Lab 1 – Eclipse Setup

*Last updated: 7.28.2024*

Contents

[1 Introduction 1](#_Toc174447596)

[2 Gain Access to GitHub Copilot 2](#_Toc174447597)

[3 Install Java 3](#_Toc174447598)

[4 Install Eclipse 4](#_Toc174447599)

[5 Test Eclipse for Java Development 6](#_Toc174447600)

[6 Using a Jar File 10](#_Toc174447601)

[7 Submitting your Work 11](#_Toc174447602)

# Introduction

The objective of this Lab is to install the most recent versions of the Java SDK, Eclipse, and then write a hello world program in Java using Eclipse. You will also create a GitHub account and apply for the GitHub Student Developer Pack and access to Copilot, an AI tool that we will use in Lab 4b.

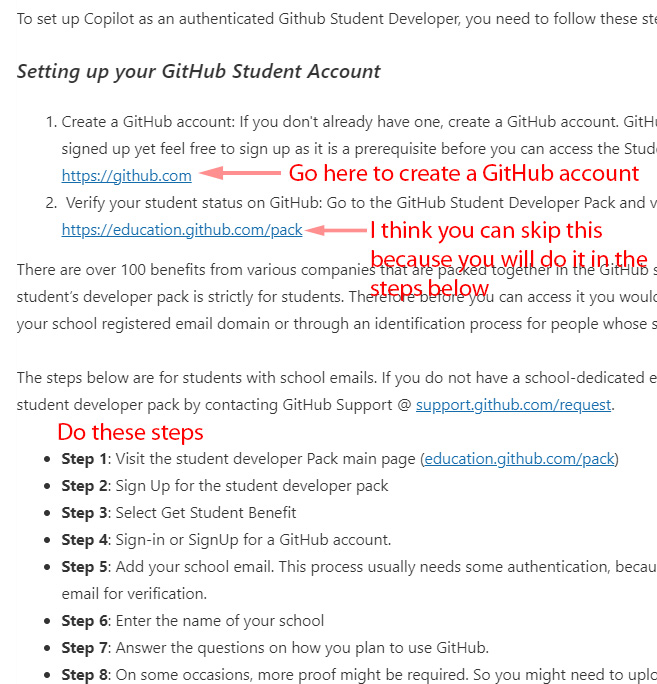
If you do not have a desktop or laptop, then you will work in the CS Open Lab (Nevins 2111). If this is the case, then do [Step 2](#_Gain_Access_to) and then advance to [Step 5](#_Test_Eclipse_for).

# Gain Access to GitHub Copilot

Do the following to obtain access to GitHub Copilot. We will set it up and learn to use this in Lab 4b. You are doing this now because it takes a few days to verify your student status.

1. Visit [Setting Up GitHub Student](https://techcommunity.microsoft.com/t5/educator-developer-blog/step-by-step-setting-up-github-student-and-github-copilot-as-an/ba-p/3736279), and as shown in the image below, do the things indicated. Notes:

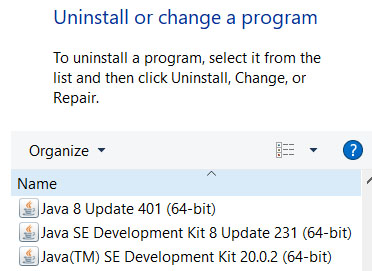
* For item “1” at the top of image, create a GitHub account. You can use your VSU email or another as your primary email. If you don’t use your VSU email (which is fine), you will eventually have to supply your VSU email account as part of “Student Status Verification” and that will be your secondary email.
* I believe you can skip item “2”, as you will do that when you do the 8 “Steps” below that.
* For “Step 5”, you will enter your VSU email for verification. I believe it will ask you for more proof of student status. Notes:
  + The official documentation says, “…upload an image of your school ID, academic transcript, or other document showing proof of your academic status that clearly shows at least one date”
  + I believe your VSU ID does not work because it doesn’t show any year or current term in it.
  + Some approaches my students have used to verify their student status:
    - A screen shot of their schedule as shown Visual Schedule Builder. Interestingly, it did not show their name. However, this worked.
    - A screen shot of Degree Works and that worked. The student did not show me the screen shot, so I don’t know exactly what the screen shot looked like.
    - A screenshot of my digital VSU 1Card, as well as a screenshot of my personal information from the Student profile tab in the MyVSU Banner. Both contained my student ID# and the student profile page showed the same VSU email that was used in the GitHub account creation.
    - I requested a document from my advisor with the VSU letterhead and a date indicating that I am taking classes remotely.
  + From talking to students, it may take several tries at this to get it to go through.
  + There is a support email, so use that as a last resort. I can’t find it right now, but it is in the pages for this process.



1. Once your student status is verified (takes a day or two), you are done with this. We will set it up and learn to use it in Lab 4b. So, move on the next section!

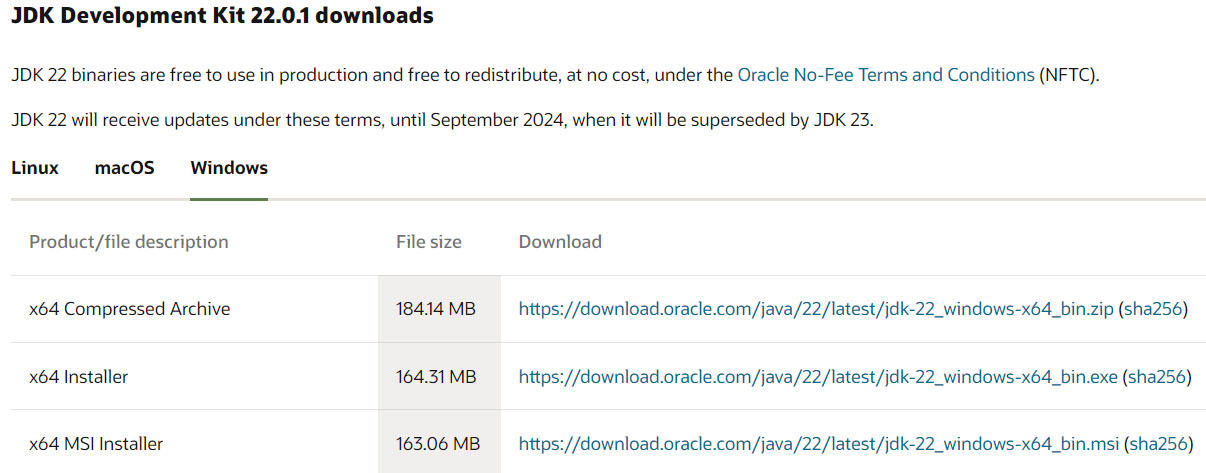
# Install Java

In this section we will verify that we have Java SE installed and if not, install it.

1. Check to see if you have some version of Java SE installed.
2. Launch the Control Panel app[[1]](#footnote-1) and (i) find “Programs” (on my computer I had to search within the CP), (ii) choose: “Add or remove programs.”
3. When the list of programs populates, scroll down and find Java. On my computer, the display shows I’m using version 20.0.2 (and others) as shown in the figure below (which is very old. You need version 20 or above. If you do have version 20 or above then you don’t need to install the latest Java and can proceed to [Stage 4](#_Install_Eclipse). Otherwise, continue below.
4. Install the most recent version of Java SE. Do this by visiting: <https://www.oracle.com/java/technologies/downloads/>. Scroll down to the downloads for the most recent version of Java. It should say something like this:

**Java Development Kit 22.0.1 downloads**

Choose Windows (or Mac or Linux) to see the correct choices as shown below for Windows (except title and version are older). Install by running the exe (for Windows, X64 Installer).



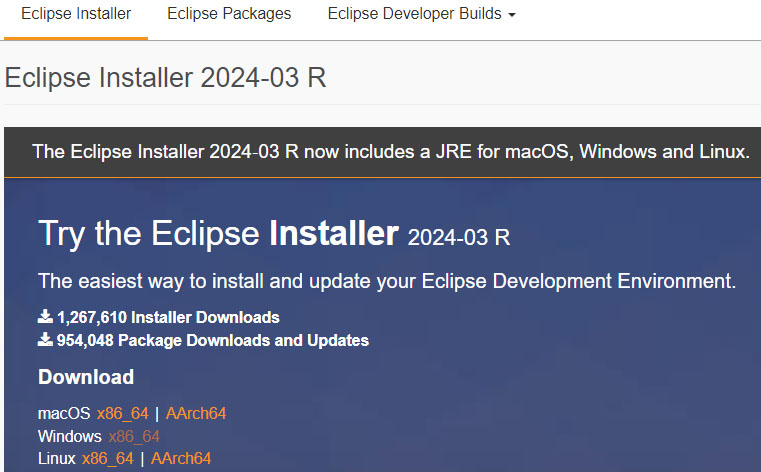
# Install Eclipse

These directions assume you are using Windows.

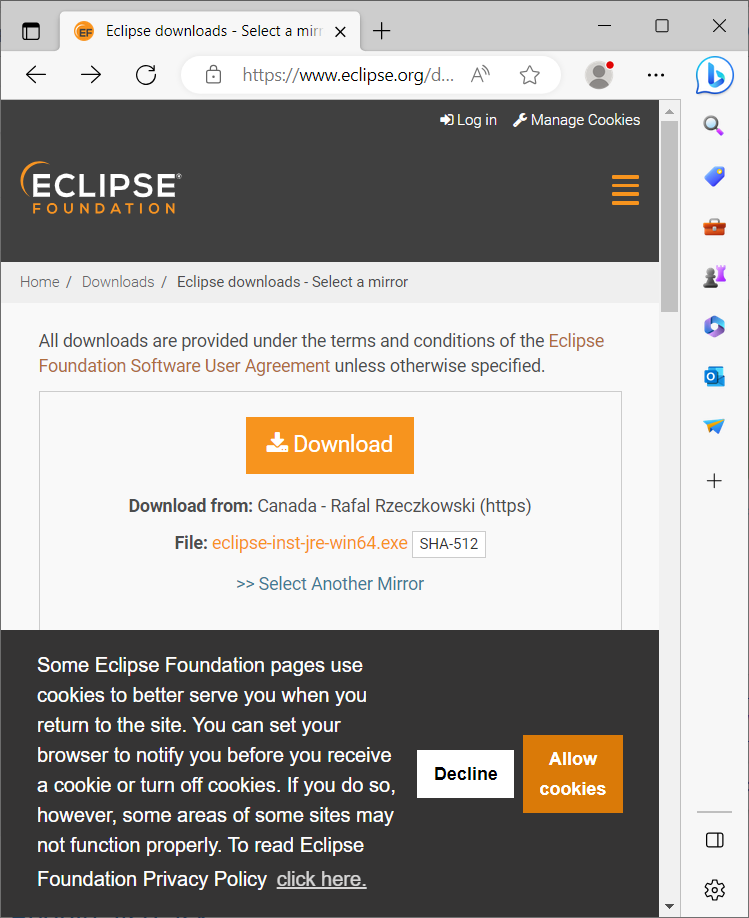
1. Visit the site below and choose: “Download 2024-03” (as of 5.17.24).

|  |  |
| --- | --- |
| <https://eclipseide.org/> |  |

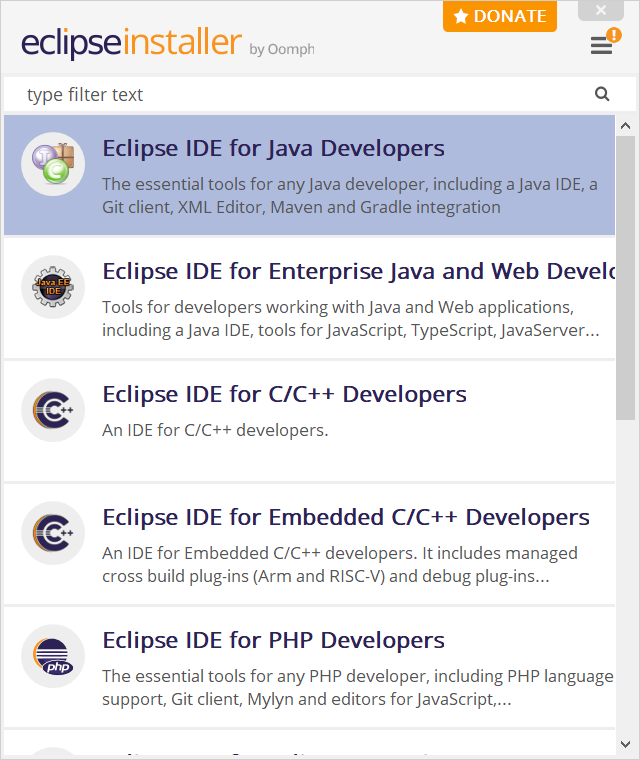
1. Choose the proper download. If needed, below the installer are instructions to set up manually.



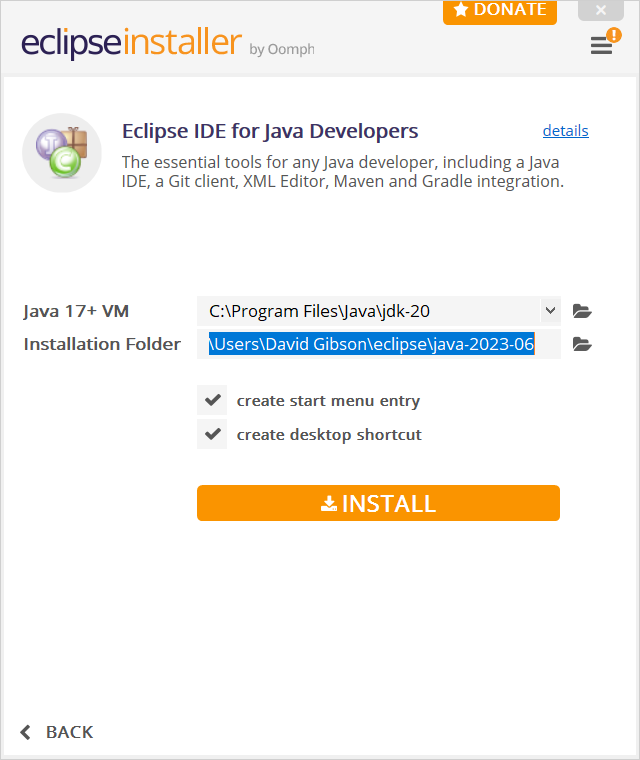
1. On the resulting dialog, choose: Download

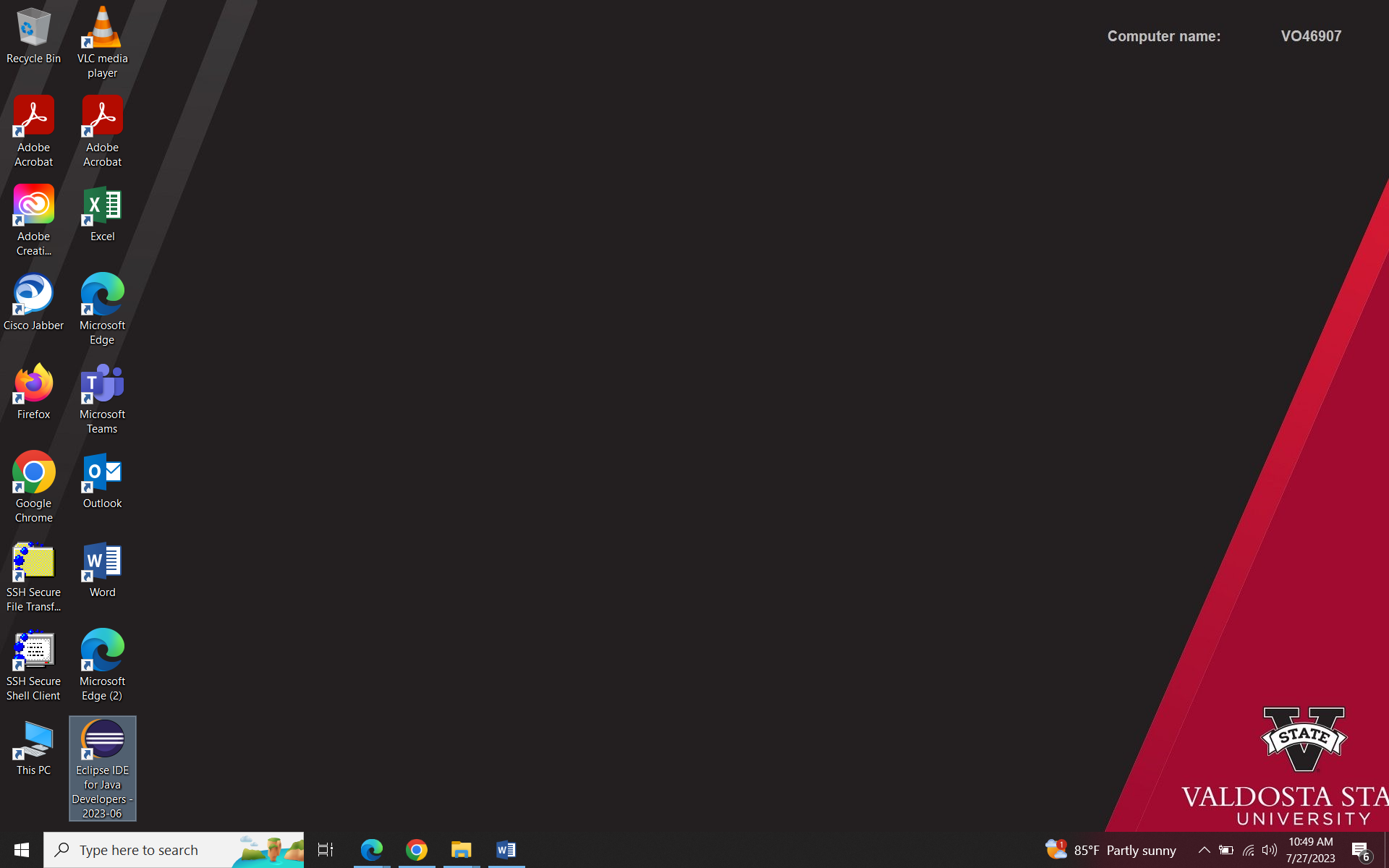


1. An *exe* installer will download. Run it to install. On the resulting dialog, choose: “Eclipse IDE for Java Developers”

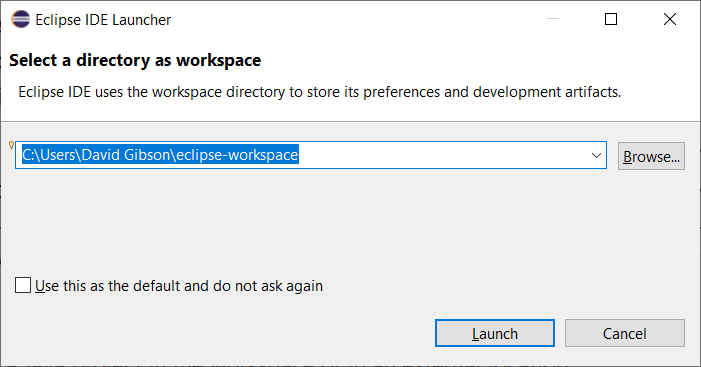


1. Note the installation folder: *c:\users\yourID\eclipse\java-…* You may need to check that some time. In other words, it is not in the usual, *Program Files.*

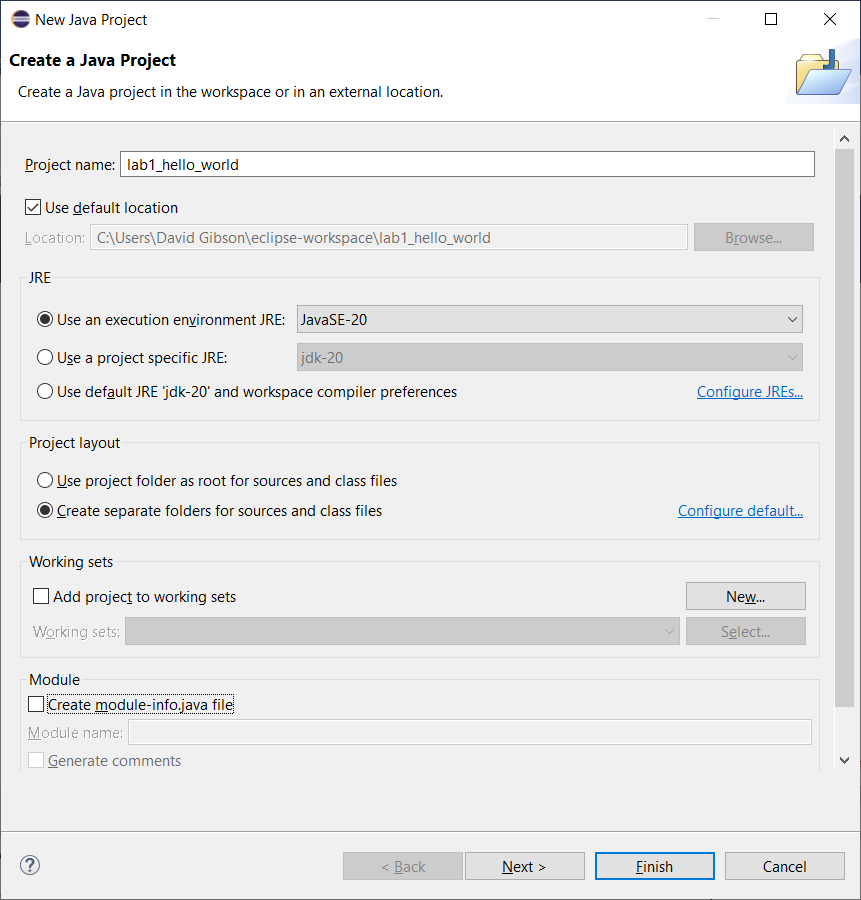


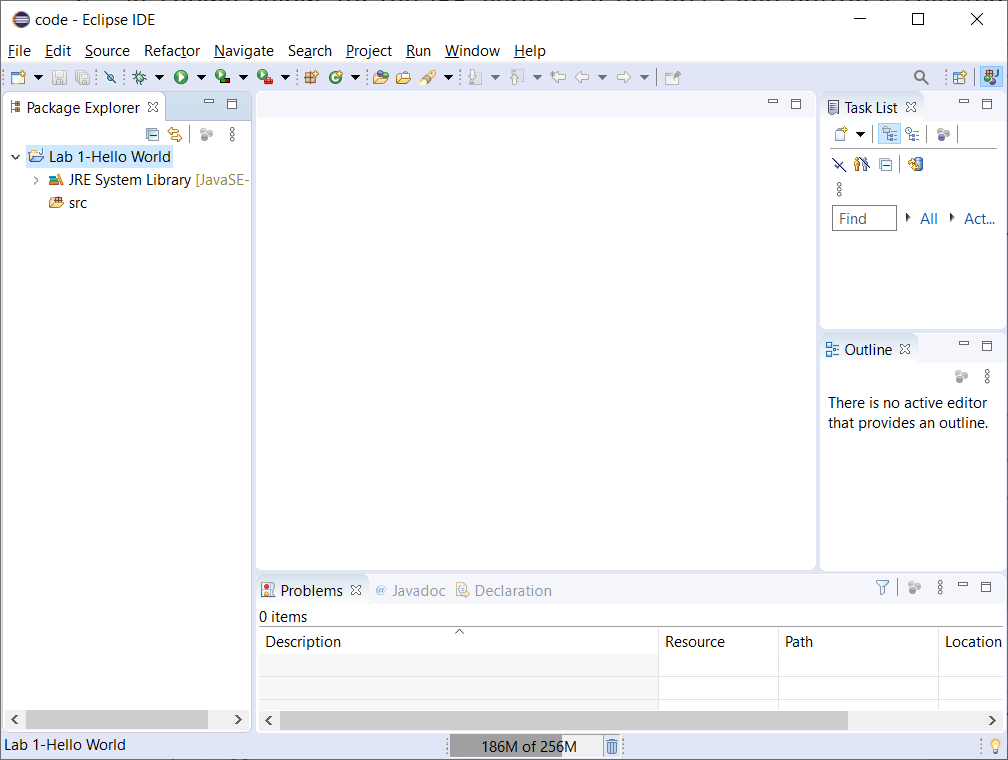
1. At the conclusion, it will ask you to launch Eclipse. Choose: Launch. At the next dialog, it will ask you to confirm the workspace. **CHOOSE: CANCEL**. We will run it in just a minute.
2. Find your desktop icon as shown on the right.

# Test Eclipse for Java Development

1. Run Eclipse, accept the default workspace. Eclipse uses the term, *workspace*, to refer to any folder where you will store one or more *Java Projects.* In a later lab, we discuss the use of a *workspace* in a bit more detail. So, accept the default, or create a folder anywhere you like and use that.
2. In Eclipse, do the following:
3. Choose: File, New, Java Project.
4. Type the Project name as shown below (no spaces): “lab1\_hello\_world”.
5. **Uncheck** “Create module-info.java file” (at the bottom).[[2]](#footnote-2)
6. Choose: Finish.

Note: your execution environment JRE will possibly be different.

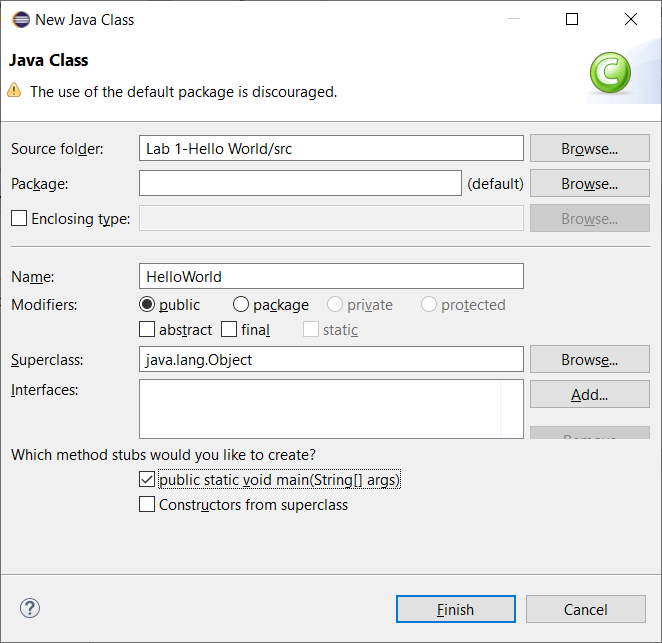


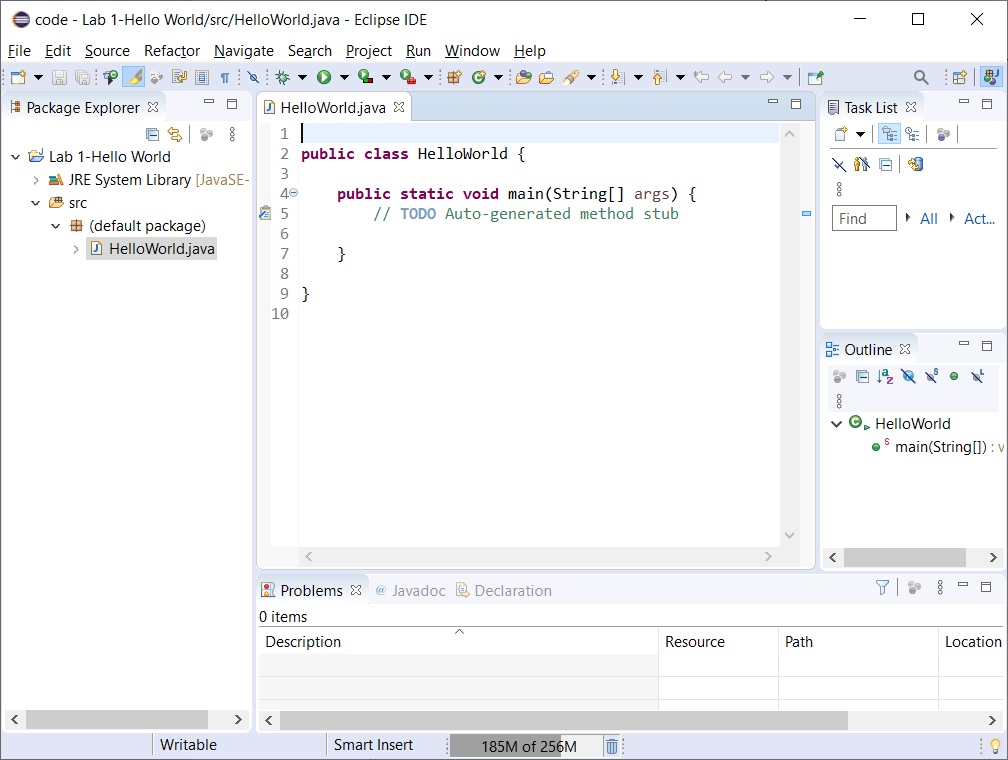
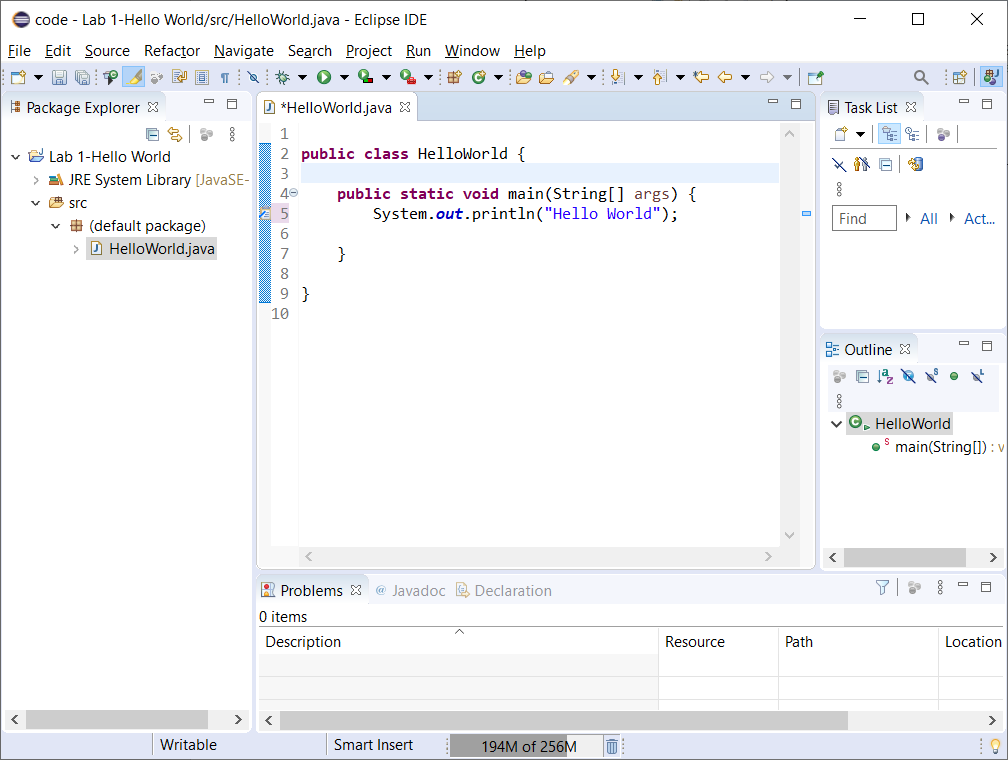
1. Do the following:
2. As shown on the right, the project will be shown in the Package Explorer. **You may need to close the Welcome page first.**

Note:

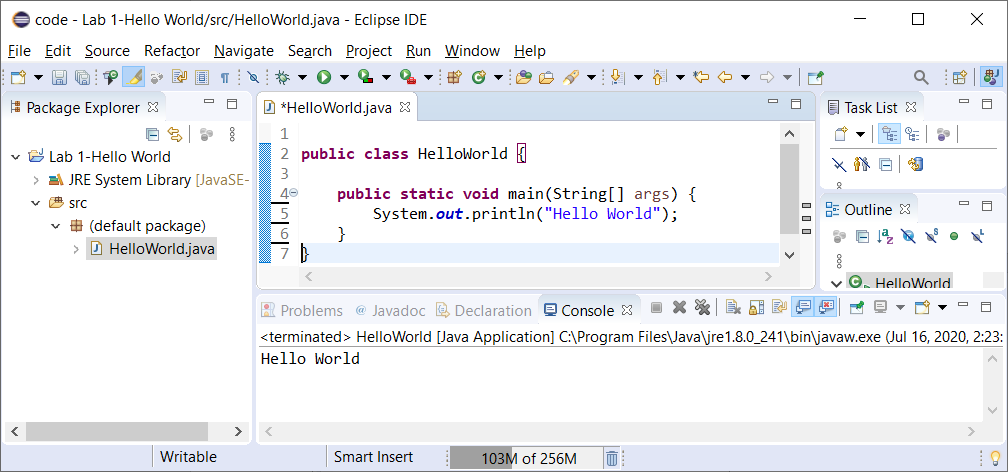
* If the Package Explorer is not visible, you can display it by choosing: Window, Show View, Package Explorer.
* The project name is using initial capital letters in the figure above, where yours will show lower case.

1. Expand the project node and identify the *src* node. This is where the *source code* is stored, i.e. the code you write will be stored here.
2. Create a class by doing the following (reference figure below):
3. Select the *src* node.
4. Choose: File, New, Class
5. Note the *Source folder* – that is where the class/file will be stored. No change is needed.
6. Supply the *Name*: “HelloWorld”
7. Select the checkbox that says: “public static void main…”
8. Choose: Finish



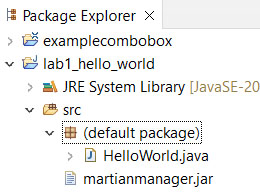
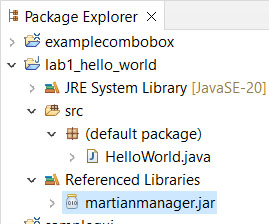
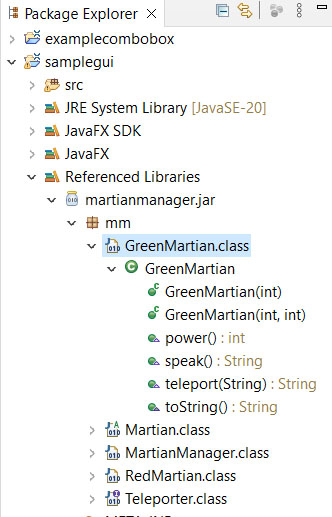
1. The code window will be displayed as shown on the figure on the right. Also shown is that node for the file has been created in the Package Explorer.
2. Add this line of code to *main*:

System.***out***.println("Hello World");

1. To run the program, Choose: Run, Run (or, alternately, press the Green arrow icon). The output is displayed in the console (see bottom of screen). (If you don’t see the console, choose: Window, Show View, Console). If you have multiple monitors, it is convenient to pull the Console tab to another monitor.
2. If all went well, congratulations, you have just run your (possibly) first Java program in Eclipse. Close Eclipse.

# Using a Jar File

A *JAR* file is the standard way that Java code is distributed. For some assignments, I may give you code in a JAR file. In this section we will show how to use a JAR file.

1. Download: *martianmanager.jar.*
2. Drag the JAR file into the *default package* node into the *application* folder in the Package Explorer as shown on the right (choose: Copy files and OK on the dialog that appears).
3. Right-click the jar file in the Package Explorer and choose: Build Path, Add to Build Path. The file will “move” from the package folder to the Referenced Libraries folder as shown on the right.
4. Expand *martianmanager.jar* in the Referenced Libraries node. As shown on the right, for this example, the available classes are shown (*GreenMartian, Martian, etc.*), as well as the methods in each class and their signature (*e.g. GreenMartian* has a constructor that accepts an integer, and another that accepts 2 integers, *etc.*). In addition, it shows that these classes are in the *mm* package, this will be important next.
5. Add this line to the top of *HelloWorld.java*:

**import** mm.\*;

Note: to use any of the classes in the JAR file, you must first import them. The statement above imports everything (\*) in the JAR file. Note that you must include the package (*mm* for this example) in the path.

If this line, or the lines below don’t compile, then check to see if you have a *module-info.java* file. If you do, then this will not work and you should create a new project without a *module-info.java* file. The reason is that a project must be in a package if *module-info* is used.

1. Replace the code in *main* with:

RedMartian rm = **new** RedMartian(3,3);

System.***out***.println("Hello World\n" + rm.toString());

1. Run the program and you should see the following in the Console:

Hello World

Red Martian - id=3, vol=1, ten=3

# Submitting your Work

Submit your *HelloWorld.java* file in the Lab 1 drop box on Blazeview by the due date.

**You are done!**

1. On my computer, I have to type, “control panel” into Windows search (Winkey+S) [↑](#footnote-ref-1)
2. If you do create a *module-info.java* file, then the next section will not work. You’ll need to create a new project without *module-info.java.*  [↑](#footnote-ref-2)