

CSCI 161: Introduction to Programming I

Pre-Lab1: Setting Up the Programming Environment: Linux and Eclipse

Goals:

- become acquainted with the Linux environment, create files using the Eclipse editor,
- become familiar with basic terminal commands,
- learn how to compile and execute a simple Java program, and
- use Autolab to submit your lab

Before you Start...

We know this is a long document. It is detailed and designed to set you up for the rest of the semester. So, take your time and **read this document carefully** – most of the instructions you will need are in this lab.

Overview

In this lab you will:

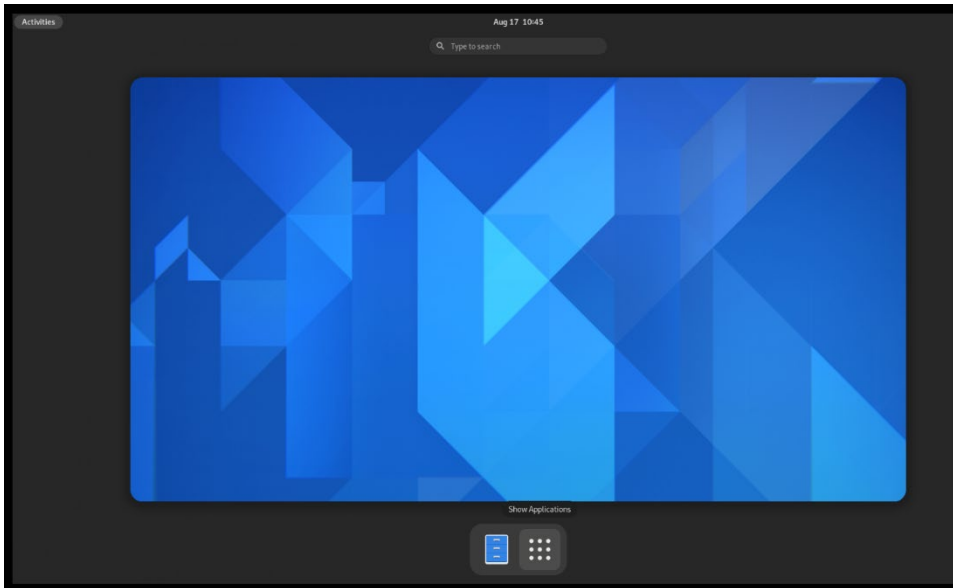
- a) login to a workstation running Linux with your personal account, using your email address as your login (e.g. YourEmail@Millersville.edu) and your password.
- b) be able to access the Terminal and carry our basic commands
- c) create a folder for storing your work for this course
- d) learn to use the Eclipse Integrated Development Environment (IDE)
- e) create and run a simple Java program
- f) Modify and save the Java program
- g) submit program to your professor for grading on AutoLab

All the aforementioned steps are detailed below. Key parts are in bold and then described in more detail.

a) Log in to the Linux environment

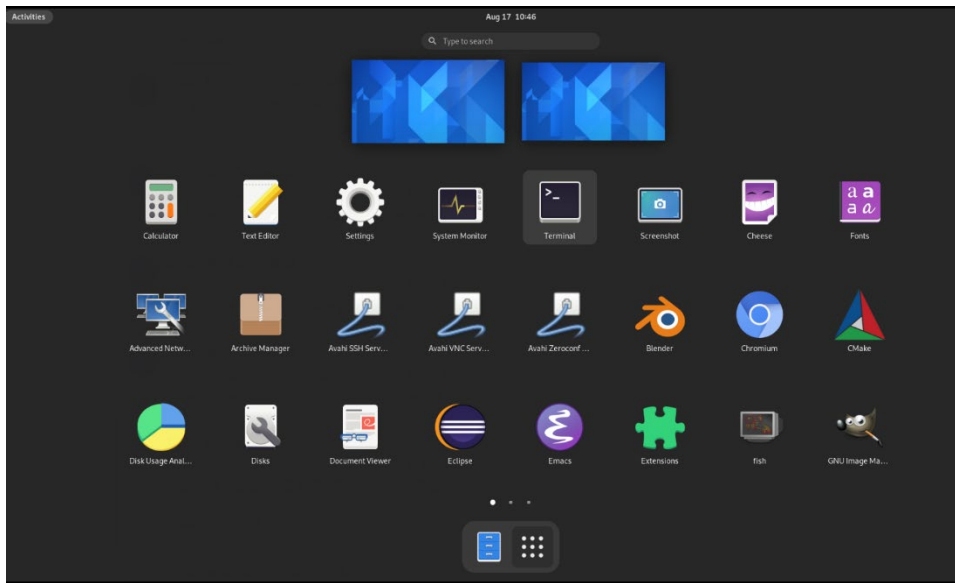
You should see a login panel asking for your user id. *Type your user id and password* (Millersville EMAIL login credentials). Use the number keys above the letters and not the numeric keypad. You won't see your password on the screen. *Press "enter" to login.*

b) Terminal: Although overall this is a graphical interface, you can interact with the system by typing commands. If you connect to the lab machines over the Internet, you will use a similar terminal interface with typed commands.



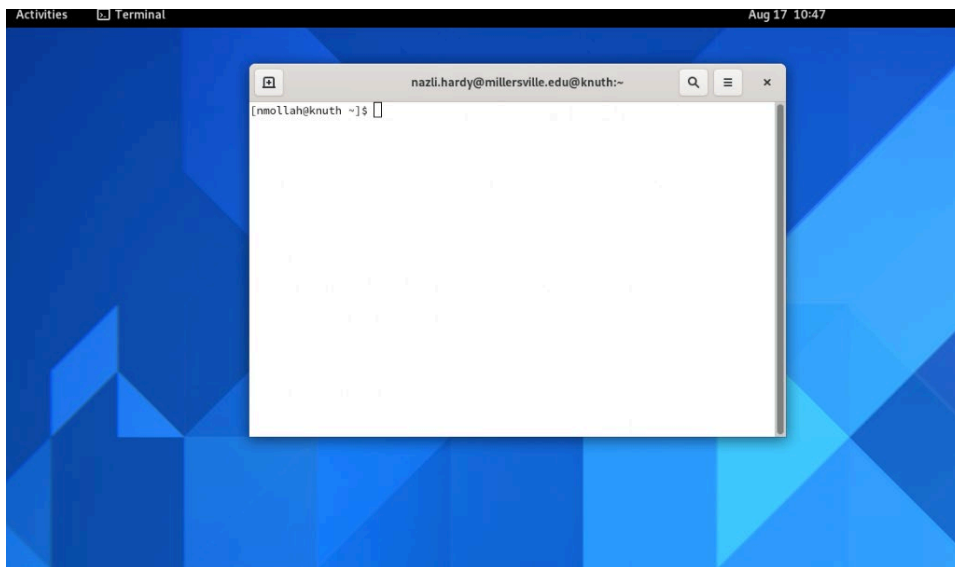
Click on "Show Applications" on the bottom.

You can now search for "Terminal" or find it from the list of applications (click on the 9-dotted icon)



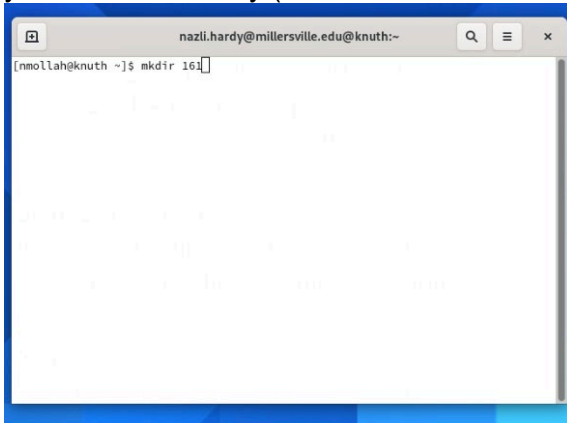
Double-click on "Terminal."

Once at the terminal window, you will type commands in this window. If you wish to adjust its size, drag the lower right corner. Many of the mouse gestures work the same under Linux as in Microsoft Windows.



c) Create and Open 161 Directory

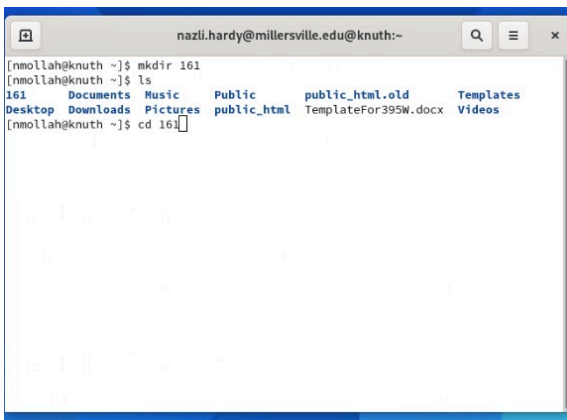
In the Terminal, type **mkdir 161**. This will create a new directory (folder) named '161' in your home directory (/home/students/<userid>).



```
nazli.hardy@millersville.edu@knuth:~  
[nmollah@knuth ~]$ mkdir 161
```

Now type **ls** (for 'list directory contents') and you will see your new directory in the list.

You will create and modify your programs for this class in your 161 directory.



```
nazli.hardy@millersville.edu@knuth:~  
[nmollah@knuth ~]$ mkdir 161  
[nmollah@knuth ~]$ ls  
161  Desktop  Downloads  Music  Public  public_html  public_html.old  Templates  
Desktop  Downloads  Pictures  public_html  TemplateFor395W.docx  Videos  
[nmollah@knuth ~]$ cd 161
```

The current directory is displayed as part of your terminal prompt, `userid@machine:<working directory>$`. The terminal starts at your home directory every time you start a terminal session or login. You can tell you're in your home directory when the prompt has a '~' (tilde) as the working directory (between the colon and the dollar sign).

In order to work with your files for this class in the terminal, you must make sure that the terminal's working directory is 161. Do this by **typing cd 161** (for 'change directory') at the prompt. You can check what the current working directory is by **typing pwd** or looking at your prompt. The output of **pwd** should be `/home/students/<userid>/161`. Note the '161' on the end.

```
nazli.hardy@millersville.edu@knuth:~/161
[nmollah@knuth ~]$ mkdir 161
[nmollah@knuth ~]$ ls
161      Documents  Music      Public      public_html.old  Templates
Desktop  Downloads  Pictures   public_html  TemplateFor395W.docx  Videos
[nmollah@knuth ~]$ cd 161
[nmollah@knuth 161]$ pwd
/home/faculty/nmollah/161
[nmollah@knuth 161]$
```

d) **Eclipse integrated development environment (IDE)**

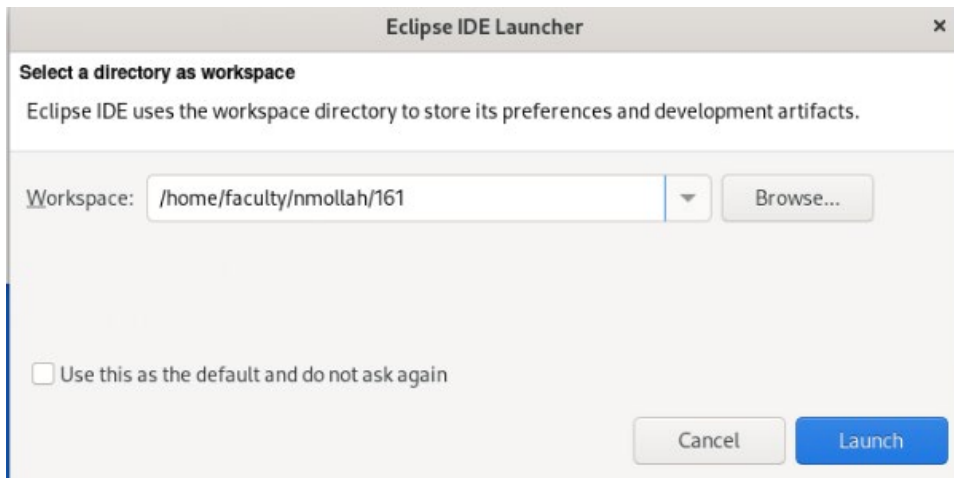
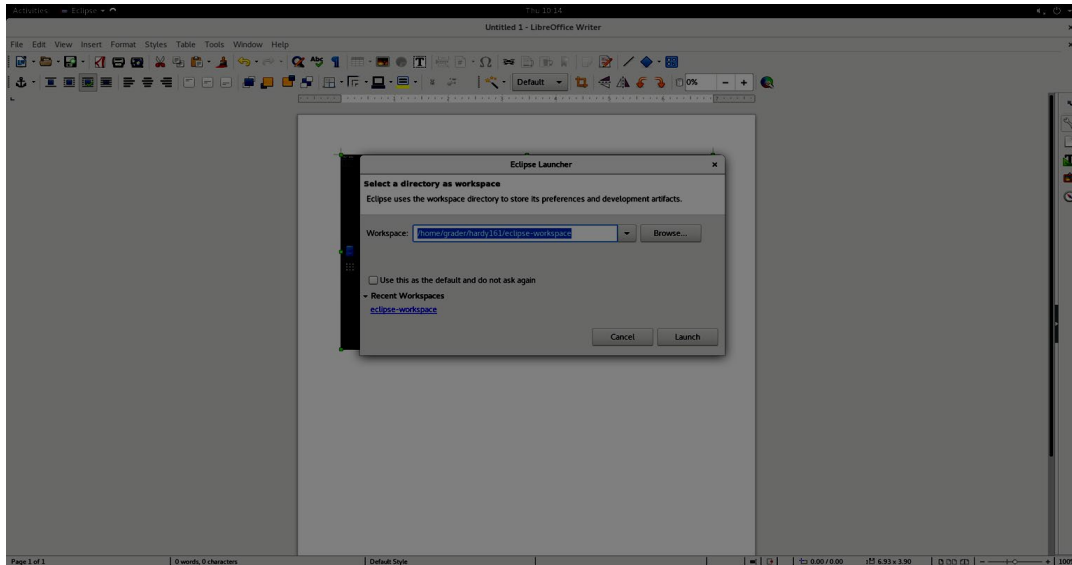
We will be using the Eclipse integrated development environment (IDE) in this class. To navigate to Eclipse, *go to Applications, then click on Eclipse*. To open Eclipse, *click on the Eclipse icon* (search or find in “applications”).



STOP! Before you proceed any further, read the next paragraph and follow instructions carefully.

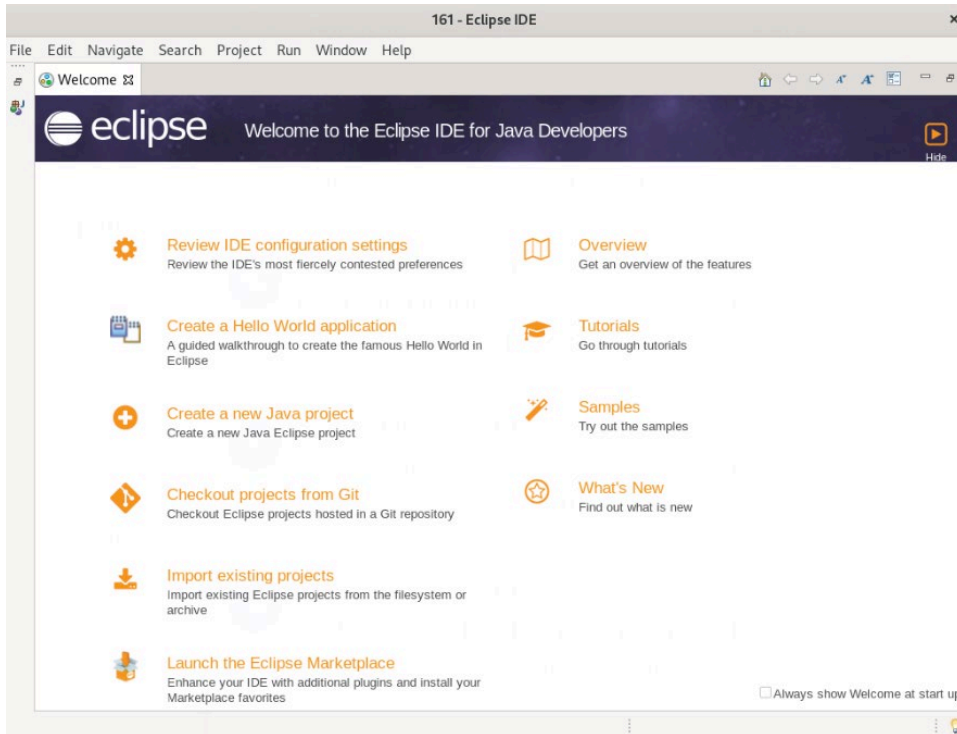
Eclipse will display a window asking for your workspace directory. *Delete the word **eclipse-workspace** and replace it with **161***. The workspace directory should now be *'/home/students/<userid>/161'*.

Also Click on the box below which asks if you want to use this as your default workspace directory so that this window is not displayed each time that you open Eclipse.

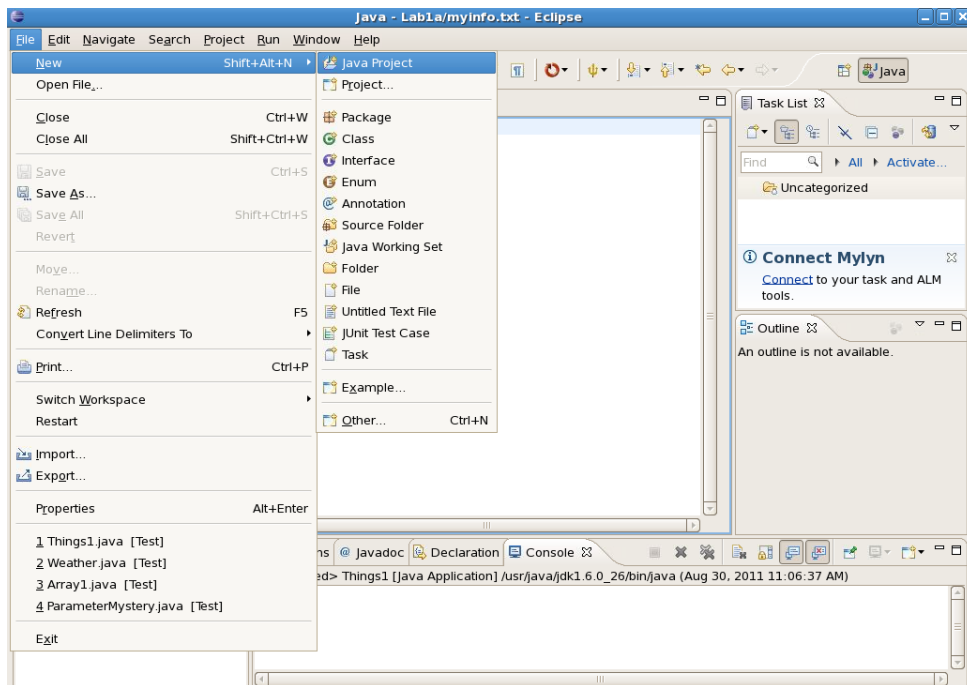


The first time you use Eclipse, it typically starts up with its Welcome window displayed. Each of the icons in this window give you an introduction to some part of the Eclipse system, except the one on the far right that looks like a downward-turning arrow.

That rightmost icon of the arrow (either “Workbench” or “Hide”) will launch the “Workbench”. Go ahead and click on that. Your window should now look like the screen shot below/ next page.

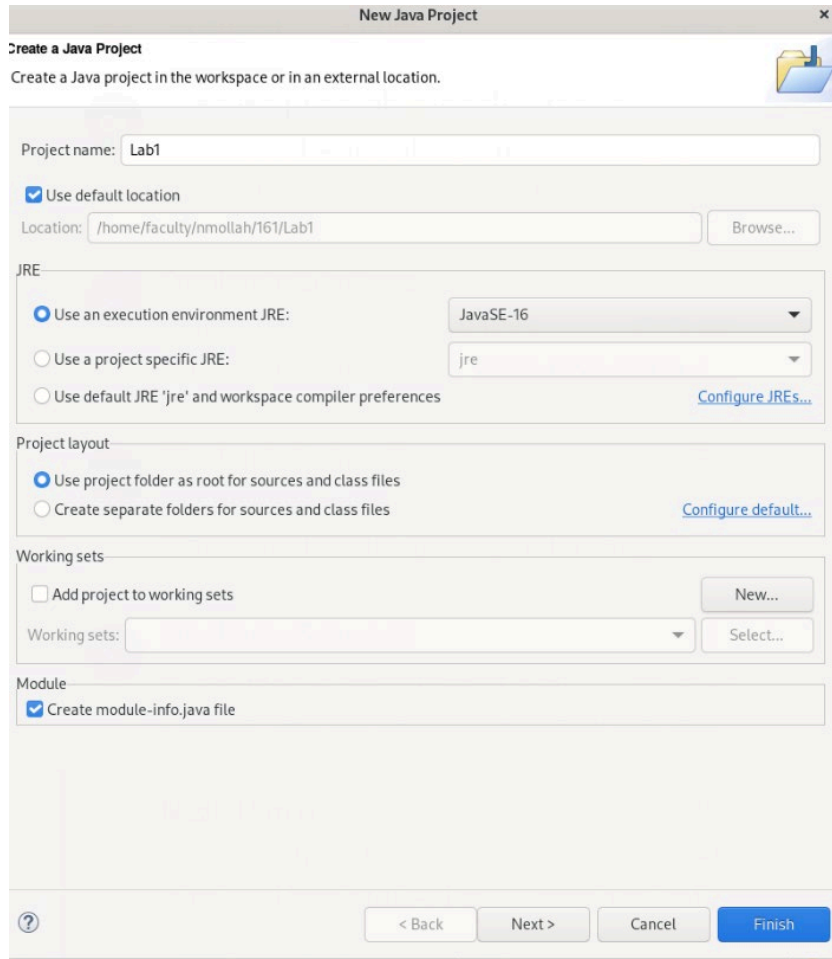


Create a new Eclipse project by selecting File-> New -> Java Project.



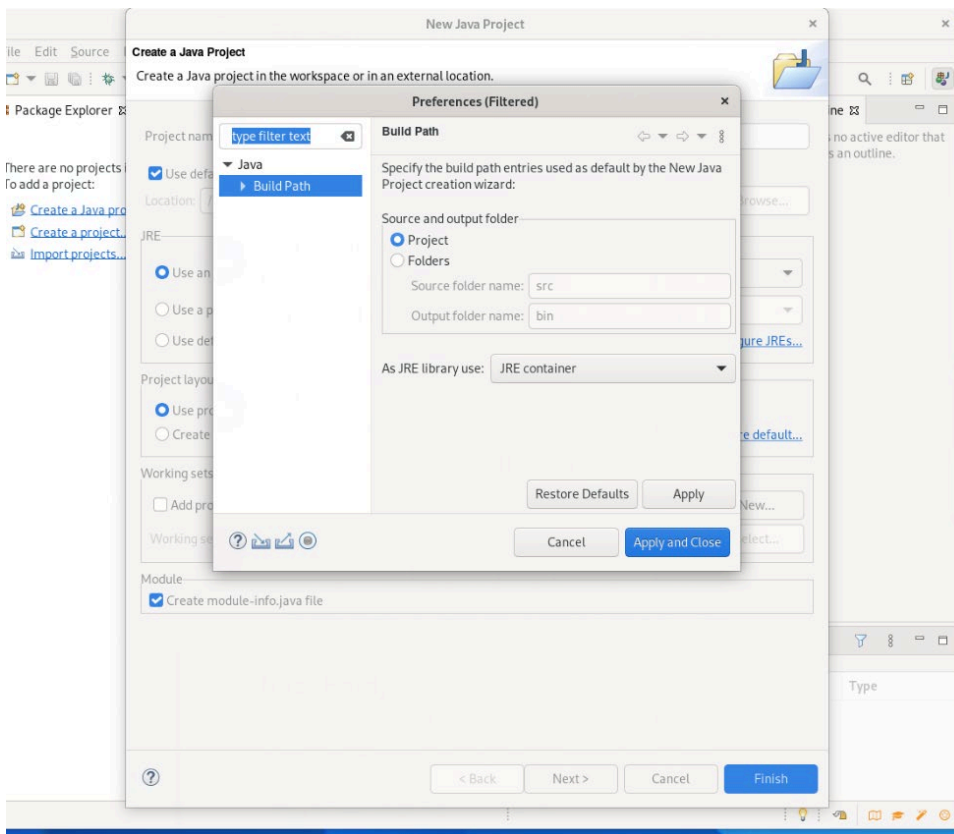
Type **Lab1** (one word) as the Project Name.

Select all buttons as shown below.



In the **Project layout** portion of the panel, click the *Configure Default* link.

On the new panel, *Click the Project button* under **Source and output folder**.



Click **Apply and Close**.

Select the **Use project folder as root for sources and class files** button.

Note that these settings should now be the default for future projects so you will not have to repeat this. But it is a good idea to check to see that this button is correctly selected each time.

Finish the project creation by *clicking the **Finish*** button at the bottom of the window.

e) **Your First Java Program**

Create a new Java class file by *clicking on the New Java Class button* (it is a green circle with a 'C' and a little '+'). If you accidentally click the arrow next to the New Java Class button (which causes a drop-down menu to appear), you need to select Class (likely the top entry).

A new dialog box will appear as shown in the figure below.

Make sure that Lab1 is the source folder name (make it so, if it isn't). **Type *Hello as the Name***.

Select all buttons as shown

New Java Class

Java Class

The use of the default package is discouraged.

Source folder: Lab1 Browse...

Package: (default) Browse...

Enclosing type: Browse...

Name: Hello

Modifiers: public package private protected
 abstract final static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

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

? Finish Cancel

Under “Which method stubs would you like to create?”, click on the box to create public static void main(String[] args) so that a check mark appears.

Now click the **Finish** button.

CREATING YOUR PROGRAM

In the editor pane, you will see a new tab entitled Hello.java. This file automatically has focus, i.e. it’s where things that you type will show up. This file contains the basic outline or stub of a Java class called Hello — the first line should read public class Hello {. You will fill in this stub with the rest of the code shown above.

Begin typing in the Java program shown, including the comment block above the line public class Hello {. Include your name in the comments, along with today’s date. As you type statements, pay close attention to capitalization (some letters must be capitalized) and spacing, both within a line and between lines. You want your code to look exactly like as shown.

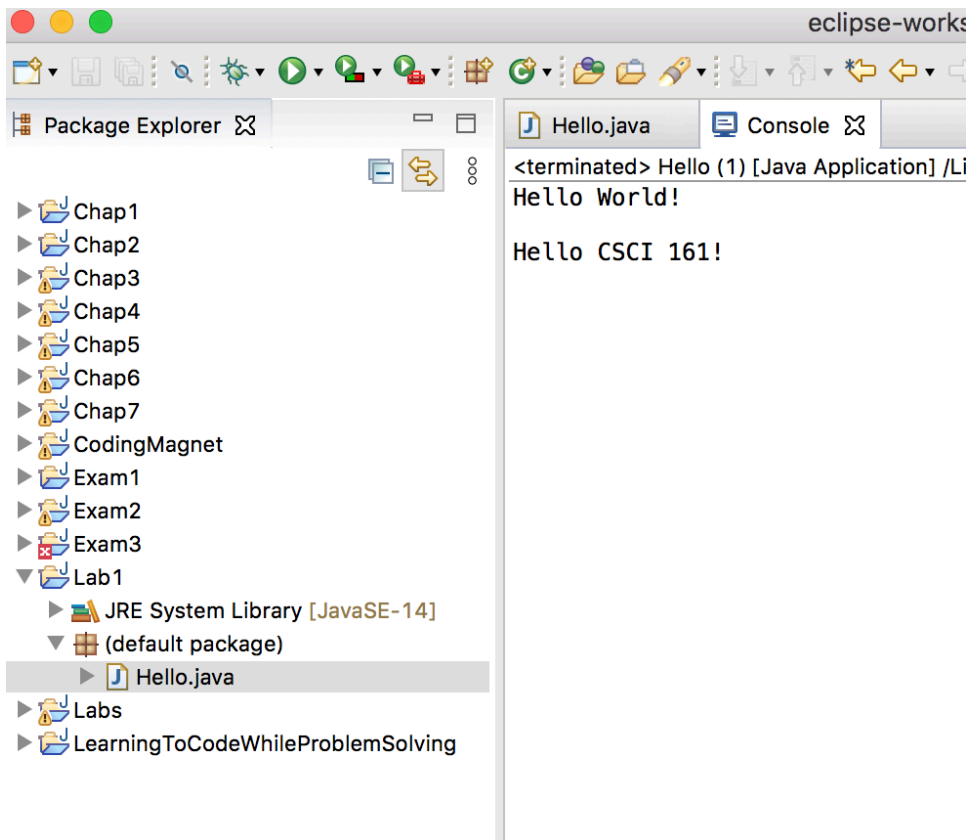
```
1 // student name
2 // professor name
3 // CSCI 161
4 // Due date of Lab
5
6 public class Hello {
7
8
9     public static void main(String[] args) {
10         System.out.println("Hello World!"); // this is a comment
11         System.out.println(); // this is an empty line
12         System.out.println("Hello CSCI 161!");
13     }
14 }
15
16
17
18
```

Insight: Code Formatting Get into the habit of indenting the program as shown above. The Eclipse editor makes it easy to use consistent indentation by automatically indenting the next line after the enter key is pressed. If you press Ctrl-Shift-F, Eclipse will automatically indent all of your code. If you are running macOS, Cmd-Shift-F will format your code.

Insight: Documentation This program has minimal comments. We will expect more on later assignments. Comments and indenting are essential to the readability of your program. Type in the comment regarding console input and output, and the println statement, as I have done above. Note that Eclipse will attempt to assist you as you are typing in the code by suggesting possible completions to what you have typed so far. You may ignore these suggestions for now.

RUNNING YOUR PROGRAM

Once you've typed in the entire program as shown above, *click the **Run** button* (the green circle with the white triangle that looks like a play button). (If you accidentally click the down arrow beside the Run button instead, you will need to select Run...).



If you have not saved the Hello.java file, you will be prompted to save your files — you should click the box for Always save resources before launching to avoid being prompted to save in the future.

This program is a very simple example of console output. When you run it, the sentence Hello, world! will appear in the console window (located at the bottom of the workbench window in the Console tab).

DEBUGGING YOUR PROGRAM

If Eclipse detects a mistake in your code as you are typing in your program, the line will be marked with a red wavy underscore. You should attempt to correct the error before running your program. If you do run a program and run-time errors occur, error messages will be shown in the Console pane. Continue to debug (correct) your program until you are able to run it without exceptions.

f) Modify and Save the Hello Program

For this next activity, you will modify the program you created above. You will make two changes.

- Line 1: Have the program greet you by your first name, for example, Hello, Moto! instead of Hello, world!.
- Line 2: a println statements with some interesting or fun general information
- Lines 3-5: at least 3 lines of information about you.

SAMPLE OUTPUT

```
Hello, Nazli!  
McDonald's once made bubblegum-flavored broccoli.
```

I had so much fun this summer
My family and I went white-water rafting at Glacier National Park
We climbed Mt. Arikok which is filled with cacti and mountain goats

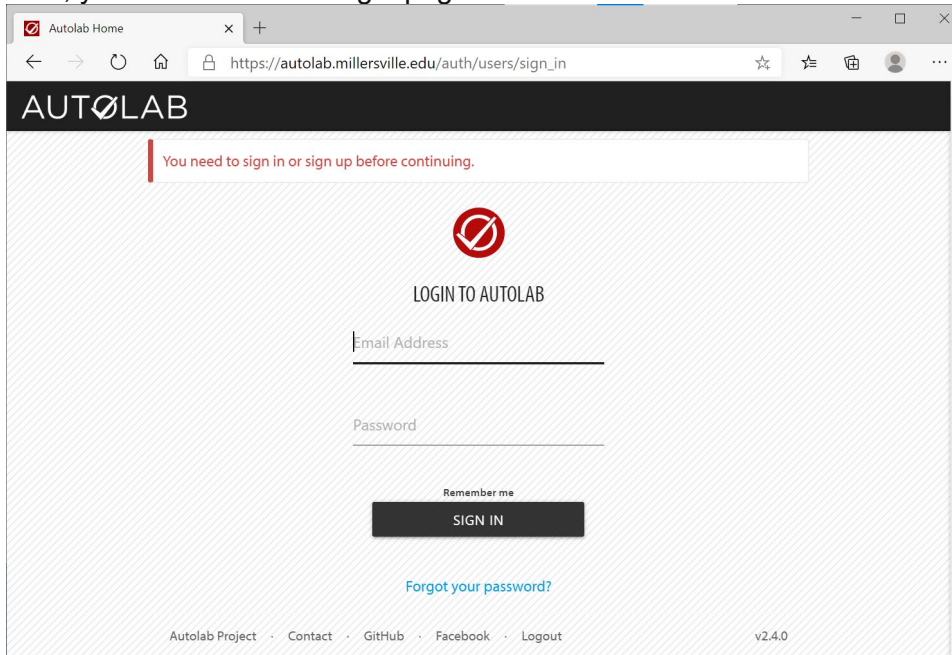
RUN THE PROGRAM

This time, you can just *click the Run button*.

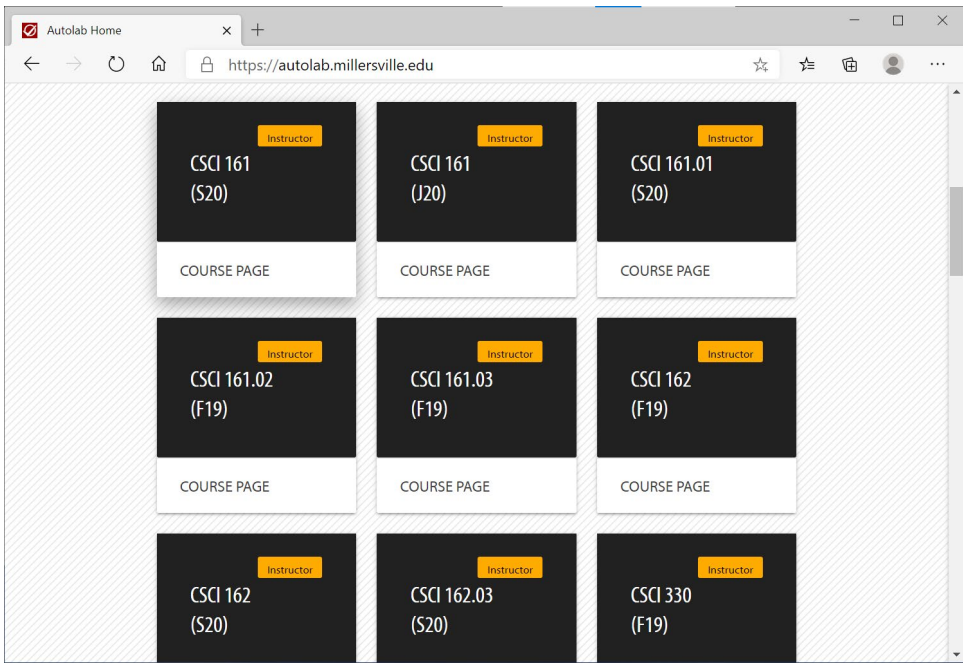
g) **Submit your lab using Autolab**

We will be using Autolab <https://autolab.millersville.edu> for assignment submissions this semester.

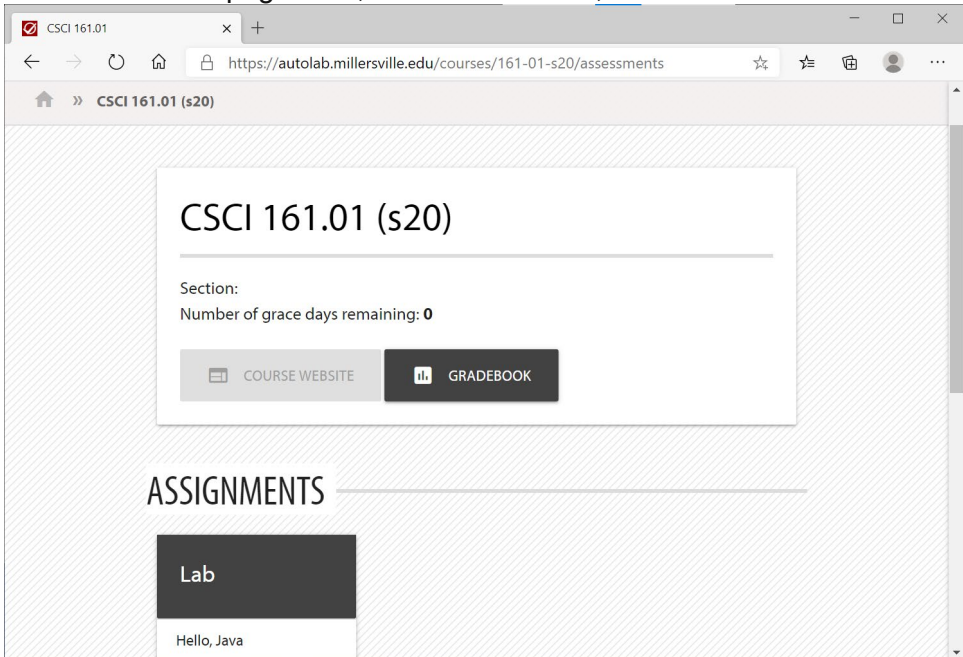
First, you should see the login page for Autolab.



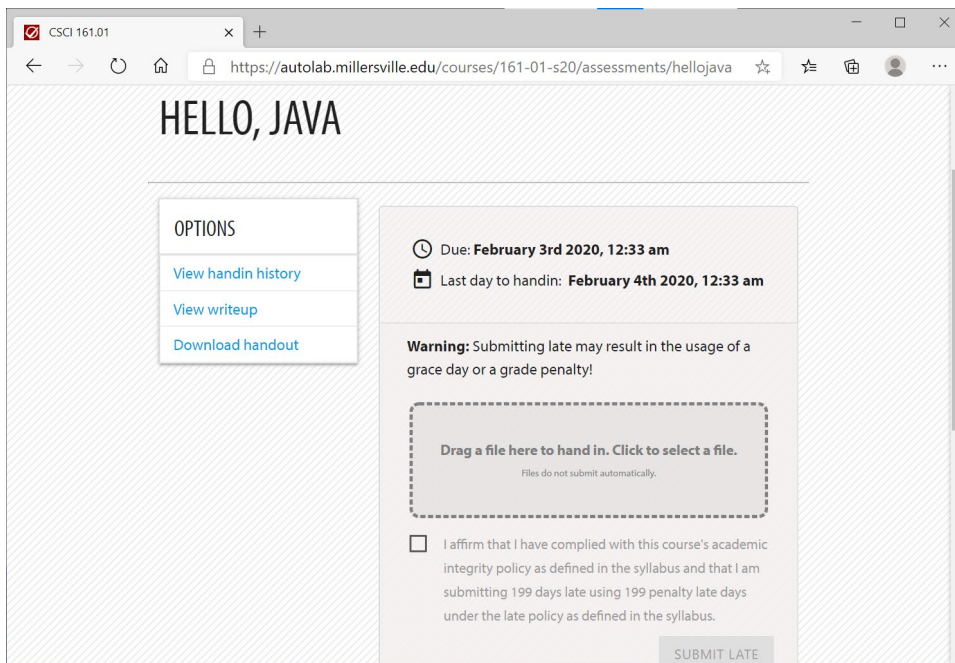
Next, you should see a list of all courses you are currently enrolled in. *Click on our course*.



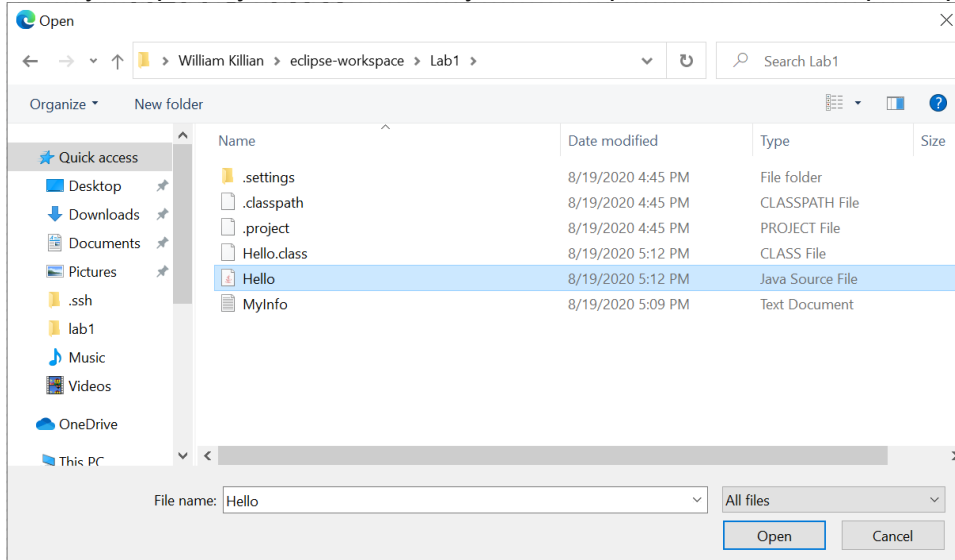
From the course page view, click on the Hello, Java link.



The "Hello, Java" assignment page should then be shown. To submit your Java file, *click the Submit File button* and browse to the location of your Hello.java file and *click Choose/Okay*.



Once you upload your submission, you will be presented with an upload page.



You will be able to view your score after waiting about 10-15 seconds. You may click on the Score link to see the output of the autograder. You can also navigate back to the Hello, Java assignment page and scroll down to view all submissions.

The screenshot shows a web browser window with the URL <https://autolab.millersville.edu/courses/161-01-s20/assessments/hellojav...>. The page title is "Handin History". The navigation bar includes "Gradebook", "Jobs", "Manage Course", "Manage Autolab", and "The Admin". The breadcrumb trail is "CSCI 161.01 (s20) » Hello, Java » Handin History".

There is a link for "View Gradesheet". Below it is a table with the following columns: Ver, File, Submission Date, auto (80.0), style (20.0), Late Days Used, Total Score, Tweak, and Errors.

| Ver | File | Submission Date | auto (80.0) | style (20.0) | Late Days Used | Total Score | Tweak | Errors |
|-----|--|---------------------|-------------|--------------|----------------|-------------|-------|--|
| 1 | william.killian@gmail.com_1_Hello.java | 2020-08-19 17:25:04 | 40.0 | -- | -- | -- | -- | (Regrade) (Destroy) (Edit) |

Page loaded in 0.016980135 seconds

Autolab Project · Contact · GitHub · Facebook · Logout v2.4.0

Congratulations! You have completed the first lab!

GRADING CRITERIA

- 20 pts – documented, well-formatted code (not part of autolab)
- 20 pts – statement printing "Hello, YourNameHere"
- 20 pts – statement printing another line of text
- 40 pts – at least three additional statements including information about you