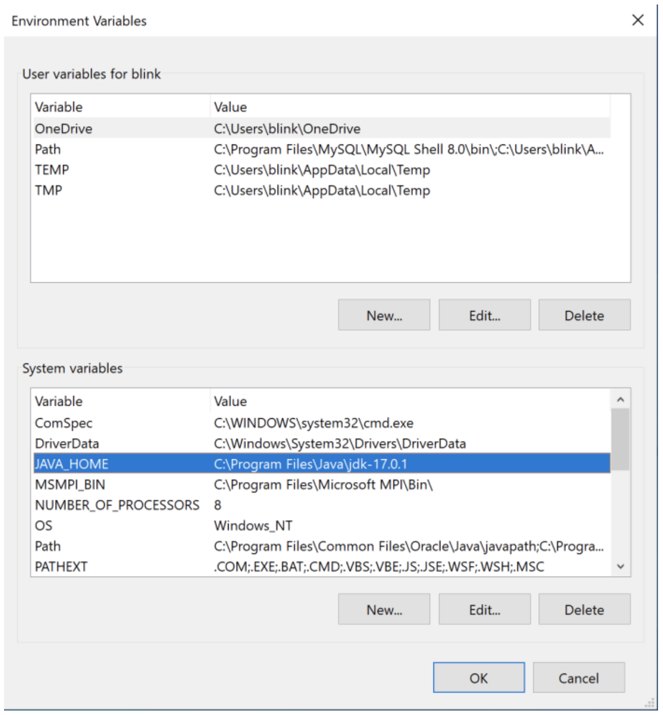
3. **WINDOWS ONLY**: Define an Environment Variable for Java

To compile Java programs in the Windows command line, we need the Java compiler to be in the Windows **PATH**.

The JDK installation should have already added Java to the PATH (and maybe set the JAVA_HOME variable), which you can verify in the following steps. In any case, you need to know how to manually add environment variables.

These instructions are for Windows 10 x64:

In the search window of the toolbar, type env and select “Edit the system environment variables” and click on the Environment Variables button:



User variables for blink

Variable	Value
OneDrive	C:\Users\blink\OneDrive
Path	C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\blink\A...
TEMP	C:\Users\blink\AppData\Local\Temp
TMP	C:\Users\blink\AppData\Local\Temp

System variables

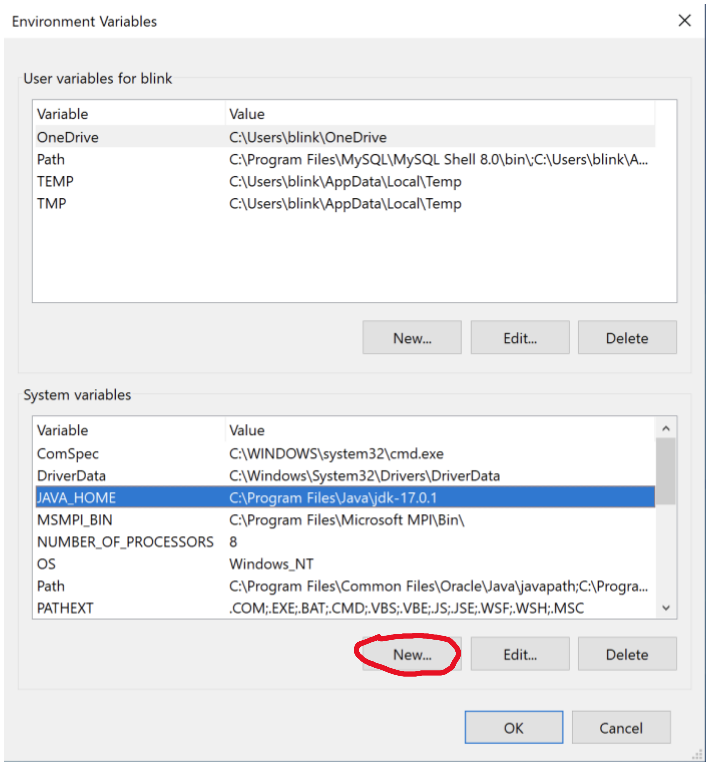
Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Program Files\Java\jdk-17.0.1
MSMPI_BIN	C:\Program Files\Microsoft MPI\Bin\
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Progra...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

Figure 1: Windows Environment Variables.

1. Note that every variable has a name and a value.
2. 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 the users on the computer.

   On your personal computer, you want to change the System variables.
3. 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., C:\Program Files\Java\jdk-17.0.5\ (use the complete actual path name).

4. If there is no variable that defines JAVA_HOME, click the New button under System variables:



Environment Variables

User variables for blink

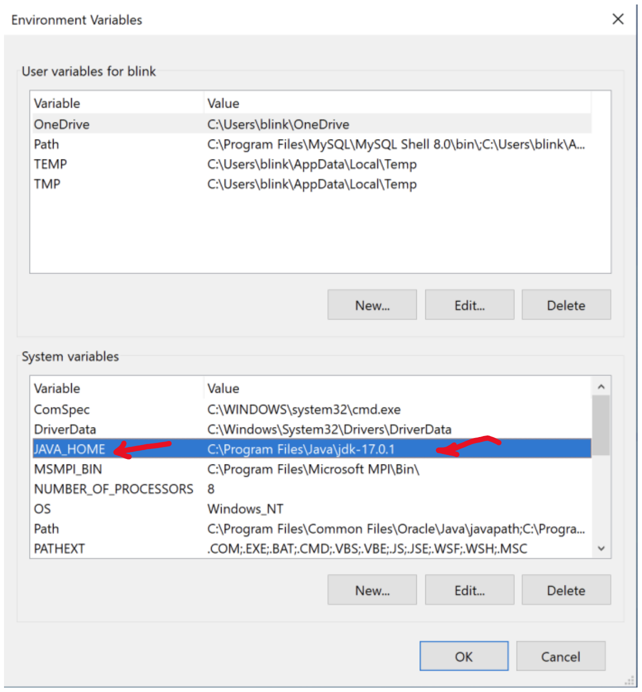
Variable	Value
OneDrive	C:\Users\blink\OneDrive
Path	C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\blink\A...
TEMP	C:\Users\blink\AppData\Local\Temp
TMP	C:\Users\blink\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Program Files\Java\jdk-17.0.1
MSMPI_BIN	C:\Program Files\Microsoft MPI\Bin\
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Progra...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

Figure 2: Add a new user variable by selecting New...

5. Create a new environment variable called JAVA_HOME, and choose Browse Directory...
6. Navigate to the location of the jdk17.0.5 installation.
7. Click OK. You should return to the New User Variable window
8. Click OK to create the new user environment variable.
9. Your User variables should contain a new Variable called JAVA_HOME with the path to the correct installation folder as its Value:



Environment Variables

User variables for blink

Variable	Value
OneDrive	C:\Users\blink\OneDrive
Path	C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\blink\A...
TEMP	C:\Users\blink\AppData\Local\Temp
TMP	C:\Users\blink\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Program Files\Java\jdk-17.0.1
MSMPI_BIN	C:\Program Files\Microsoft MPI\Bin\
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Progra...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

Figure 3: JAVA_HOME created!

4. WINDOWS ONLY: add %JAVA_HOME%\bin to the Windows Path

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\jdk-17.0.5\bin.

We now want to change the Windows Path environment variable so that it includes the directory that contains javac (the Java compiler) and the other command line tools that were installed with the JDK. If the JDK installation instructions above were followed, this has already been done (Oracle uses …\Oracle\Java\javapath to do this).

Only if necessary, manually add the %JAVA_HOME%\bin folder to the Windows Path environment variable:

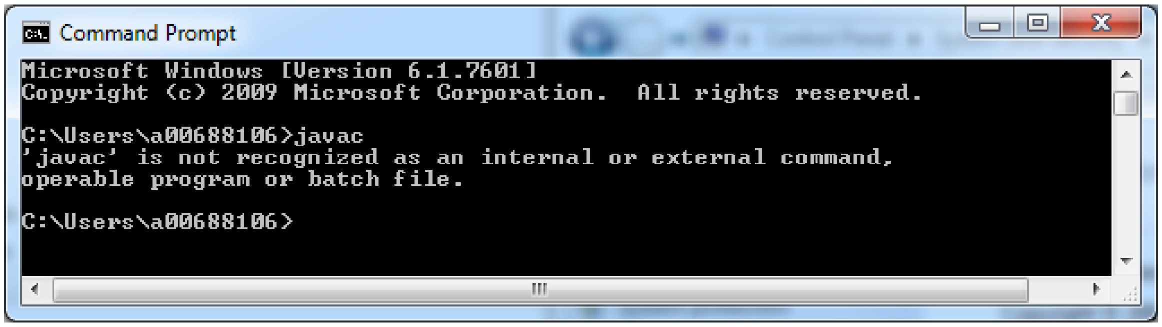
1. The first rule of editing the Path: **do not delete anything already in it**. If there are existing values in the Path, do not erase them. We will prepend (add) our environment variable to the beginning of the Path.
2. Select the Path variable in the User variables, and click Edit:
3. Move your cursor 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/top* of the Path.
5. 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 the operating system (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 (which we updated) *plus* whatever is in the User PATH.

For example, if the System Path is set to %JAVA_HOME%\bin;C:\temp the Windows Path should start with this (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:



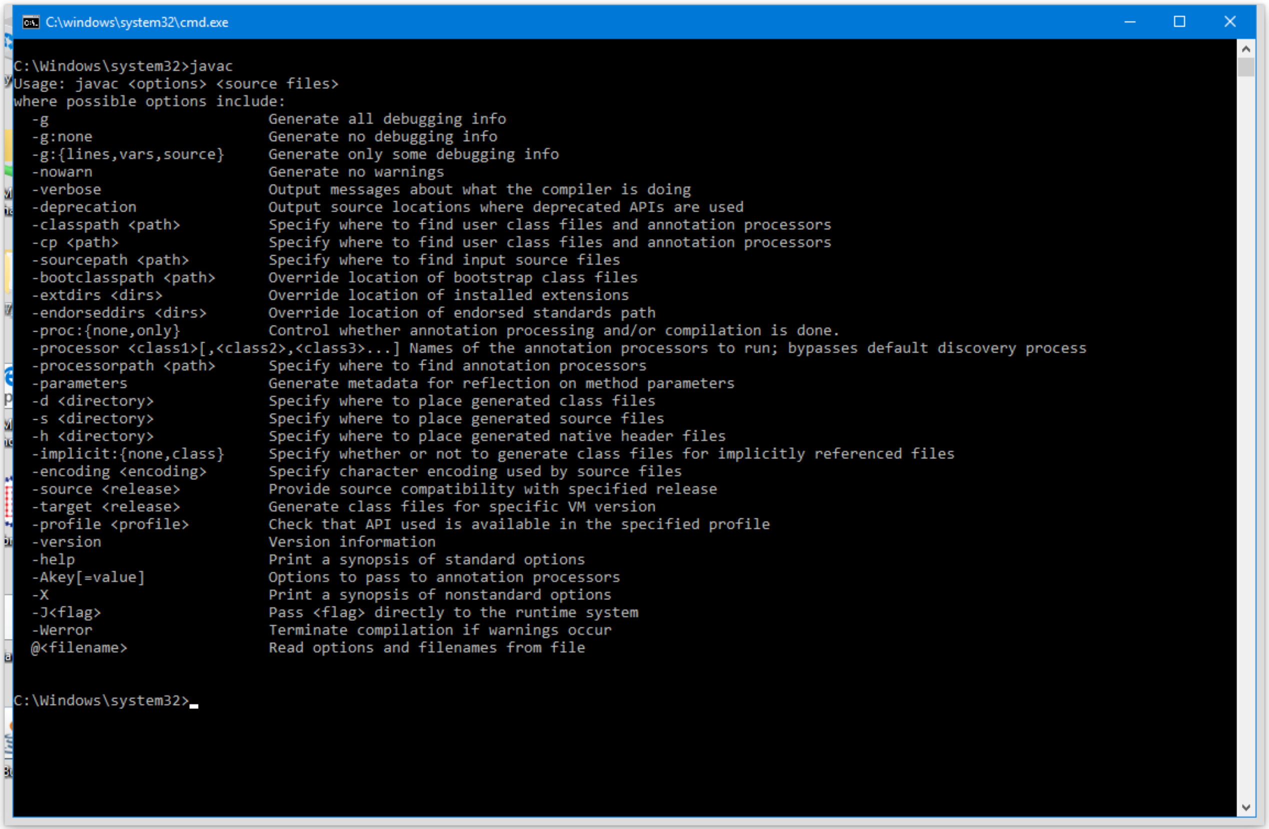
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\a00688106>javac
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\a00688106>
```

Figure 4: javac is not recognized...

When we type javac the Command Prompt in our example looks for javac.exe in each directory and does not find 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*).

1. Open a Command Prompt window and enter the javac command.
2. 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 to include the new variable. You should see something like this. This is the javac program telling you how to use itself:



```
C:\Windows\system32>javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are used
  -classpath <path>          Specify where to find user class files and annotation processors
  -cp <path>                 Specify where to find user class files and annotation processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>      Specify where to find annotation processors
  -parameters                Generate metadata for reflection on method parameters
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -h <directory>             Specify where to place generated native header files
  -implicit:{none,class}     Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release
  -target <release>          Generate class files for specific VM version
  -profile <profile>         Check that API used is available in the specified profile
  -version                   Version information
  -help                      Print a synopsis of standard options
  -Akey[=value]              Options to pass to annotation processors
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system
  -Werror                    Terminate compilation if warnings occur
  @<filename>                Read options and filenames from file


C:\Windows\system32>_
```

Figure 5: javac works hooray

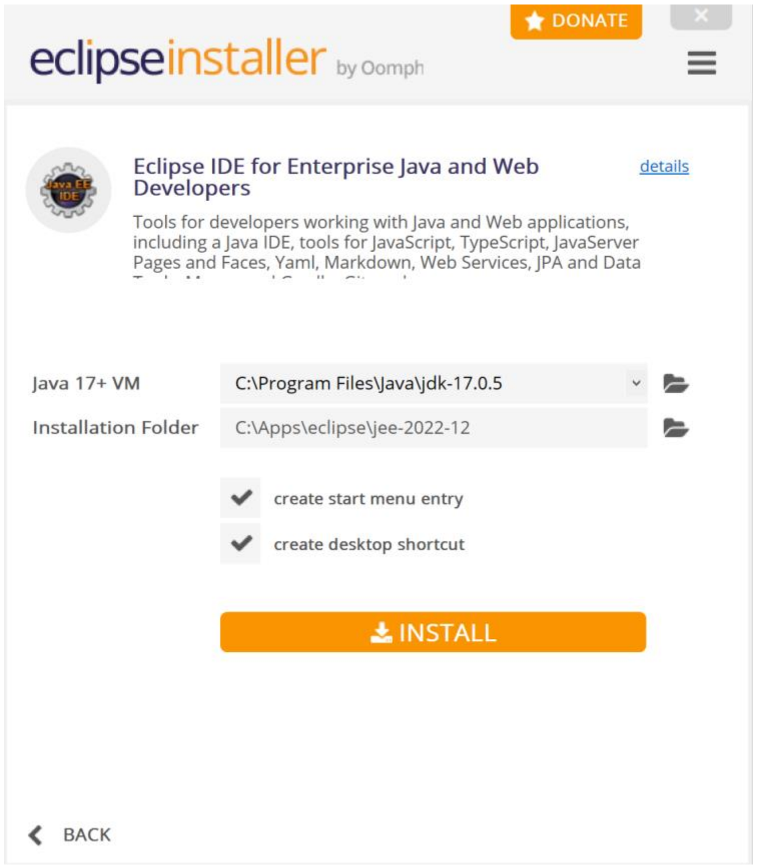
We're going to return to the Command Prompt later in this lab. Right now, let's install the rest of our tools.

5. Install the Eclipse IDE

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 Eclipse 2022-12:

1. Navigate to https://www.eclipse.org/downloads/
2. Chick the download button for Eclipse 2022-06 and run the program you downloaded.to run the eclipse installer.
3. Click the "Eclipse IDE for Enterprise Java and Web Developers "(64 bit version to match the JDK)
   a. In the install screen in the line Java 17+ VM select the Java 17 SDK that you installed above (see following image):



Eclipse IDE for Enterprise Java and Web Developers

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data

Java 17+ VM: C:\Program Files\Java\jdk-17.0.5

Installation Folder: C:\Apps\eclipse\jee-2022-12

✓ create start menu entry

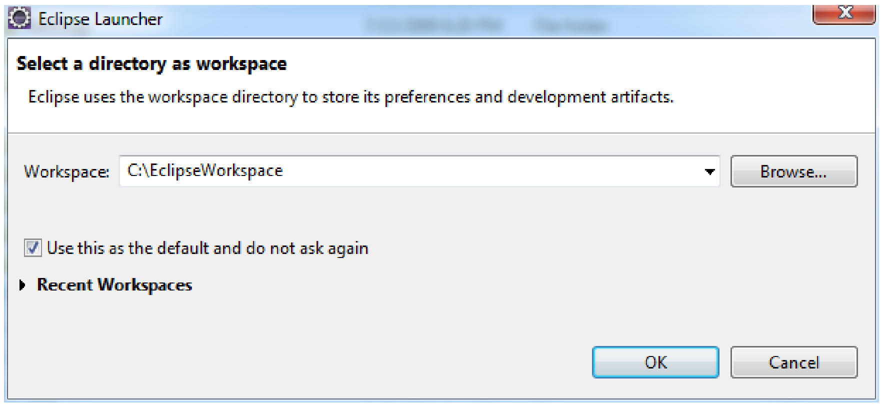
✓ create desktop shortcut

4. The installation will take a few moments.
   a. Click OK if you get a warning about unsigned content.

Eclipse is one of the most popular Integrated Development Environments (IDEs) used in industry. Other common IDEs are IntelliJ IDEA, and Microsoft Visual Code (which is for C# development, but also used for Java and many other languages). Your subsequent courses may want you to use a specific 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.

6. Launch Eclipse

1. Once you have installed Eclipse 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 (suitable for personal computers), the workspace is at C:\EclipseWorkspace. (avoid your User area if it might be shadowed to the cloud).


Eclipse Launcher

Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\EclipseWorkspace

Browse...

☑ Use this as the default and do not ask again

Recent Workspaces

OK Cancel

Figure 6: Set the Eclipse workspace directory

3. Once you have selected your Workspace, Eclipse finishes loading and displays the Welcome window.
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 primarily 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 like this (there is a lot of white space):

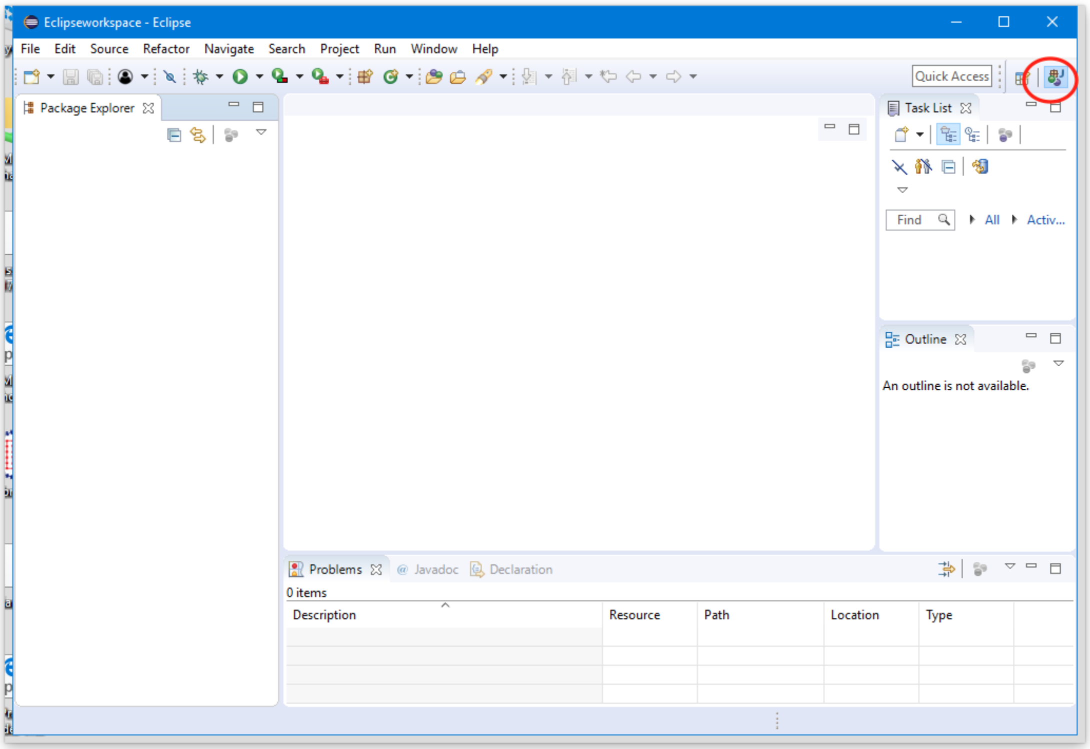


Figure 7: Fresh Eclipse installation

7. Install the Checkstyle Plugin

One of the benefits of 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.

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.
2. Click on the Install button for the **Checkstyle Plug-in 10.4.0** 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. Install Apache Ant

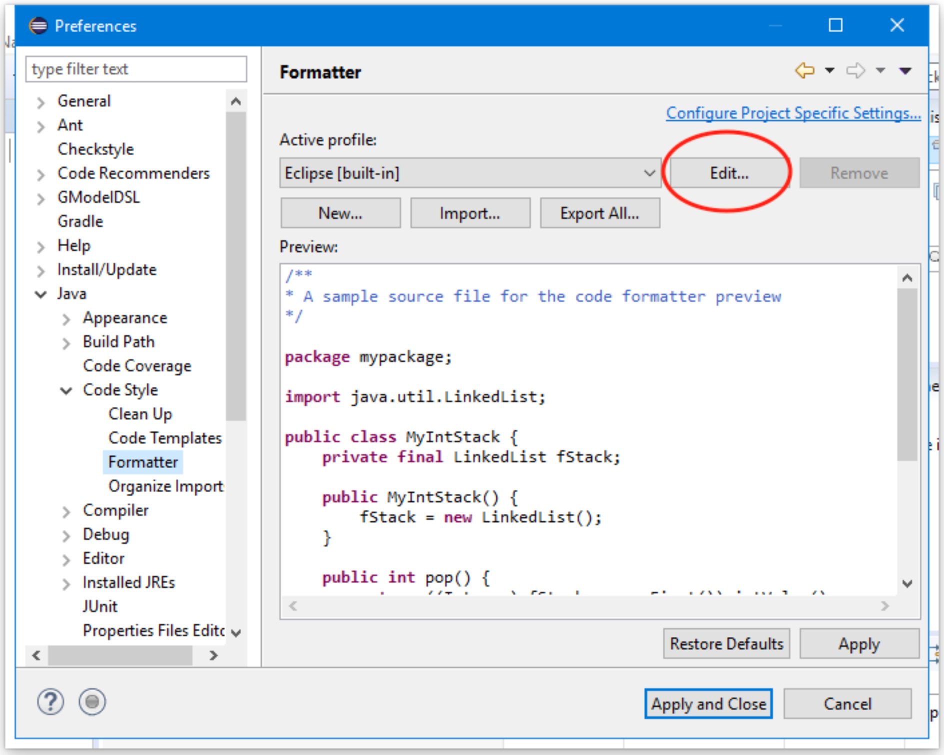
The Apache Ant tool will be used in assignments. It is quick to install, so do so now.

a. You can find the proper site by googling "Apache ant", but the link is https://ant.apache.org/.
b. Click on the left side on Download>Binary Distributions and scroll down to "Current Release of Ant"
c. Click on the link to download a current 1.10.x release for your operating system (current version is 1.10.12 when this is written). The zip format will work for all operating systems.
d. Unzip the downloaded file into the Apps directory you set up earlier.
e. Make sure there are no extraneous directories, you should see Apps>apache-ant-1.10.12>bin (where > is the directory separator, "/" for Mac, "\" for Windows)
f. Define ANT_HOME to be the full directory path to the folder which contains the bin directory (such as `c:\Apps\apache-ant-1.10.12`) as described in section 3.
g. Now we need to add the ANT_HOME bin folder to the PATH environment variable. You can do this as described in section 4 (prepend `%ANT_HOME%\bin` for Windows, `$ANT_HOME/bin` for Mac)
h. Verify the installation by opening a command window and typing the command `ant`. If you get a message about build.xml does not exist, then it is properly installed.

9. Configure Eclipse

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

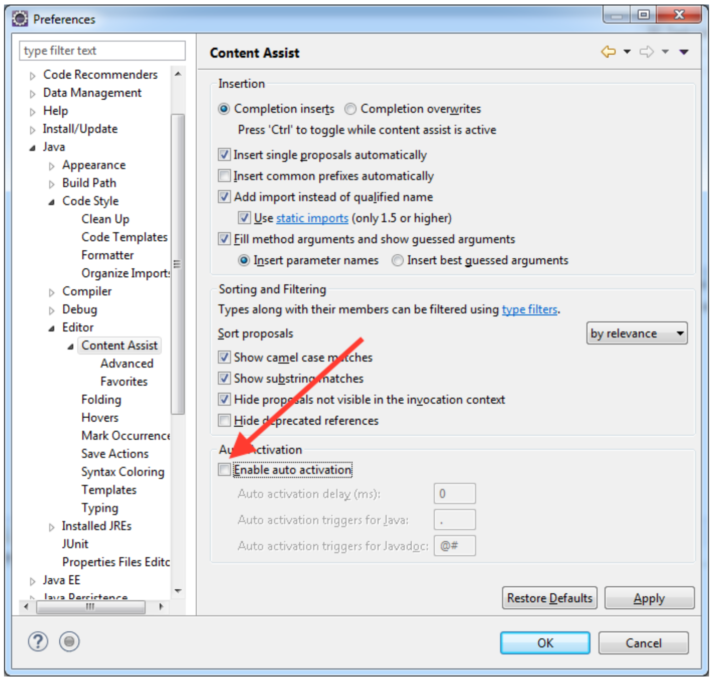
1. From the Menu, select Window > Preferences (Eclipse > Settings on the Mac) and expand Java/Code Style/Formatter:



```
/**
* A sample source file for the code formatter preview
*/

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
    private final LinkedList fStack;

    public MyIntStack() {
        fStack = new LinkedList();
    }

    public int pop() {
```

Figure 8: Window > Preferences > Java > Code Style > Formatter > Edit

2. Create a profile named **Standard** 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.
5. Click OK to accept the new profile.
6. 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:



Content Assist

Insertion

- [✓ Completion inserts] Completion overwrites
- Press 'Ctrl' to toggle while content assist is active
- [✓] Insert single proposals automatically
- [ ] Insert common prefixes automatically
- [✓] Add import instead of qualified name
  - [✓] Use static imports (only 1.5 or higher)
- [✓] Fill method arguments and show guessed arguments
  - [✓ Insert parameter names] Insert best guessed arguments

Sorting and Filtering

Types along with their members can be filtered using type filters.

Sort proposals: by relevance

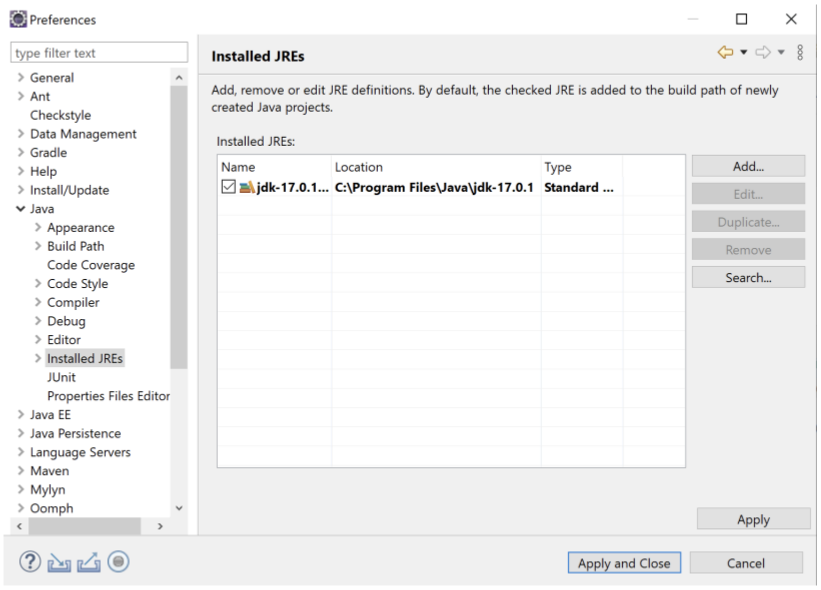
- [✓] Show camel case matches
- [✓] Show substring matches
- [✓] Hide proposals not visible in the invocation context
- [ ] Hide deprecated references

Auto Activation

- [ ] Enable auto activation
- Auto activation delay (ms): 0
- Auto activation triggers for Java: .
- Auto activation triggers for Javadoc: @#

Figure 9: Disable code assist auto activation

7. Click OK in the Preferences window to accept the changed preferences and close the window. **<u>Content assist is still available by hitting control-space</u>** 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.
9. Still in Eclipse Preferences, check the installed JREs and ensure that your Oracle JDK is the default and only compiler showing (If eclipse added another one, add your Oracle JDK, make it the default, and remove the one Eclipse installed):



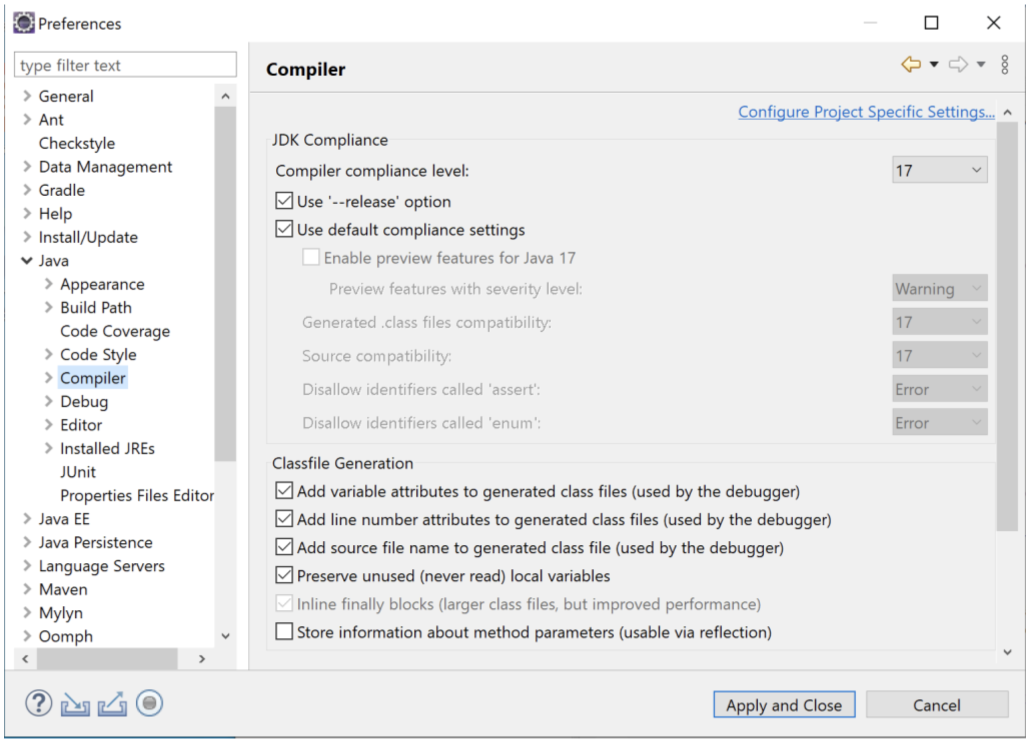
Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type
✓ jdk-17.0.1...	C:\Program Files\Java\jdk-17.0.1	Standard ...

10. Click on the Java>Compiler header as shown and set the JDK Compiler compliance level to 17 (if it is different):



Compiler

Configure Project Specific Settings...

JDK Compliance

Compiler compliance level: 17

☑ Use '--release' option

☑ Use default compliance settings

☐ Enable preview features for Java 17

Preview features with severity level: Warning

Generated .class files compatibility: 17

Source compatibility: 17

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

☑ Inline finally blocks (larger class files, but improved performance)

☐ Store information about method parameters (usable via reflection)

10. Create an Eclipse Project

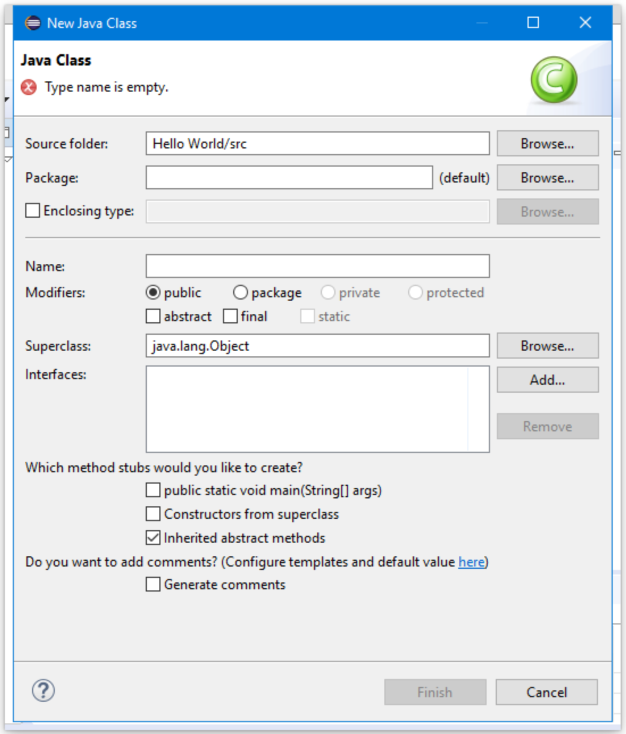
Eclipse requires that you create a project for your source files. A project is simply a collection of files and settings for a program or, in our case, an assignment or lab.

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 several default settings that a new Java project has. We will start by providing a project name, `Hello World`. Use meaningful names for your projects that will allow you to remember what you are doing.
3. Go ahead and click Finish. For the module name prompt, enter `helloworld`. In the leftmost pane of the workspace you should see your new Java project appear.

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.
2. The **JRE System Library folder** links your project to the Java 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:


New Java Class

Java Class

Type name is empty.

Source folder: Hello World/src

Package: (default)

☐ Enclosing type:

Name:

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

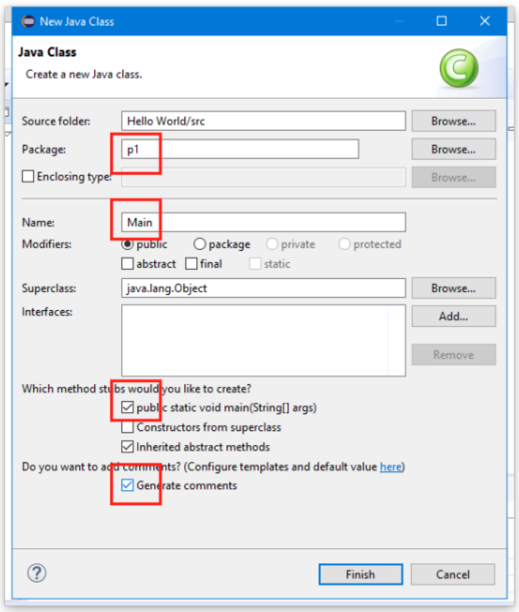
☑ Inherited abstract methods

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

☐ Generate comments

Figure 10: 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:



New Java Class

Java Class

Create a new Java class.

Source folder: Hello World/src

Package: p1

☐ Enclosing type:

Name: Main

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

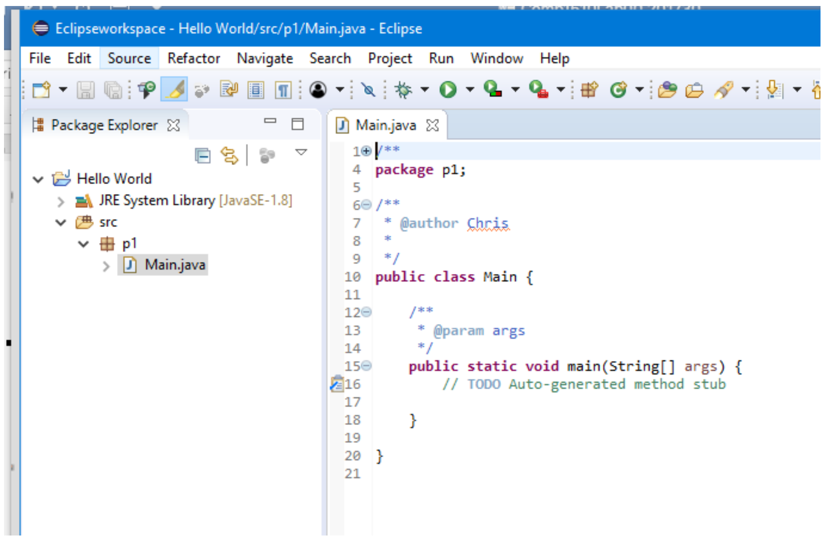
☑ Inherited abstract methods

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

☑ Generate comments

Figure 11: 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:



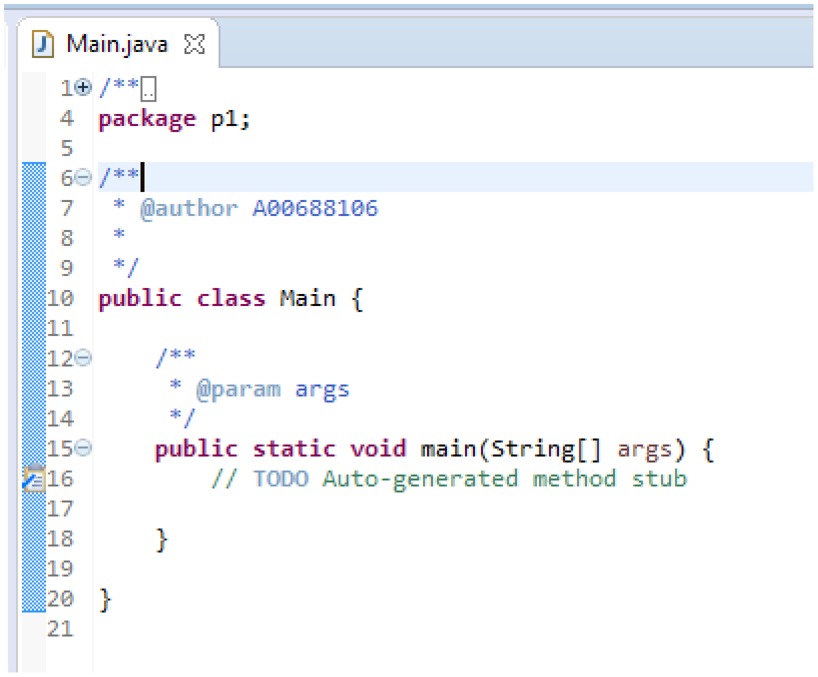
Eclipseworkspace - Hello World/src/p1/Main.java - Eclipse

Hello World
JRE System Library [JavaSE-1.8]
src
p1
Main.java

```
/**
package p1;

/**
 * @author Chris
 *
 */
public class Main {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub

	}

}
```

Figure 12: 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. In Java class names always begin with a capital letter.
12. **NOTE: Never modify source files or move them around unless you use Eclipse. If you try to modify project files outside of Eclipse, you may corrupt the project. Never use the file system to save files *inside* of your Eclipse workspace or unzip files inside of the Eclipse workspace!**
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:



```
/**
package p1;

/**
 * @author A00688106
 *
 */
public class Main {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Figure 13: Main.java source file

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 and is the one used in this course.
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.

12. Compile and Execute a Java Project

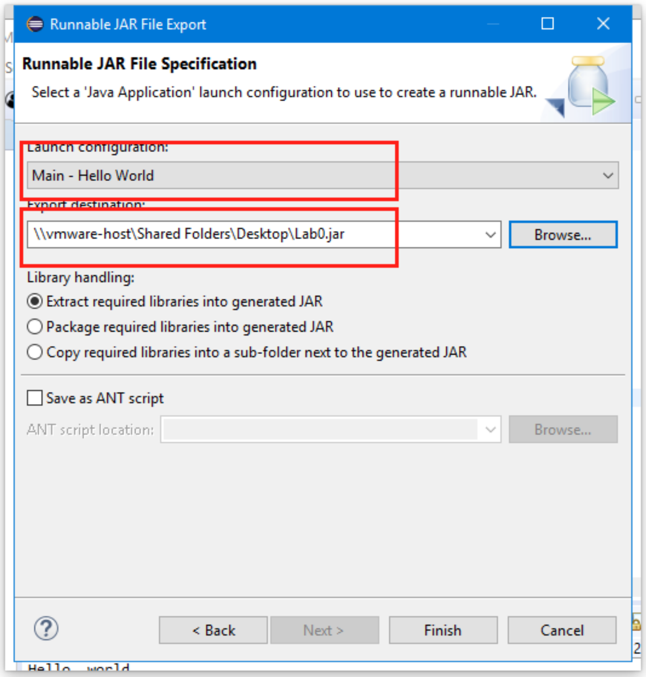
1. Each Java program must first be compiled from source code into bytecode. The bytecode is then executed by the Java Virtual Machine.
2. Eclipse automatically compiles (builds) your program when you save it. If you want to recompile (rebuild) your project from scratch, select Project > Clean from the 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!
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:
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):



Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: Main - Hello World

Export destination: \\vmware-host\Shared Folders\Desktop\Lab0.jar

Library handling:
- (●) Extract required libraries into generated JAR
- ( ) Package required libraries into generated JAR
- ( ) Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location:

Figure 14: Runnable JAR settings

6. 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

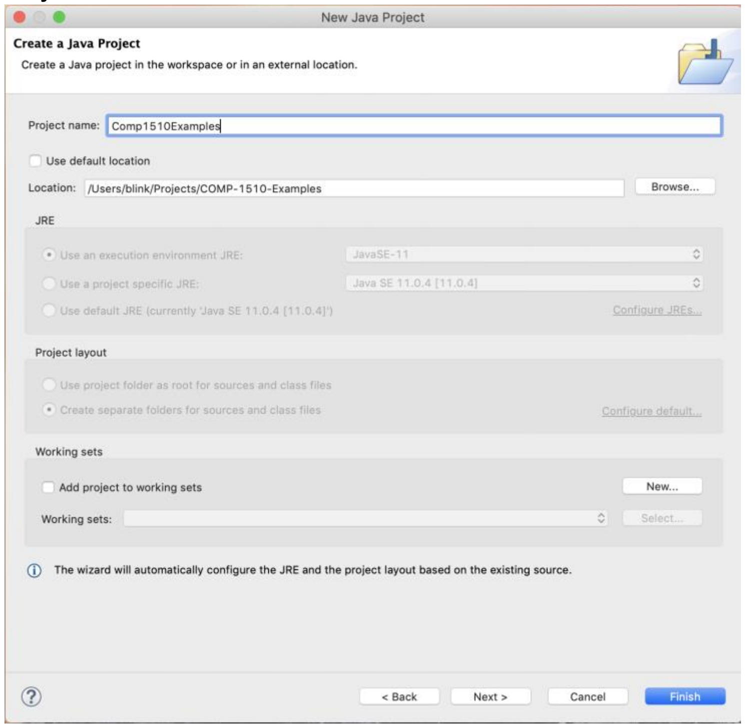
   java -jar Lab0.jar (try it!).

14. Import and Run Examples from the Text

The next thing you should do is add in the Comp 1510 Examples into your Eclipse workspace. To do this, download the file comp-1510-examples-master.zip from D2L Content>Resources section, unzip the file in a projects folder (create and then use a C:\Projects or ~/Projects folder – DO NOT use your eclipse workspace) and ensure there are no extra directories (unzipping on Windows often produces an extra directory).

Import the project into Eclipse as described in the following steps:

1. Select File > New > Java Project (you may need to select Other to find Java Project) and click Next.
2. Enter the project name (say Comp1510Examples). ***UNCLICK*** "Use default location". For location, click Browse and select the folder containing the unzipped contents from the comp-1510-examples-master.zip file. The COMP-1510Examples checkbox should be selected under Projects:



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: Comp1510Examples

Use default location

Location: /Users/blink/Projects/COMP-1510-Examples

JRE

Use an execution environment JRE: JavaSE-11

Use a project specific JRE: Java SE 11.0.4 [11.0.4]

Use default JRE (currently 'Java SE 11.0.4 [11.0.4]')

Project layout

Use project folder as root for sources and class files

Create separate folders for sources and class files

Working sets

Add project to working sets

Working sets:

The wizard will automatically configure the JRE and the project layout based on the existing source.

Figure 15: Importing the text project

3. Click on Finish. If you get a message about overwriting a module-info.java file, select **No**. If you get a message about opening a Java Perspective, select Open Perspective.
4. Right click on the new project, select Build Path>Configure Build Path… and click on the Library tab. With Modulepath highlighted, check Eclipse picked up Junit5, if not click Add Library, highlight Junit, click next, select Junit5 and click Finish. When done the Java Build path should look as follows:

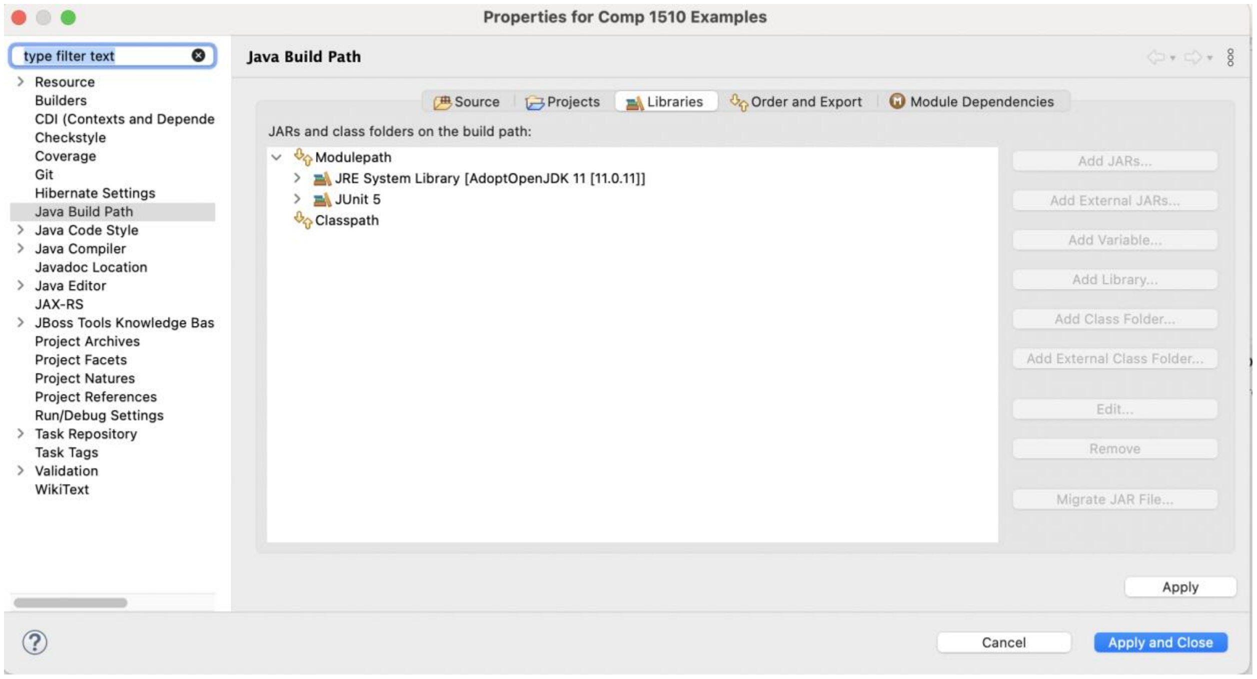
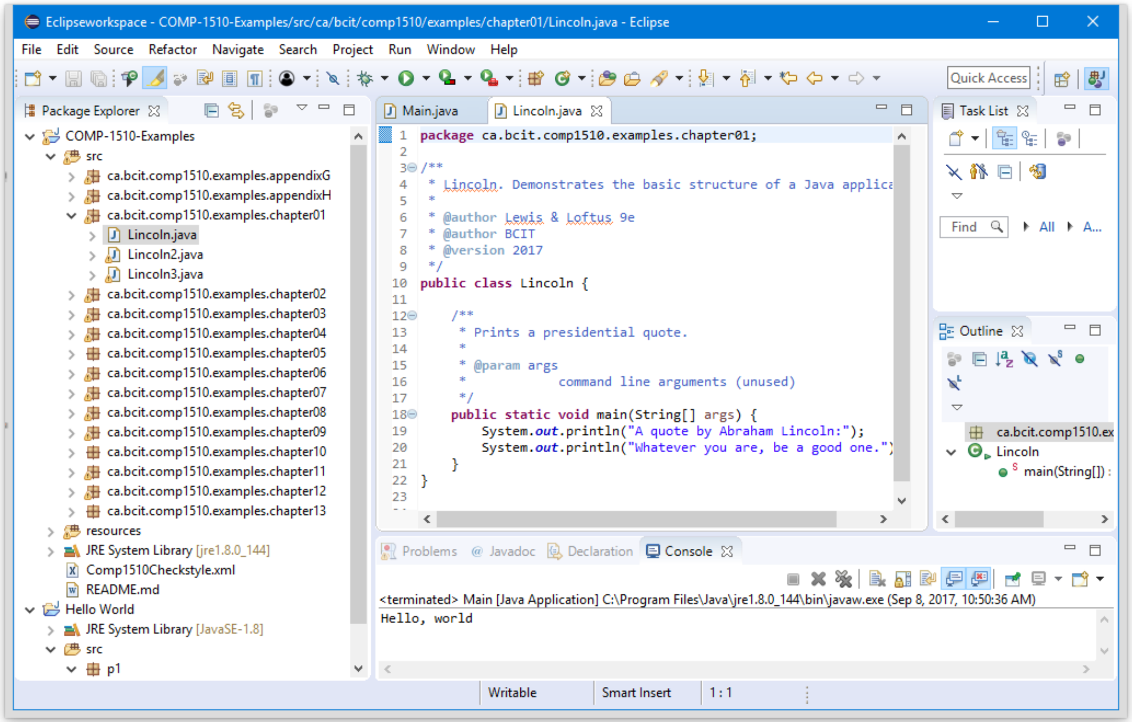


Figure 16: Libraries for Comp 1510 Examples

5. Click on the Source tab and verify that Eclipse has picked up the resources, src, and test folders for the project. If not, click Add Folder… and add them.
6. After the checks under Java Build Path, click Apply and Close.
7. 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):


Eclipseworkspace - COMP-1510-Examples/src/ca/bcit/comp1510/examples/chapter01/Lincoln.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
COMP-1510-Examples
src
ca.bcit.comp1510.examples.appendixG
ca.bcit.comp1510.examples.appendixH
ca.bcit.comp1510.examples.chapter01
Lincoln.java
Lincoln2.java
Lincoln3.java
ca.bcit.comp1510.examples.chapter02
ca.bcit.comp1510.examples.chapter03
ca.bcit.comp1510.examples.chapter04
ca.bcit.comp1510.examples.chapter05
ca.bcit.comp1510.examples.chapter06
ca.bcit.comp1510.examples.chapter07
ca.bcit.comp1510.examples.chapter08
ca.bcit.comp1510.examples.chapter09
ca.bcit.comp1510.examples.chapter10
ca.bcit.comp1510.examples.chapter11
ca.bcit.comp1510.examples.chapter12
ca.bcit.comp1510.examples.chapter13
resources
JRE System Library [jre1.8.0_144]
Comp1510Checkstyle.xml
README.md
Hello World
JRE System Library [JavaSE-1.8]
src
p1
Main.java
Lincoln.java
Task List
Find
All
A...
Outline
ca.bcit.comp1510.ex
Lincoln
main(String[]) :
Problems
Javadoc
Declaration
Console
Writable
Smart Insert
1 : 1

```
package ca.bcit.comp1510.examples.chapter01;

/**
 * Lincoln. Demonstrates the basic structure of a Java applica
 *
 * @author Lewis & Loftus 9e
 * @author BCIT
 * @version 2017
 */
public class Lincoln {

    /**
     * Prints a presidential quote.
     *
     * @param args
     *            command line arguments (unused)
     */
    public static void main(String[] args) {
        System.out.println("A quote by Abraham Lincoln:");
        System.out.println("Whatever you are, be a good one.")
    }
}
```

```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Sep 8, 2017, 10:50:36 AM)
Hello, world
```

Figure 17: Source file for a Chapter 1 program

8. Open the source examples for any chapter by clicking on the triangle beside the corresponding package name.

9. Open any of the programs for reading/editing/running by double clicking on it.
10. 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.

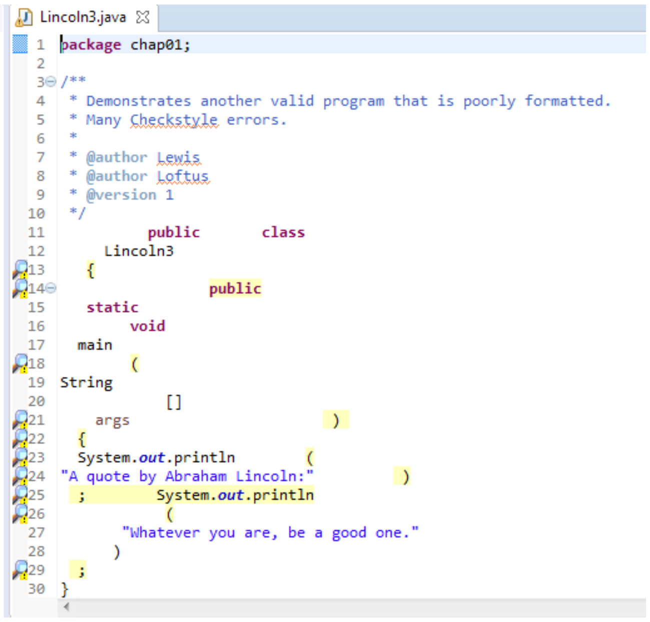
15. Configure Checkstyle

1. We need to add the Comp1510Checkstyle parameters to define our initial style (as described next). You may need to repeat this for assignments to get specific style definitions. 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.

1. Select Type: External Configuration File.
2. Enter a Name (a good name is 1510 style)
3. Browse to the location of the Comp1510Checkstyle.xml in the Comp1510Examples project
4. Click OK.
5. Select the configuration just created and click the Set as Default button to make that configuration file the default.
6. Click Apply and Close to close the Preferences window and answer "Yes" if Eclipse suggests that some projects need to be rebuilt.
7. 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 and the Comp 1510 Examples project.

Working with Checkstyle Violations

8. The Checkstyle plugin will flag lines that contain style violations by placing a little yellow magnifying glass symbol in the left margin. We will see 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:



```
package chap01;

/**
 * Demonstrates another valid program that is poorly formatted.
 * Many Checkstyle errors.
 *
 * @author Lewis
 * @author Loftus
 * @version 1
 */
          public       class
     Lincoln3
   {
                 public
   static
        void
  main
         (
String
            []
    args                          )
  {
  System.out.println         (
"A quote by Abraham Lincoln:"          )
  ;       System.out.println
            (
       "Whatever you are, be a good one."
      )
  ;
}
```

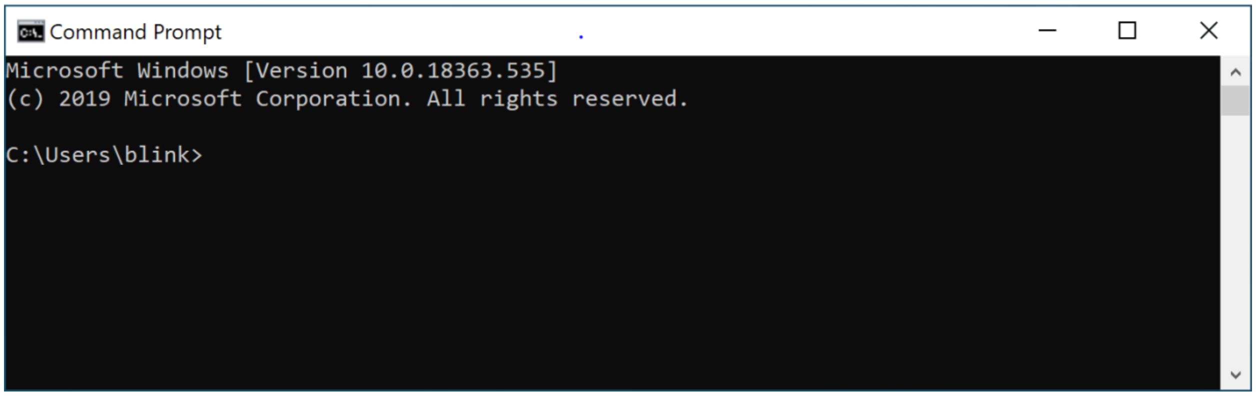
Figure 18: Lincoln3.java contains Checkstyle violations

9. Move the mouse cursor over the magnifying glasses to see the errors.
10. 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.
11. When you edit a source file and save it, Checkstyle will automatically reassess the file and update the violation list.

16. 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):

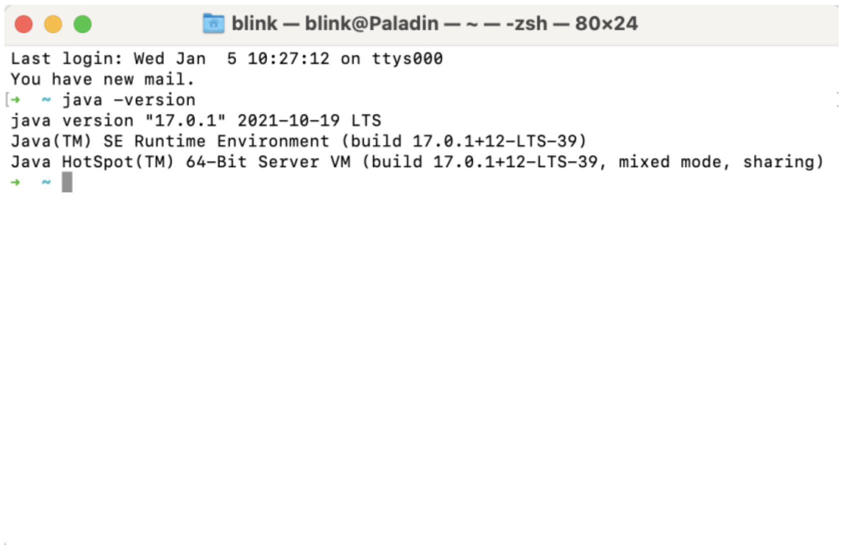
1. To open a Command Prompt in **Windows**, open the Start Menu and type Command in the search field. Windows will find Command Prompt. Click it to open it.
2. To open a **Terminal** in macOS, choose Applications > Utilities > Terminal.
3. The Command Prompt looks like this in Windows:



```
Microsoft Windows [Version 10.0.18363.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\blink>
```

Figure 19: Windows Command Prompt

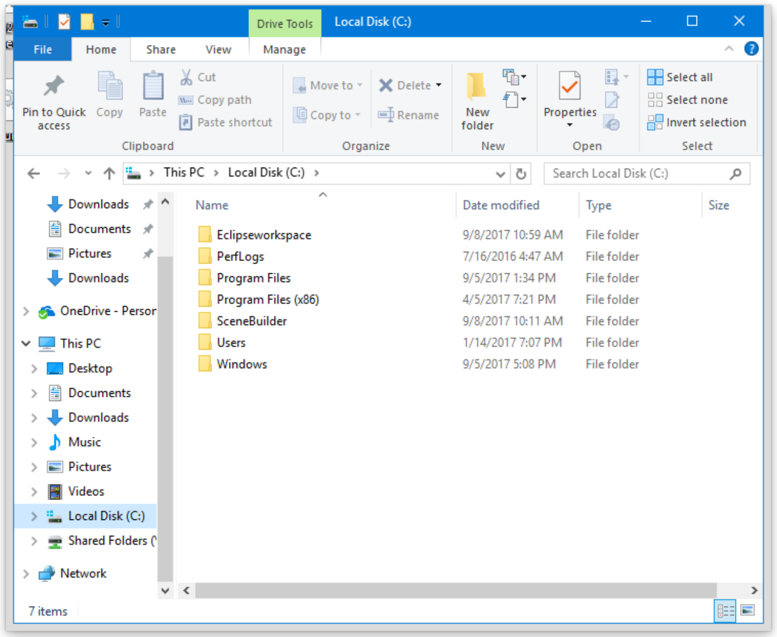
4. The Terminal in macOS looks like this:



```
Last login: Wed Jan  5 10:27:12 on ttys000
You have new mail.
➜  ~ java -version
java version "17.0.1" 2021-10-19 LTS
Java(TM) SE Runtime Environment (build 17.0.1+12-LTS-39)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.1+12-LTS-39, mixed mode, sharing)
➜  ~
```

Figure 20: 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:



Name	Date modified	Type	Size
Eclipseworkspace	9/8/2017 10:59 AM	File folder	
PerfLogs	7/16/2016 4:47 AM	File folder	
Program Files	9/5/2017 1:34 PM	File folder	
Program Files (x86)	4/5/2017 7:21 PM	File folder	
SceneBuilder	9/8/2017 10:11 AM	File folder	
Users	1/14/2017 7:07 PM	File folder	
Windows	9/5/2017 5:08 PM	File folder	

Figure 21: The C drive viewed in Explorer

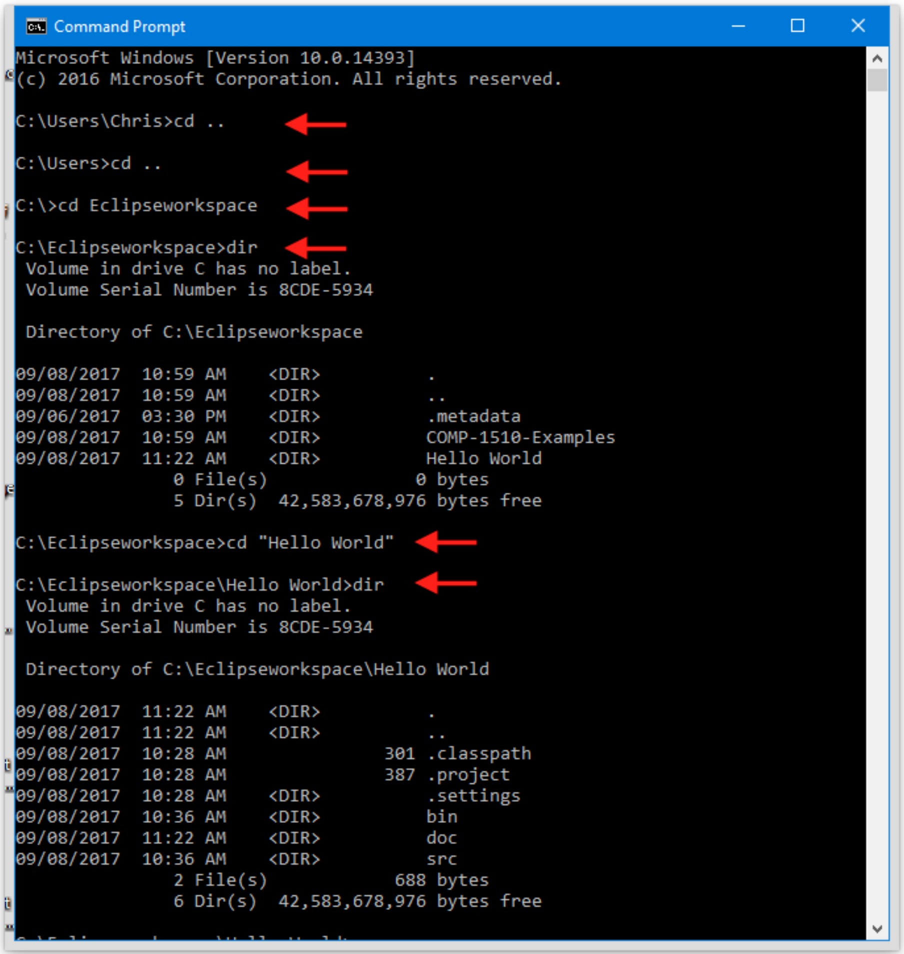
Command Line Commands: cd and dir (Windows) / ls (macOS)

The first command is 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. Navigate to your project folder. In my case it is in C:\Eclipseworkspace\Hello World. You should remember where your workspace is, if not, find it.
8. Note the following when using the cd command:
    a. To go "Up" or "back" a level, use cd .. (that's cd followed by a space 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 22: 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 for Windows). DIR gives you a "DIRectory listing" of files and directories are located in a given directory.

macOS: The macOS system uses **LS** (LS, lS, 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.

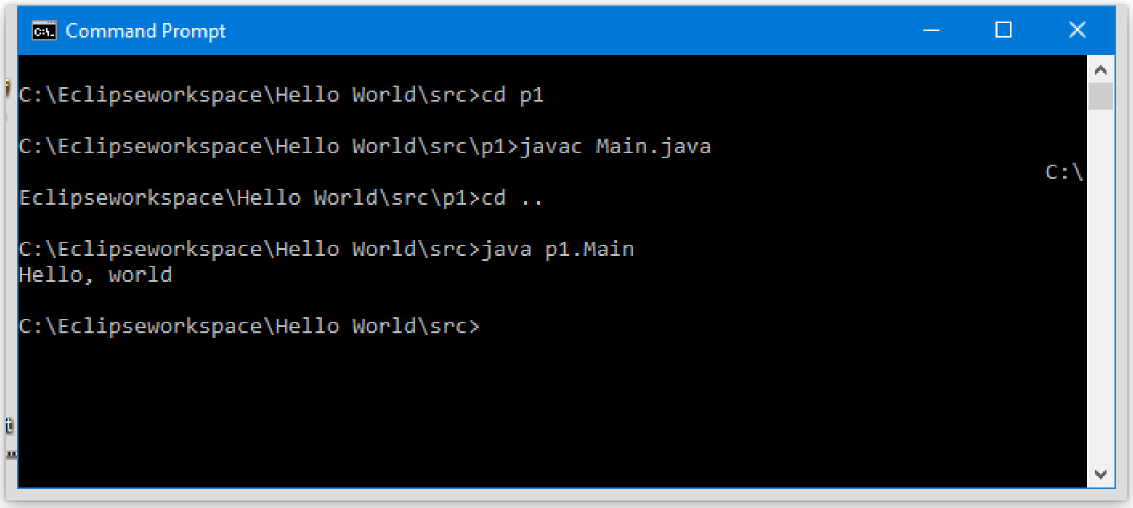
Compiling from the Command Line

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 compile source code manually, i.e., from the command line, we have to run javac (the java compiler). Remember that this command in 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.

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):



```
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>
```

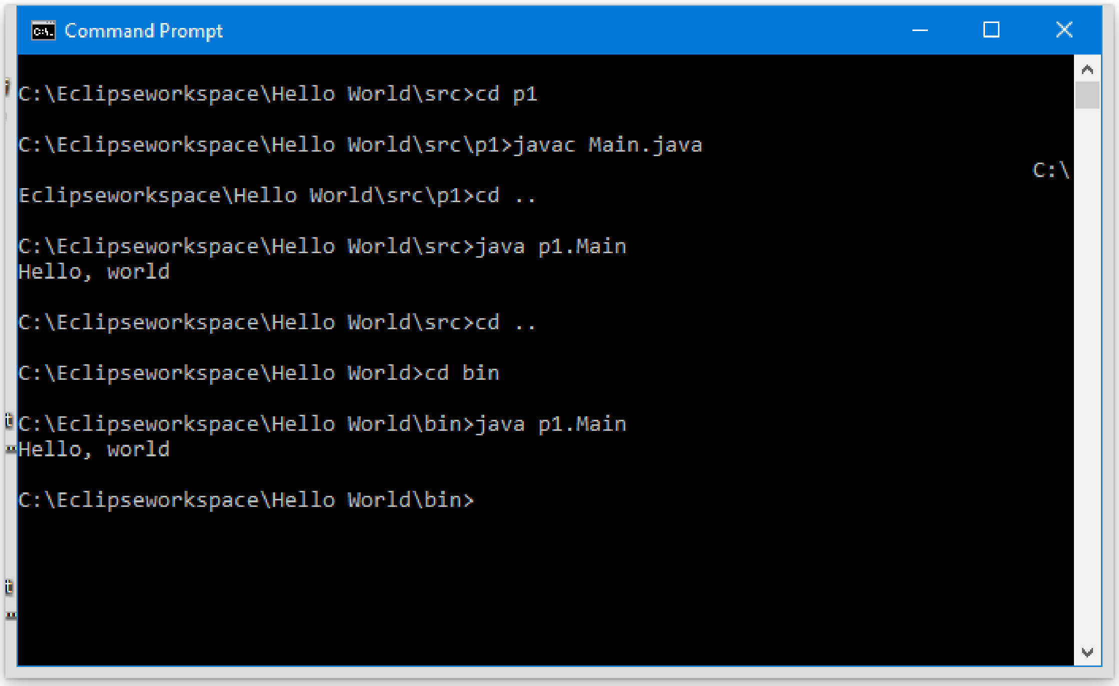
14.

Figure 23: Compiling from the Command Prompt

Some important notes:

15. When we compile a java file, we need to tell the javac command the name of the file. Use the entire file name including the extension .java with the javac command.
16. 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 24: Execute the class file Eclipse auto-compiled

Additional Things to Know about the Command Prompt

Remember: 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 (you will use this later, so remember it)

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.

There are also an input redirection < that takes the file following the < and makes it the input to the program (come back to this after we learn how to read from the keyboard (standard in).

You Have Completed Lab 0