Meme Credit: https://dzone.com/articles/intellij-idea-vs-eclipse-which-is-better-for-begin

Lab 1: Getting Started

CS 2112 Fall 2022

August 29 / August 31, 2022

Lab Staff

Monday

Omkar Bhalerao
Kevin Cui
Stanley Jiang
William Wang
Jerry Xu

Wednesday

Noah Schiff
Charles Shrek
Esther Wang
William Wang
Jerry Xu

Reminders

- Get on Ed

Course Structure

Lecture + Discussion + Lab

Lab Modalities

In-Person

- Attend in-person every week
- Masks encouraged

Versions

We recommend IntelliJ
(any relatively new version should be acceptable)

Community should be fine for our needs, but you can download Ultimate for free with your Cornell e-mail (see Ed).

We require Java 11.

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.

Step 0 - Downloading IntelliJ

Go to

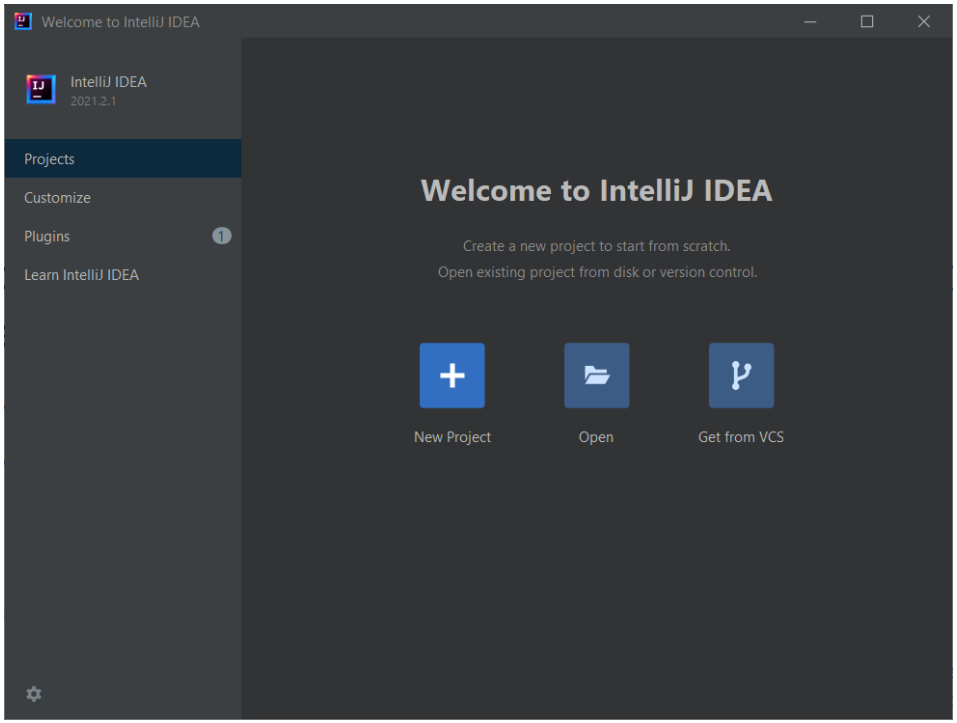
www.jetbrains.com/help/idea/installation-guide.html

Follow the directions for "Standalone Install" for your OS

Linux: Use your appropriate package manager

Step 1 - First IntelliJ Run

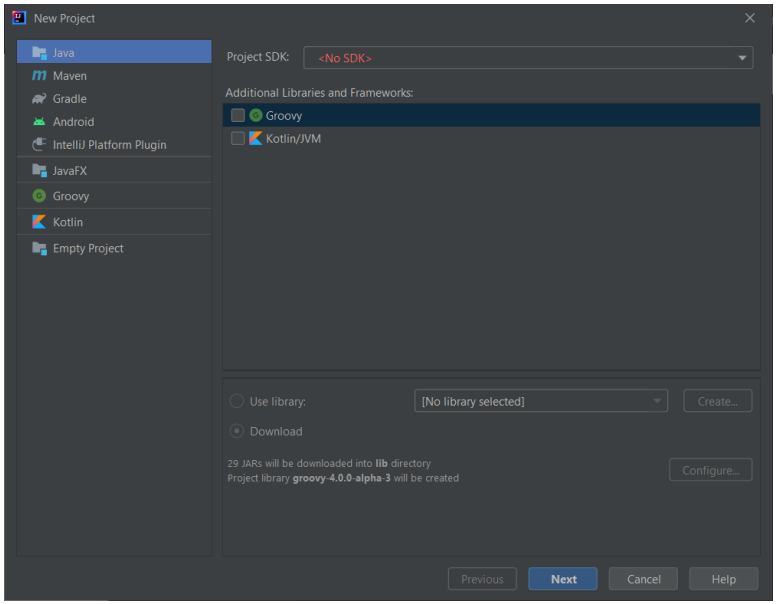
You should come to a screen like this:



Click ‘New Project’

Step 2 - Download a JDK

You should see this:

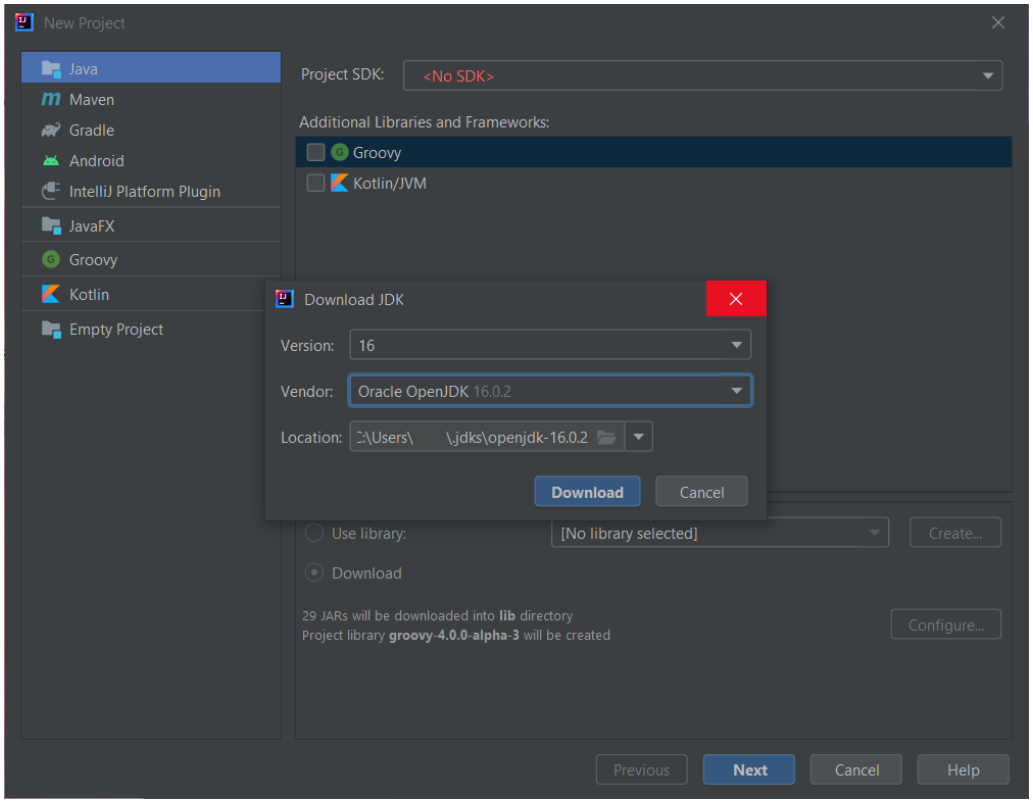


you might have a JDK already installed that's been detected

We're going to re-download a fresh one, though.

Step 2 - Download a JDK

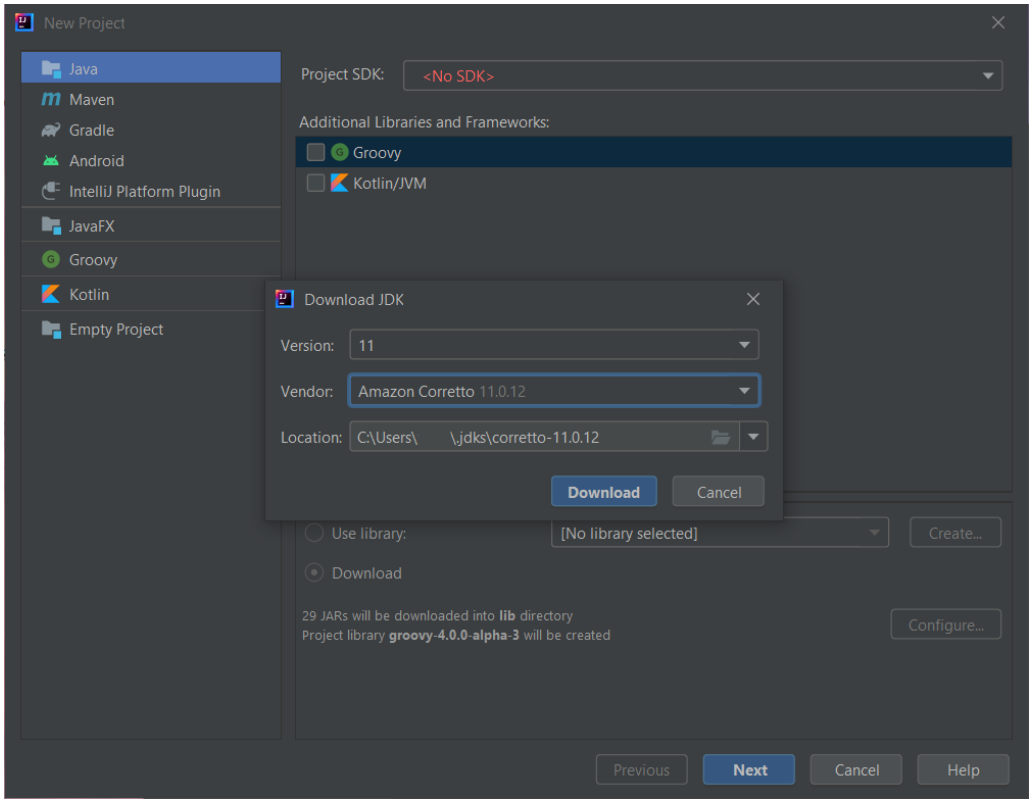
Click on the 'Project SDK' dialog box



Step 2 - Download a JDK

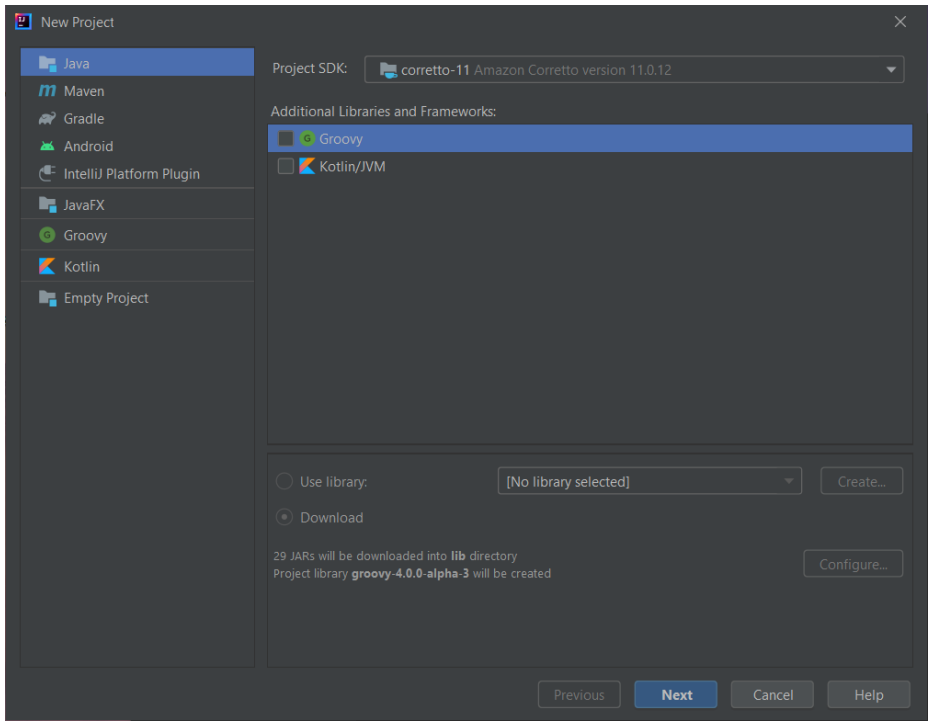
First select **Version 11**

Then, select **Amazon Corretto** as the distribution



Step 3 - Make A New Project

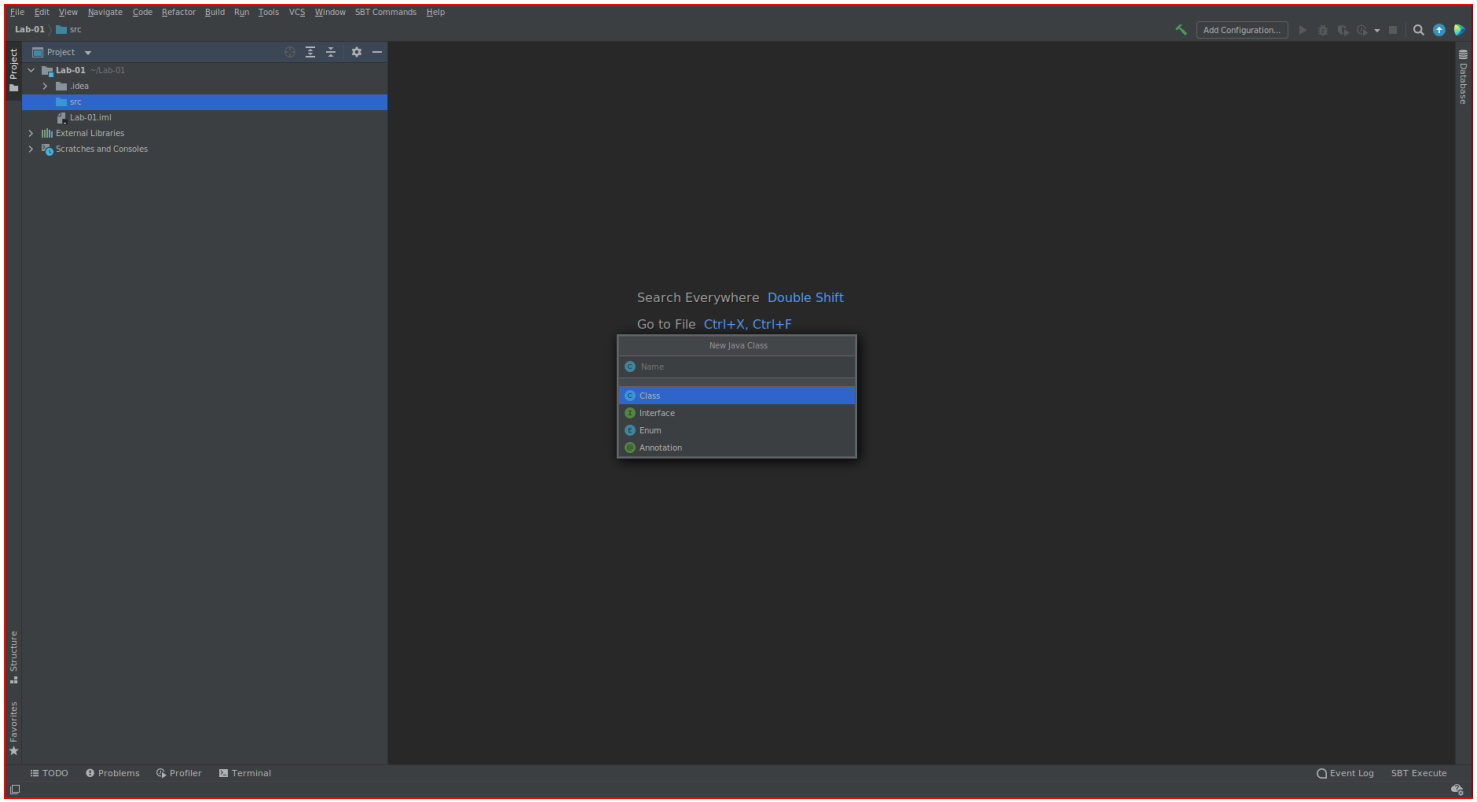
After you've downloaded the JDK, wait for it to auto-detect.
Make sure you've selected Corretto 11.



Click through the dialogs - we'll be creating a blank project.

Add a file

Right click on the blue src folder, and go to New → Java Class



Name the class Main (or whatever you like)

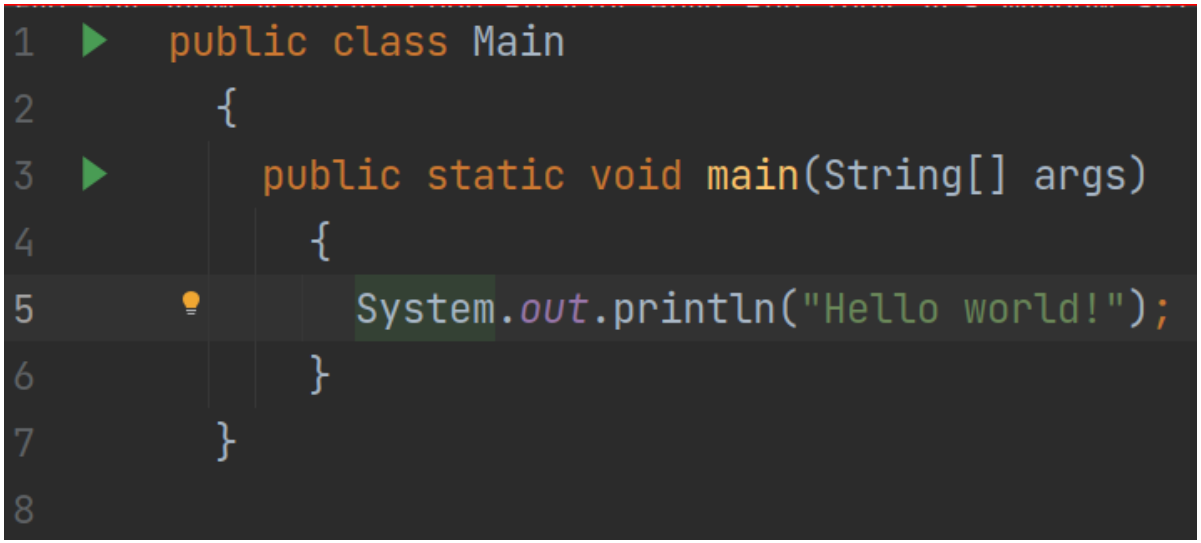
Hello World

Feel the power of IntelliJ by typing "psvm<TAB>" to autocomplete a boilerplate main method.

```
1 public static void main(String[] args) {
2     System.out.println("Hello World");
3 }
```

Running your code

Click one of the green triangles on the sidebar



After the first run, you can also use the Run/Debug toolbar on the top right

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 IntelliJ

- **Auto-save**
- Autocomplete
- Autoindent
- Compile & Run
- Refactoring
- Autoformat (Ctrl + Alt + L/Opt + Cmd + L)
  - We suggest turning on autoformat on (auto)save
  - Autosave should be enabled by default, but you can also save manually with Ctrl + S or Cmd + S
  - File > Settings > Tools > Actions on Save > Check "Reformat Code" > Select "Changed lines" instead of "Whole file"
  - If your group members don't have the same format, you will get terrible merge conflicts
- 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/cs2112/2022fa/handouts/IO.pdf

Paths

A path represents the location of a file, typically on your computer.

e.g.,: `C:\Users\Nate\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
- e.g.,:

  `C:\Users\Nate\Documents\CS 2112\Lab 1.tex`

Relative Paths

- Relative to current directory
- In IntelliJ, project folder
- Typically used when programming
- e.g.,:

  `Documents\CS 2112\Lab 1.tex`

  (if we're in the `Nate` 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.

```
1 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.

```
1 InputStream is = Files.newInputStream(p);
2 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

```
1  InputStream is = Files.newInputStream(p);
2  BufferedReader br = new BufferedReader(is);
3                    // or
4  BufferedReader br = Files.newBufferedReader(p);
5
6  // read whole line (or null if empty)
7  String s = br.readLine();
8  br.close(); // close stream
```

Buffered Writers

```
1 BufferedWriter bw = Files.newBufferedWriter(p);
2 // Overwrites p if exists, creates if not
3
4 bw.write("..."); // No newline
5 bw.close(); // Don't forget
```

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

```
1 PrintWriter pw =
2     new PrintWriter(Files.newBufferedWriter(p));
3 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/cs2112/2022fa/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.

```
1 long wordsInCommon(File file1, File file2) {
2     // TODO implement
3 }
```