Hey, wanna come watch the total solar eclipse?
No thanks. I've got the only eclipse that matters.

Meme Credit: Thomas Rachman, Any Person Any Meme (Facebook)

Lab 1: Getting Started

CS 2112 Fall 2020

September 7 / 9, 2020

Lab Staff

Monday	Wednesday
Ted Bauer*	Shiyuan Huang
Ashneel Das	Annabel Lin*
Changyuan Lin	Bryan Tabor
Michael Xing	Michael Xing

* Remote

Lab Modalities

In-Person

- Attend in-person every week
- Must be registered as in-person
- Assigned seats, masks, social distancing, etc.

Online

- Attend online every week
- You can join even if you're signed up to be in person
- One lab each week will be recorded (for time zone reasons)

Versions

We recommend Eclipse 2020-06. We require Java 11.

Any relatively new version of Eclipse should be okay.

Can I Use IntelliJ?

Yes.

However, note that not all members of course staff will be able to support you if you run into difficulties.

Specifically, please direct all questions about IntelliJ to the following people:

Ted Bauer	Sam Sorenson
Changyuan Lin	Michael Xing
Charles Sherk	Sam Zhou

Can I Use Another IDE?

Yes, but we will not be able to provide any technical support.
If something breaks, you're on your own.

Uninstalling Java

We recommend completely removing all prior versions of Java from your system, to avoid potential conflicts.

Windows
Open Settings → Apps
Select your Java install(s) and choose Uninstall

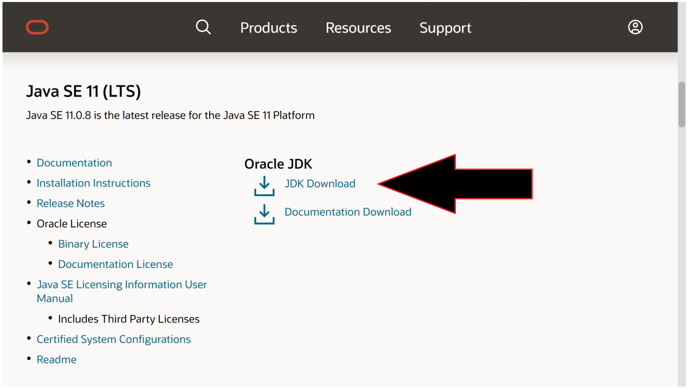
macOS
Navigate to
`/Library/Java/JavaVirtualMachines`
(From Finder, choose Go → Computer, then Macintosh HD, then Library, etc.)
Delete everything

Linux: Instructions will vary depending on your distribution

If you had an older version of Eclipse on your computer, you may want to uninstall that too.

Step 0

Go to https://www.oracle.com/java/technologies/javase-downloads.html to download and install the Java 11 JDK.



Java SE 11 (LTS)

Java SE 11.0.8 is the latest release for the Java SE 11 Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
  - Binary License
  - Documentation License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

Oracle JDK

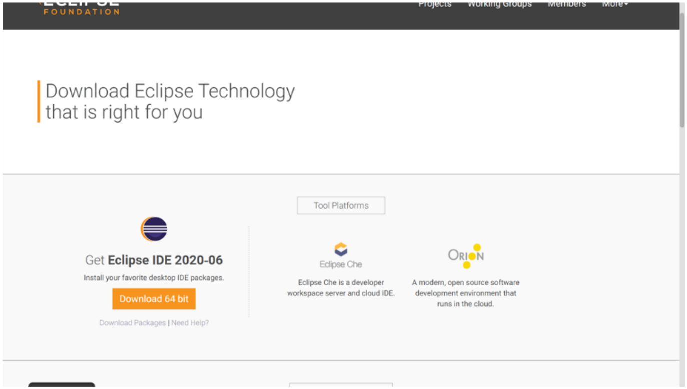
JDK Download

Documentation Download

We recommend downloading the Oracle JDK, as students have had issues with OpenJDK in the past.

Step 1

Go to www.eclipse.org/downloads



Download Eclipse Technology that is right for you

Get **Eclipse IDE 2020-06**

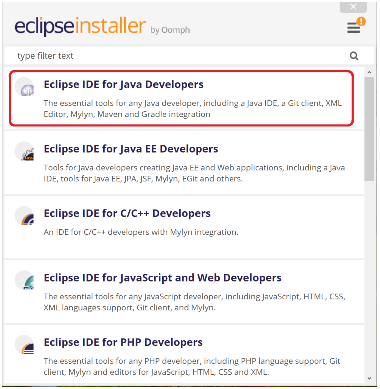
Install your favorite desktop IDE packages.

Eclipse Che is a developer workspace server and cloud IDE.

A modern, open source software development environment that runs in the cloud.

Step 2

Run the file as an admin, let it load (it'll take a while), then



Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Eclipse IDE for Java EE Developers
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

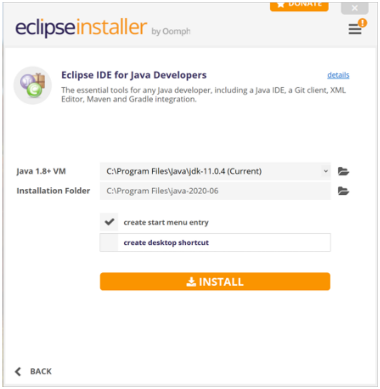
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.

Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.

Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Step 3

Ensure the Eclipse installer is pointed to the Java 11 JDK before starting the install process



The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.

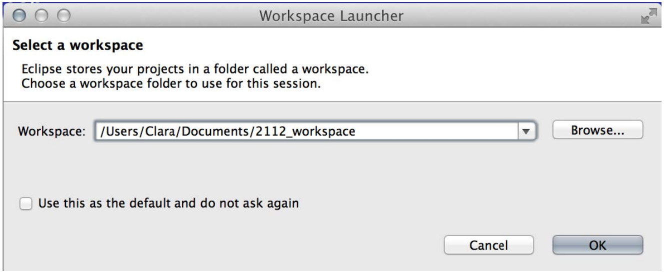
Java 1.8+ VM: C:\Program Files\Java\jdk-11.0.4 (Current)
Installation Folder: C:\Program Files\java-2020-06

✓ create start menu entry
create desktop shortcut

INSTALL

Step 4

Select a location for your workspace
This is where all of your projects will be stored



Select a workspace

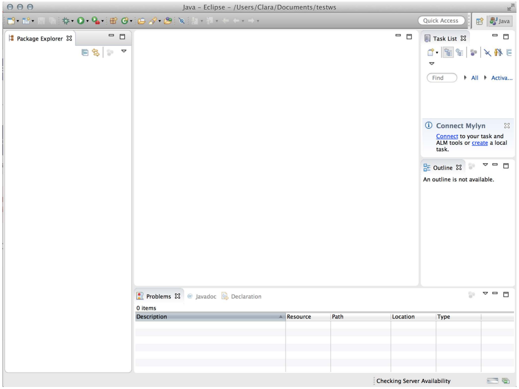
Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /Users/Clara/Documents/2112_workspace

We recommend choosing a folder on your desktop or somewhere you will remember.

Step 5

Close the Welcome screen. This is the default Java Perspective.



Connect to your task and ALM tools or create a local task.

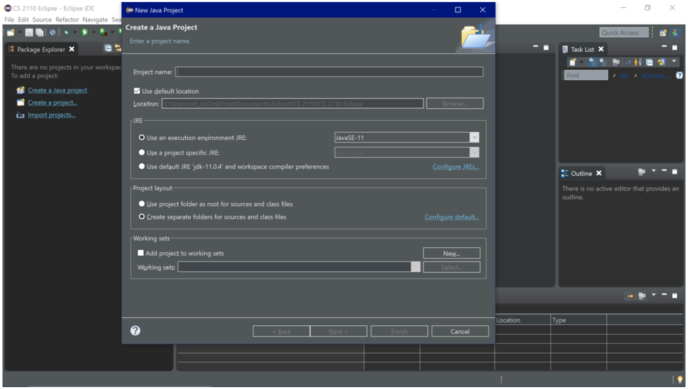
An outline is not available.

0 items

Checking Server Availability

Make A New Project

Make sure to select Java 11 as the execution environment.



New Java Project

Create a Java Project

Enter a project name.

Project name:

☑ Use default location

Location: C:\Users\mt_xi\OneDrive\Documents\School\CS 2110\CS 2110 Eclipse

JRE

- [✓ Use an execution environment JRE: JavaSE-11]
- Use a project specific JRE: jdk-11.0.4
- Use default JRE 'jdk-11.0.4' and workspace compiler preferences

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

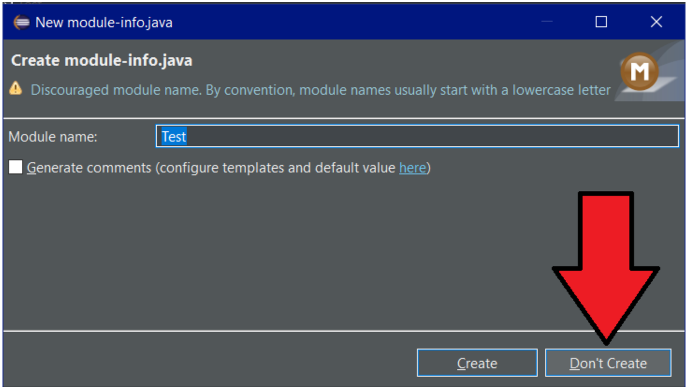
Working sets:

There are no projects in your workspace. To add a project:

There is no active editor that provides an outline.

Modules

Click "Don't Create" when asked to create the file `module-info.java`.



New module-info.java

Create module-info.java

⚠ Discouraged module name. By convention, module names usually start with a lowercase letter

Module name: Test

☐ Generate comments (configure templates and default value here)

Hello World

```java
public static void main(String[] args) {
    System.out.println("Hello World");
}
```

Coding Exercise

There is a 4-digit number which, when the order of its digits is reversed, yields a number 4 times greater.
Write a program to find and print out this number.

Useful Features Of Eclipse

- Autocomplete
- Autoindent
- Compile & Run
- Refactoring
- Autoformat (Ctrl + Shift + F)
  - We suggest turning on autoformat on save
  - Preferences > Java > Editor > Save Actions > Perform the selected actions on save > Format Source Code > Format edited lines
- Javadoc
- Comments

I/O Handout

A detailed reference on I/O can be found in the I/O handout on the course webpage:
https://courses.cs.cornell.edu/cs2112/2020fa/handouts/IO.pdf

Paths

A path represents the location of a file, typically on your computer.
eg: `C:\Users\Andrew\Documents\CS 2112\Lab 1.tex`

Types of Paths

There are two types of paths: absolute and relative.

Absolute Paths

- Starting at root, full path of file
- Usually only works on your machine
- eg: `C:\Users\Andrew\Documents \CS 2112\Lab 1.tex`

Relative Paths

- Relative to current directory
- In Eclipse, project folder
- Typically used when programming
- eg: `Documents\CS 2112\Lab 1.tex` (if we're in the `Andrew` directory)

Using Paths in Java

You can call `Paths.get(...)` with a relative path to acquire a `Path` object, which represents the location of a file.

```java
Path p = Paths.get("res", "map1.xml");
```

The above code returns a reference to the relative path `res/map1.xml`.
Note you can seperate directories as separate arguments, or pass an entire relative path in.

Files

Once you have a path to a file, Java provides many methods that allow you to operate on it, listed under the `Files` class.

eg: `exists(Path p)`, `isReadable(Path p)`, `createFile(Path p)`, `delete(Path p)`, `isWritable(Path p)`, `size(Path p)`, and more.

Check the official documentation for more:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/nio/file/Files.html

Streams

A **stream** is a sequence of data being processed (read / written) from beginning to end.
Input streams are data coming into a program (for example, reading from a file).
Output streams are data leaving a program (for example, writing to a file).

Types of Streams

- Byte Stream
- Character Stream
- Raw Stream
- Blocking Stream
- Buffered Stream
- NIO Stream
- Object Stream
- etc.

Basic Streams

Reads one byte at a time.

```java
InputStream is = Files.newInputStream(p);
is.read(); // Gets the next byte in the file
```

We can use a **Buffered Stream** to get more than one byte at a time, for convenience.
Remember to always **close** a stream when finished working with it.

Buffered Readers

```java
InputStream is = Files.newInputStream(p);
BufferedReader br = new BufferedReader(is);
                     // or
BufferedReader br = Files.newBufferedReader(p);

// read whole line (or null if empty)
String s = br.readLine();
br.close(); // close stream
```

Buffered Writers

```
BufferedWriter bw = Files.newBufferedWriter(p);
// Overwrites p if exists, creates if not

bw.write("..."); // No newline
bw.close(); // Don't forget
```

Use a `PrintWriter` to write non-String objects and get additional methods.

```
PrintWriter pw =
    new PrintWriter(Files.newBufferedWriter(p));
pw.println(6); // Includes newline
```

Standard Streams

Your OS provides every program with three "standard" I/O streams. These streams have defaults, but can be changed per program. For example, a user may want to redirect standard error into a log file instead of showing it in the console.

Standard Input: What the user types into your program, typically in the console.

Standard Output: What your program shows to the user, typically in the console.

Standard Error: Error messages from your program, typically in red in the console.

Standard Streams in Java

Java exposes each of the standard streams to the programmer as fields in the `System` class: `System.in`, `System.out`, and `System.err`.

Standard input is an `InputStream`, and the other two are `PrintWriter`.

Thus, `System.out.println("")` is calling the `println("")` method on a `PrintWriter` that just happens to be standard output.

Character Encoding

Character encoding defines how characters we recognize get stored to disk as individual bytes.

For this class, use Unicode **UTF-8**.

I/O Exercise

Write a program to read user input from the console and print back the user input.
Feel free to reference the IO handout:
https://courses.cs.cornell.edu/cs2112/2020fa/handouts/IO.pdf

- Create a class with a main method
- Accept user input and echo it back

I/O Challenge Exercise

Find out what words are shared by two files, and return the number of unique words in common. Output the words you find to a different file.

```java
long wordsInCommon(File file1, File file2) {
    // TODO implement
}
```