IntelliJ Setup Guide

Introduction

Important: If you already have IntelliJ installed (because you've taken CS0150 or for any other reason), please follow the section Configuring IntelliJ and Configuring IntelliJ settings to make sure your IntelliJ is configured for the settings we need in this course.

If you are using IntelliJ from a department machine, see this section for instructions on how to run it.

IntelliJ is an IDE (Integrated Development Environment) for Java, one of the two languages we will be using in CS200. Similar to DrRacket in CS17, code.pyret.org in CS19 or PyCharm in CS111, IntelliJ is an incredibly useful tool that does syntax highlighting, error-checking, and much more. It is *highly* recommended that you install IntelliJ and use it throughout CS200, as this will be the IDE that the TAs have practice with. *You may use a different Java IDE if you wish, but note that our staff will only support/help with IntelliJ.*

In this document, you will find a step-by-step guide for installing and setting up IntelliJ on your local machine or using a department machine. Note that you only need to do this a single time on each machine.

Installing IntelliJ

To install IntelliJ, do the following:

1. Go to https://www.jetbrains.com/idea/download to visit the IntelliJ download page, then select your operating system version ("Mac", "Windows", or "Linux")
2. Under "Ultimate" (make sure you select the Apple Silicon option if you have an M1 Mac), click the "Download" button. **If you have an M1 or M2 mac (how to check), be sure to select the "Apple Silicon" version by clicking on the right half of the download button.**
3. Once the application has downloaded, install and open it and accept the license agreement.
4. If you have not installed IntelliJ before, a window will pop up asking you to provide a license. JetBrains (the company that produces IntelliJ) provides free licenses for students. If you do not already have a JetBrains account from another class (or have not used it in the past year), do the following:
    a. Go to https://www.jetbrains.com/student/ and click the **"Apply Now"** button toward the bottom of the page. Follow the instructions on the page to use your brown.edu email address to verify you are a student and obtain an account and license.

b. Once the instructions prompt you to do so, go back to IntelliJ and log in with your JetBrains account to get your license. After you log in, IntelliJ should show that it found your license: click **"Activate"** then continue to start the IDE!

5. A window might pop up asking if you want to import any settings. If you have not used IntelliJ before, select **"Do not import settings"** and accept any other defaults while installing. If you have used IntelliJ in the past and would like to import settings, you can choose to do so.
6. You should now see a window "Welcome to IntelliJ IDEA" prompting you to create or open a project. To continue with setup, you can select **"New project"**, leave all of the settings as-is, and click **"Create".** If you get a warning saying that no SDK is installed, just click **"Continue"** (this is what we'll resolve in the next step!).

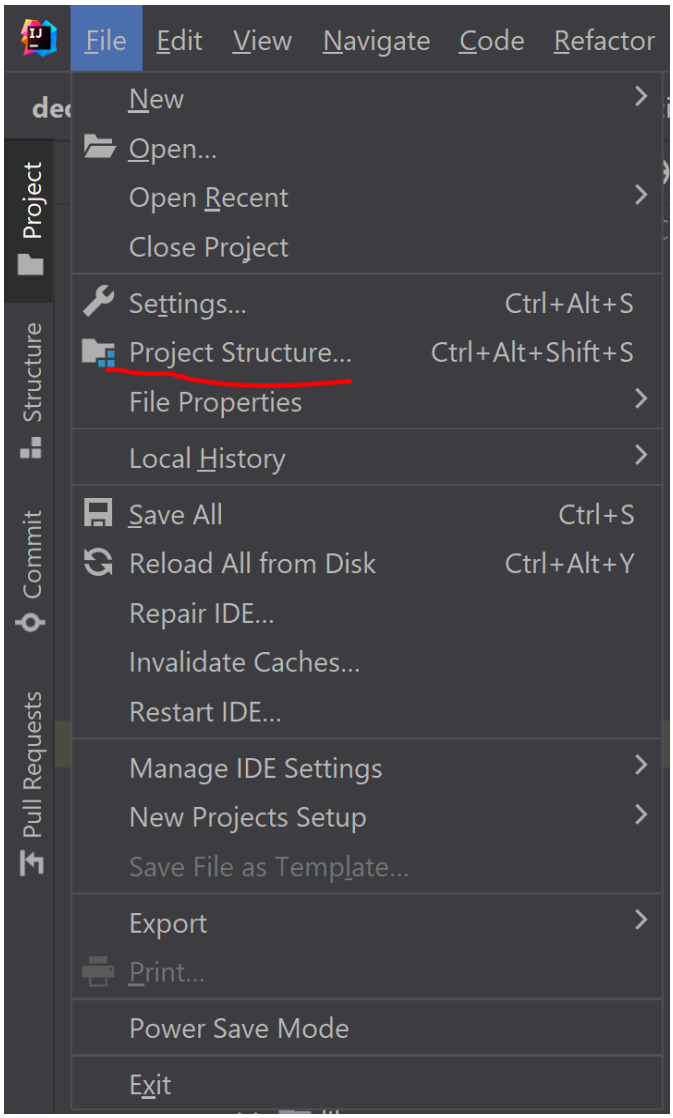
Configuring IntelliJ (and installing Java)

IntelliJ is only an IDE for Java; it is not Java itself. To actually run Java programs, you need to install a Java SDK (Software Development Kit), which contains the tools to build and run Java programs.

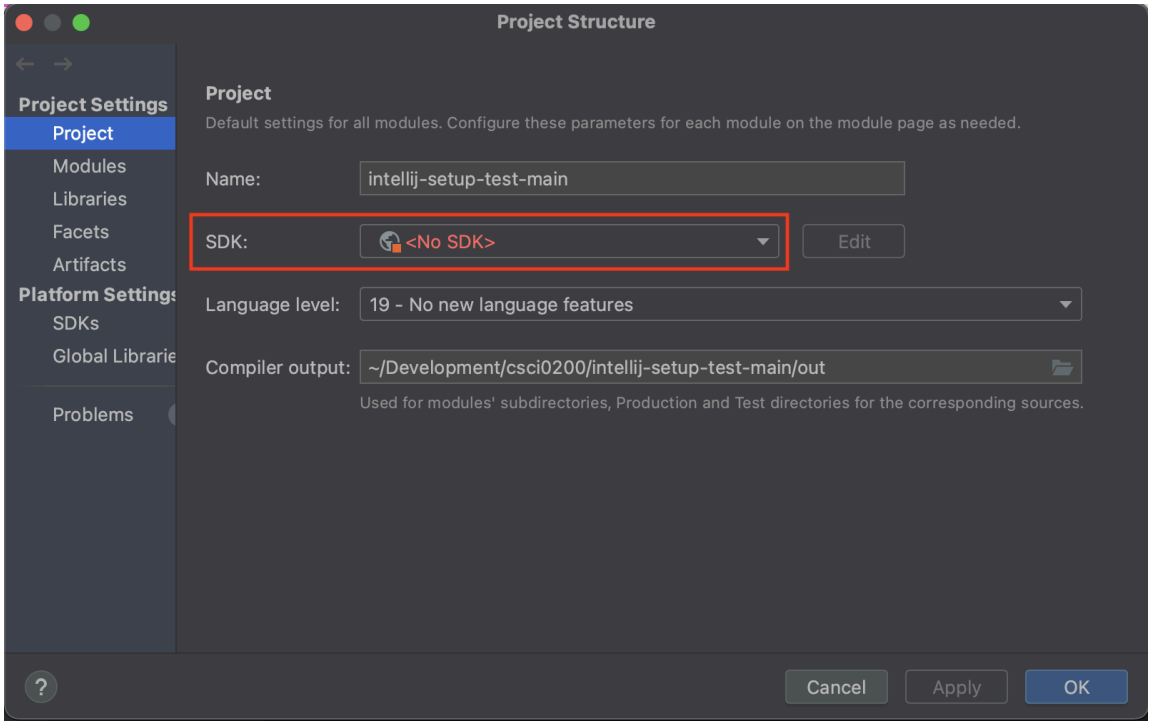
> ***Note:*** For this class, we will use Java SDK version 17 or higher. The current version is 19, so if you are installing Java for the first time, you will use this version. Otherwise, if you have Java 17 or 18 installed, it's okay to use these instead.

To install a Java SDK, do the following:

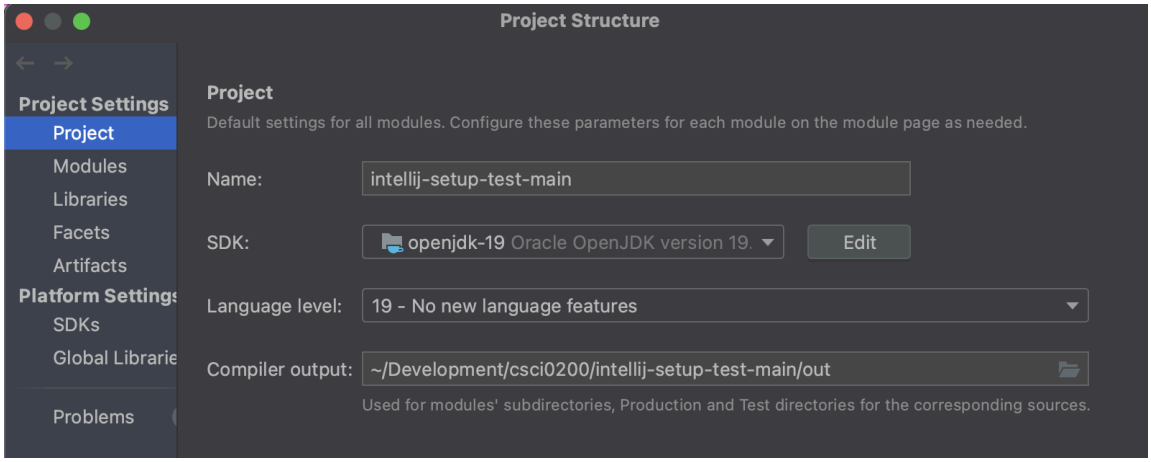
1. First, open any IntelliJ project. If you just installed IntelliJ from the previous section, you should already have a project open. Otherwise, you can continue by opening any repository for this class. See this section of the Github guide for instructions on cloning assignments from Github classroom–return to this guide once you have cloned a project into a folder on your system.
2. With IntelliJ open, go to **File > Project Structure** in your toolbar (Mac Users can use ⌘+; as a shortcut), which should look like this:



3. When the project structure window opens, look at contents of the SDK field, as shown below:



4. If the field says "<No SDK>" or any version lower than 17, click on the dropdown and look at the options:
    a. **If you see an entry for Java 17, 18, or 19 in the list**, you already have Java installed–select it to set your SDK to this version.
    b. If you don't see any SDK entries, select **"Add SDK…"** > **"Download JDK…"** and select Oracle OpenJDK 19 (which should be the default), then click **"Download".** This should install Java on your system!
5. Once your JDK has been configured, your project structure window should look something like this:



6. To save your changes, click **"Apply"**, then **"OK"**.
7. Each time you load an IntelliJ project, you may need to set your JDK like this (though you won't need to install it again). Keep this process in mind so that you can return to it later!

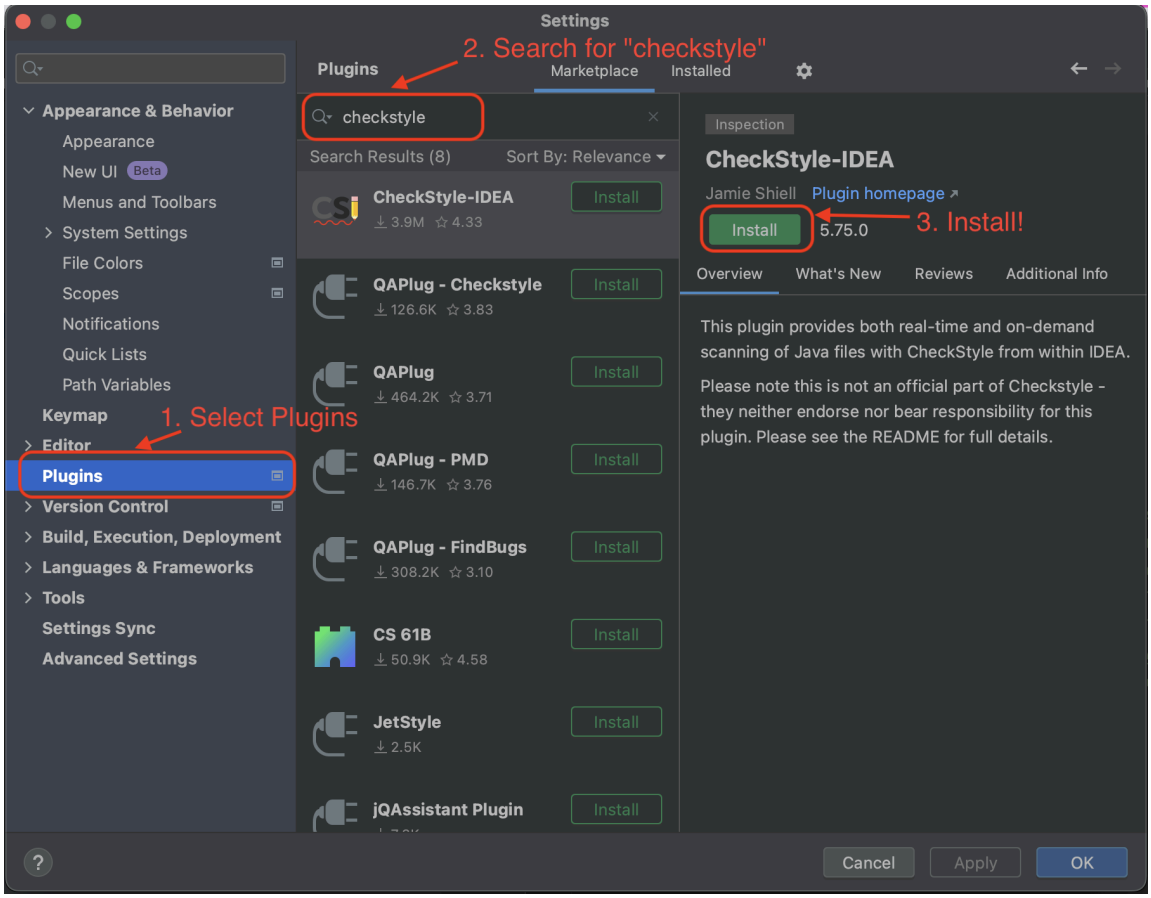
Configuring IntelliJ Settings

Now that you have IntelliJ and Java installed we are going to edit a few IntelliJ settings to make your life a little easier and to maintain consistency with the course's style guide.

Installing Checkstyle

We will use an IntelliJ plugin called checkstyle which will automatically check to make sure your code conforms to the course style guidelines. To install Checkstyle:

1. Open the IntelliJ settings menu. On a Mac, in the menu bar at the top, go to **IntelliJ Idea > Preferences (**or **IntelliJ Idea > Settings).** On Windows or Linux, go to **File > Settings..**
2. In the sidebar, select **"Plugins"** and type "checkstyle" in the search box. The first item in the list should be called **"Checkstyle IDEA"**, as shown in the figure below. Click "Install" to install the plugin–you can accept any warnings about third-party plugins.

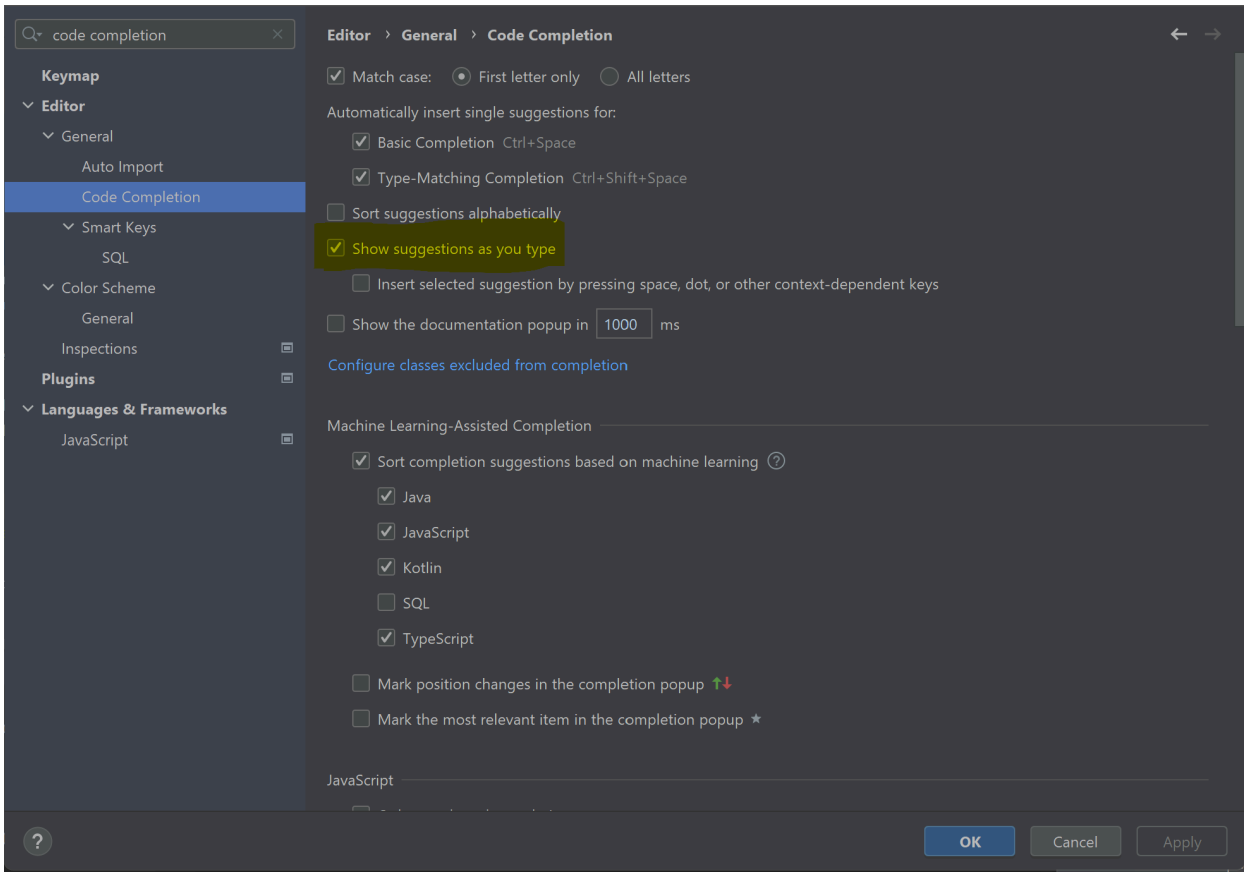


3. Once installed, IntelliJ may need to restart–if so, click **"Restart"**.

Configuring autocomplete

Next, we will configure autocomplete, which will automatically make suggestions for your code as you type. To do this:

1. Open the IntelliJ settings menu. On a Mac, in the menu bar at the top, go to **IntelliJ Idea > Preferences (**or **IntelliJ Idea > Settings).** On Windows or Linux, go to **File > Settings.**
2. In the search box at the top left, search for “code completion” and click on the **“Code completion”** page in the sidebar on the left.
3. On this page, check the box for **“Show suggestions as you type”**, which should look like this:



After this step, your IDE should be set up for developing in Java! Yay!

The remainder of this document has instructions on working with IntelliJ, and an FAQ section for common bugs. If you’re setting up IntelliJ for the first time in this class, you can stop here for now–but remember that you can return to this guide later!

Running Assignments

Directory Structure

Each assignment (hw, labs, and projects) will be packaged on GitHub as its own project. To keep your projects organized, we recommend you have a ‘CS200’ folder stored in an accessible place on your computer where you can download (or “clone” in git terms) for each project.

For ease of access, we recommend creating your CS200 folder on your Desktop (Mac) or C: drive (Windows), but you can use whatever location is most convenient for you!

Working with Projects

Now that you have your development environment set up, it's time to start coding! To do this:

To pull code from GitHub, follow the Github Guide to clone your projects. Then, you can open the project with IntelliJ with `File > Open`, and then select the folder that you downloaded.

> ***Note:*** If for whatever reason you need to create your own projects without our stencil code, go to **Projects** > **New Project**. You will need to select a Project SDK, which should be whichever version of Java you have (but at least Java 17). Give your project a convenient name and make sure to save it in the proper directory (like your CS200 folder).

Configurations, Compiling, and Running

Java programs in CS200 tend to use multiple files that will interact with each other (with containment/association, for example). While this makes organizing your code much easier, IntelliJ won't know which file you want to run when you press the primary play button until you tell it.

The easy way of doing this is by pressing the green play button to the left of the line of code in a file you want to run (for instance, to the left of the main method), or right clicking on the file name in the file structure and clicking "`Run '<filename>'`". The primary play button will compile and run this file and associated files until you run a different main file.

```
39 ▶ ⊟    public static void main(String[] args) { Tester.run(new ListTest()); }
42    }
43
```

> ***Note:*** In CS15, we started you off with the terminal commands "javac *.java" and "java <filename>" to compile and run. THIS WILL NOT WORK because you are now using a different version of java and it got wonky, so please stick to the IntelliJ compiling/running process outlined in this section.

> ***Note:*** When running more complicated projects, instead of using the inline play button, you might need to input arguments. If this is the case, go to Edit/Add configurations, which can be done by selecting the 'Add/Edit Configurations' button at the top right of your window to the left of the green run arrow and selecting the appropriate SDK (Java 18) and telling IntelliJ where your `main` function is. More information can be found here.

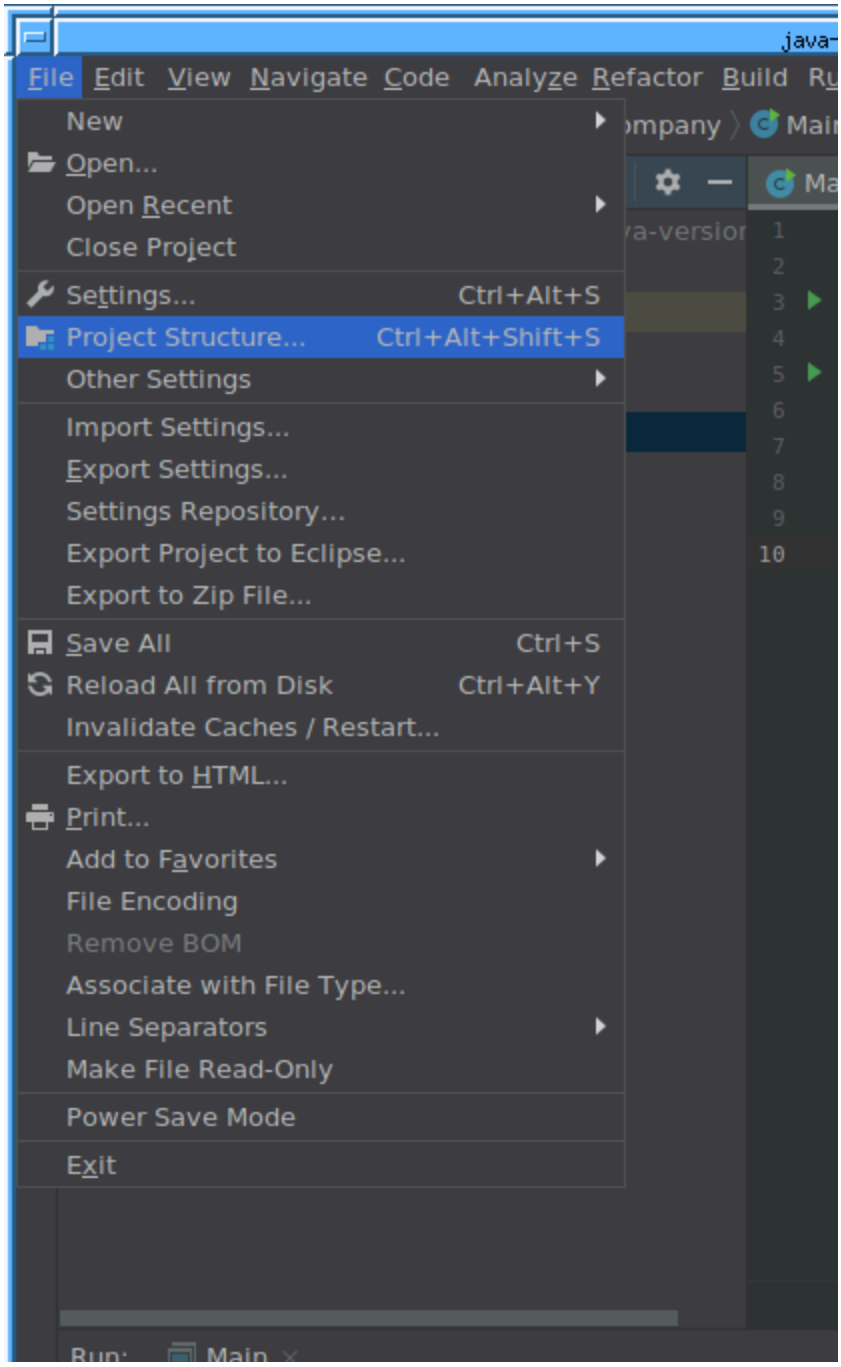
- Assignments that require you to do this will have more detailed instructions on what configuration you should be adding.

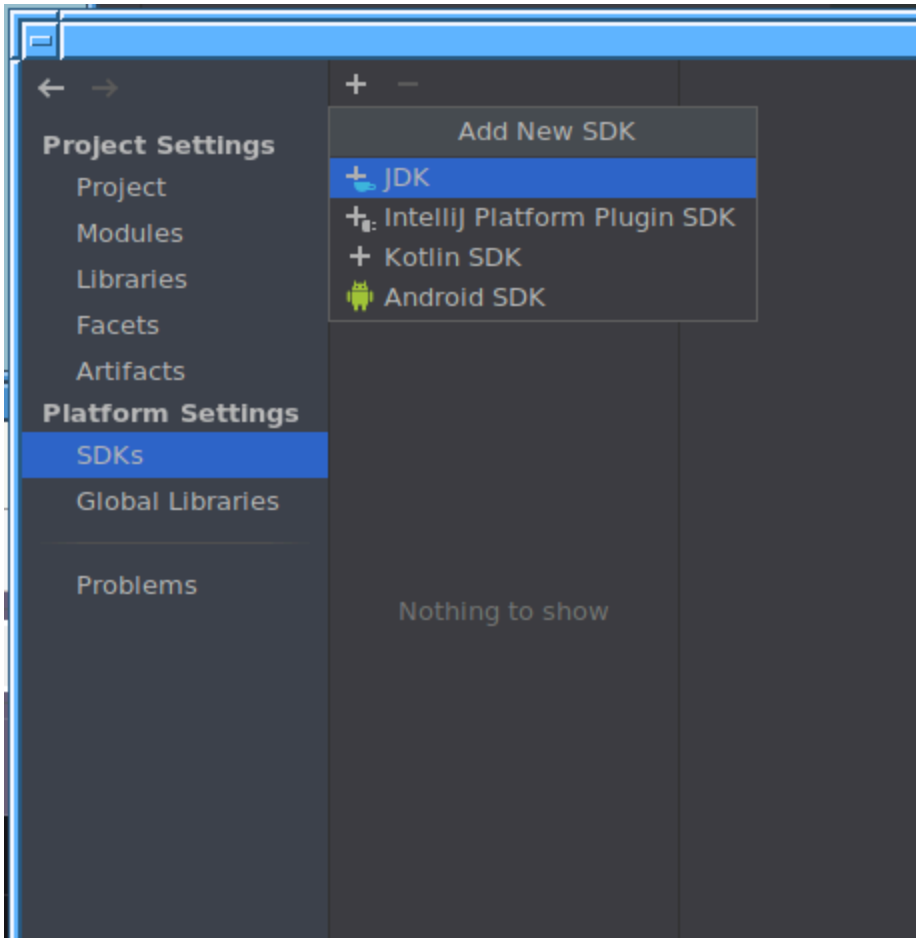
Department Machine Setup

For some students, connecting to the department machines (which use Linux) will be a better option than working on your personal computer. You can connect to the machines using this guide, or use one of the computers in person at the CIT.

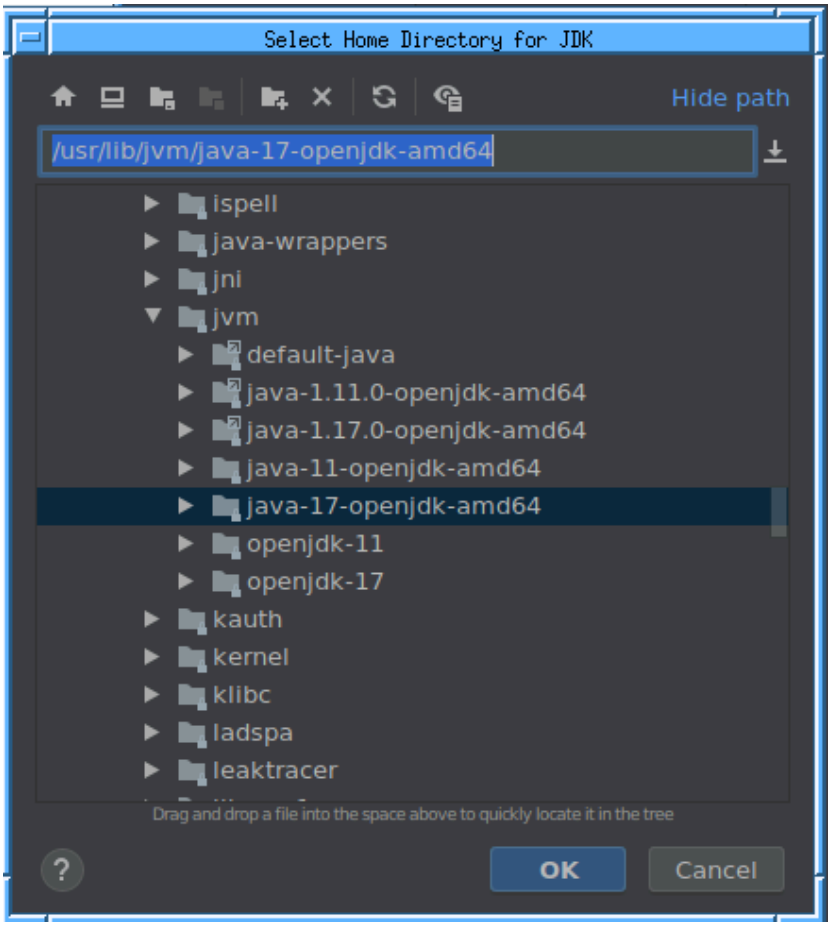
For working on a department machine, you can launch IntelliJ from a terminal by typing "`intellij &`". Once you do that, all you need to do is specify Java 18 for your project SDK.

Go to `File > Project Structure > SDKs`, and click the "+" icon.


File Edit View Navigate Code Analyze Refactor Build
New
Open...
Open Recent
Close Project
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
Other Settings
Import Settings...
Export Settings...
Settings Repository...
Export Project to Eclipse...
Export to Zip File...
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Invalidate Caches / Restart...
Export to HTML...
Print...
Add to Favorites
File Encoding
Remove BOM
Associate with File Type...
Line Separators
Make File Read-Only
Power Save Mode
Exit
Run: Main


Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
Add New SDK
JDK
IntelliJ Platform Plugin SDK
Kotlin SDK
Android SDK
Nothing to show

From there, you want to add the JDK at `/usr/lib/jvm/java-18-openjdk-amd64`.



Once you do this, you should be all set!

If you want to verify your setup, you can create a project and run this program and see if the output version is 18.0.x. (you may need to change com.company if it errors).

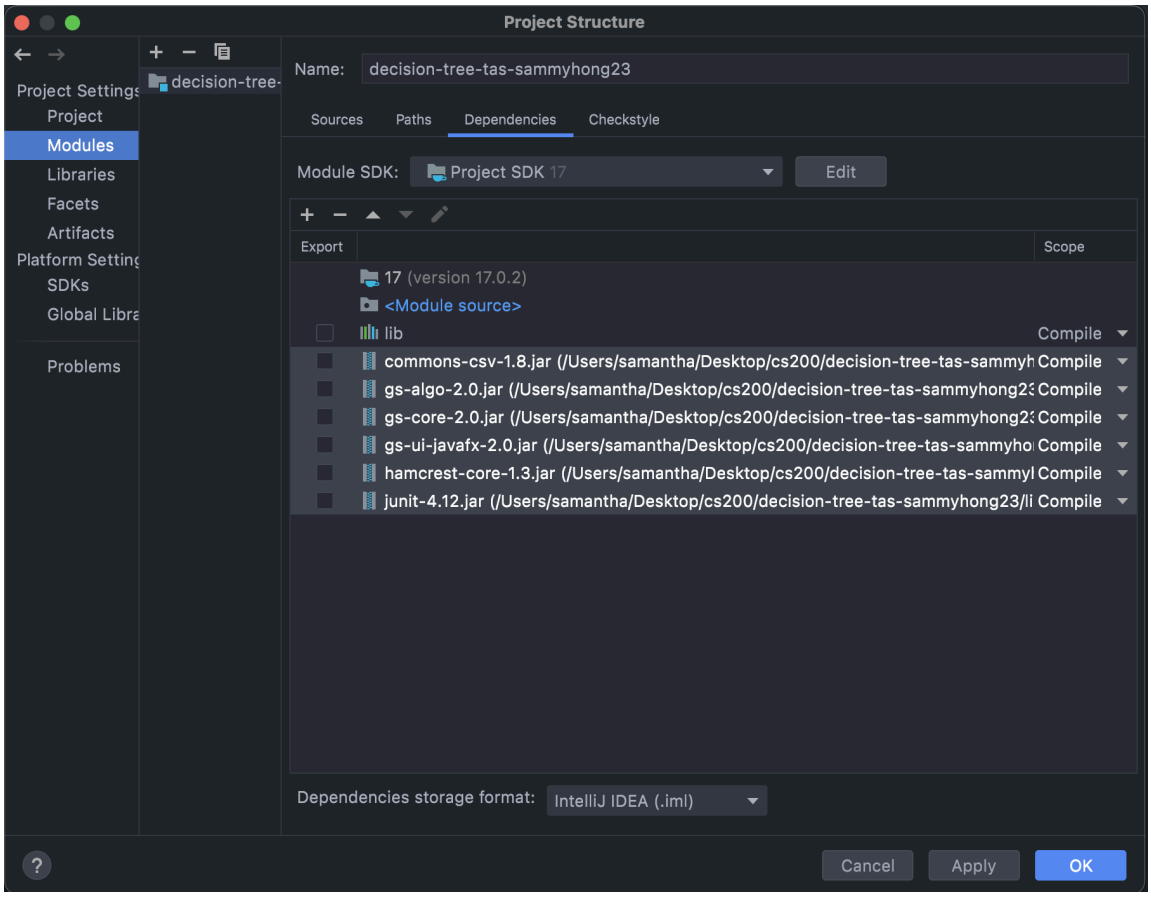
```
01| package com.company;
02|
03| public class Main {
04|
05|     public static void main(String[] args) {
06|    // write your code here
07|         System.out.println(System.getProperty("java.version"));
08|     }
09| }
```

If necessary: adding Jars/Dependencies

Most Java projects depend on support code found in libraries called “jar files.” IntelliJ needs to know where these files are in order to use them in your code.

Our stencil projects should already be configured to load our required jar files automatically. However, it may be necessary to add jar files to your IntelliJ project yourself. In our stencil projects, these files should be located in the `lib` folder in your cloned repository. To use them in IntelliJ, we need to add them as “Dependencies” of our project. To do this:

- Open the project window and select `File > Project Structure > Modules > Dependencies`.
- Next, select the plus (+) button, select “JARs or directories”, then locate and select the `.jar` files from the lib folder. Make sure under “Scope” that “Compile” is selected, then click “Apply” and “OK”. (It should look similar to the screenshot below).



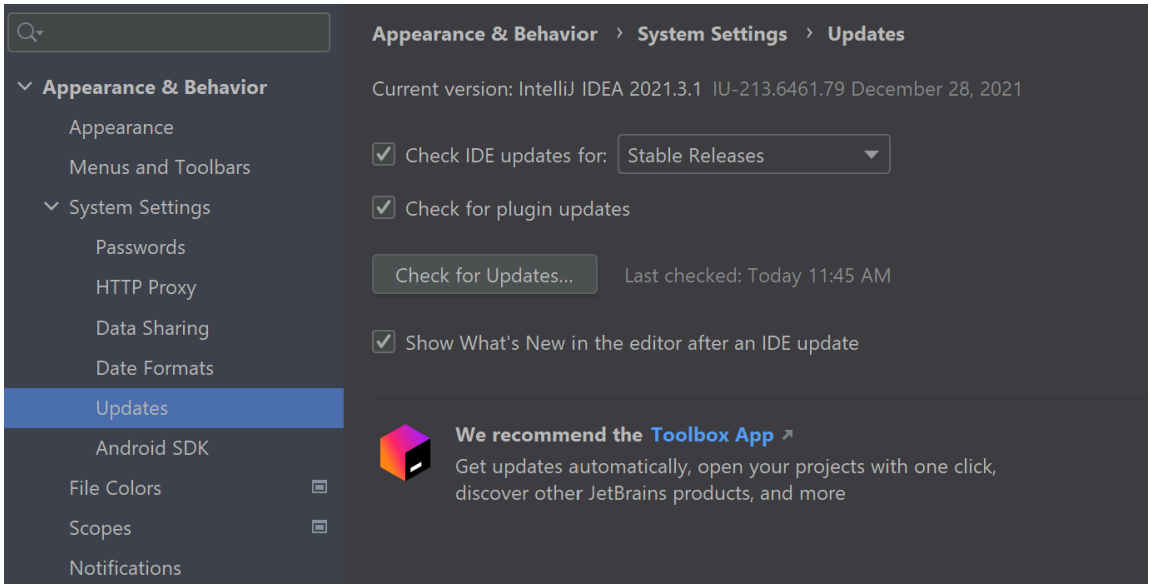
- Now, IntelliJ is set up to use the proper dependencies.

Note: *You will have to repeat these steps for each project.*

Common Bugs/FAQ

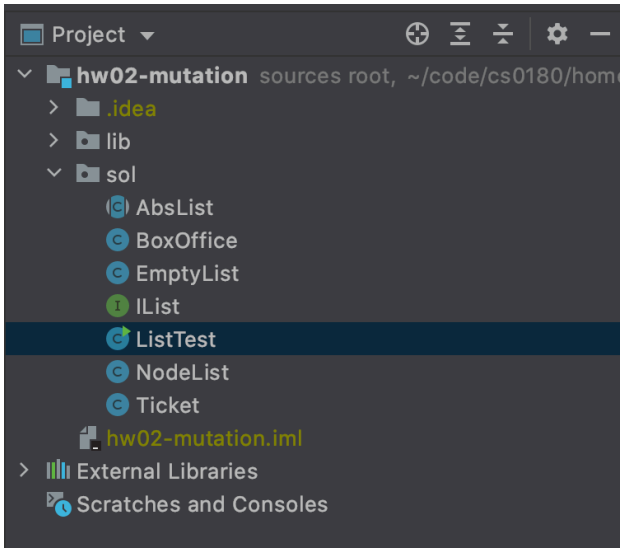
We will try to update this with bugs that we encounter during the semester!

- If you have previously installed IntelliJ, make sure that you have updated to the most recent version. See directly below for instructions on how to do this!
- **I'm getting an error like: "Cannot determine path to 'tools.jar' library."** Follow these steps to update your IntelliJ.

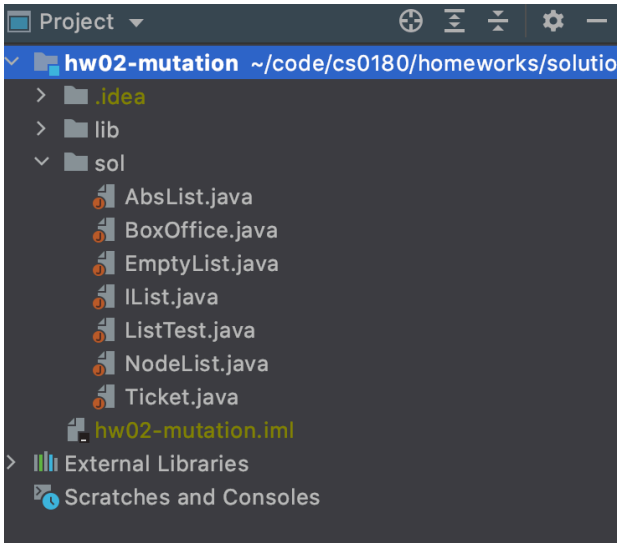


Above: To update IntelliJ go to File > Settings > Appearance & Behavior > System Settings > Updates and click Check for Updates. You may have to complete this step twice depending on when you first installed IntelliJ

- **Support code is showing up as red or raises errors after cloning it from Github.** Double check that you have correctly added all the .jar files from the lib folder as dependencies. See here for instructions.
- **IntelliJ is saying that there is an error with the sol package.** To solve this, go to File > Invalidate Caches/Restart, and if prompted click "Invalidate and Restart"
- **IntelliJ is confused about package names and/or imports.** Always check to make sure that the top folder in your IntelliJ project is the name of the GitHub assignment, and that it says "sources root":

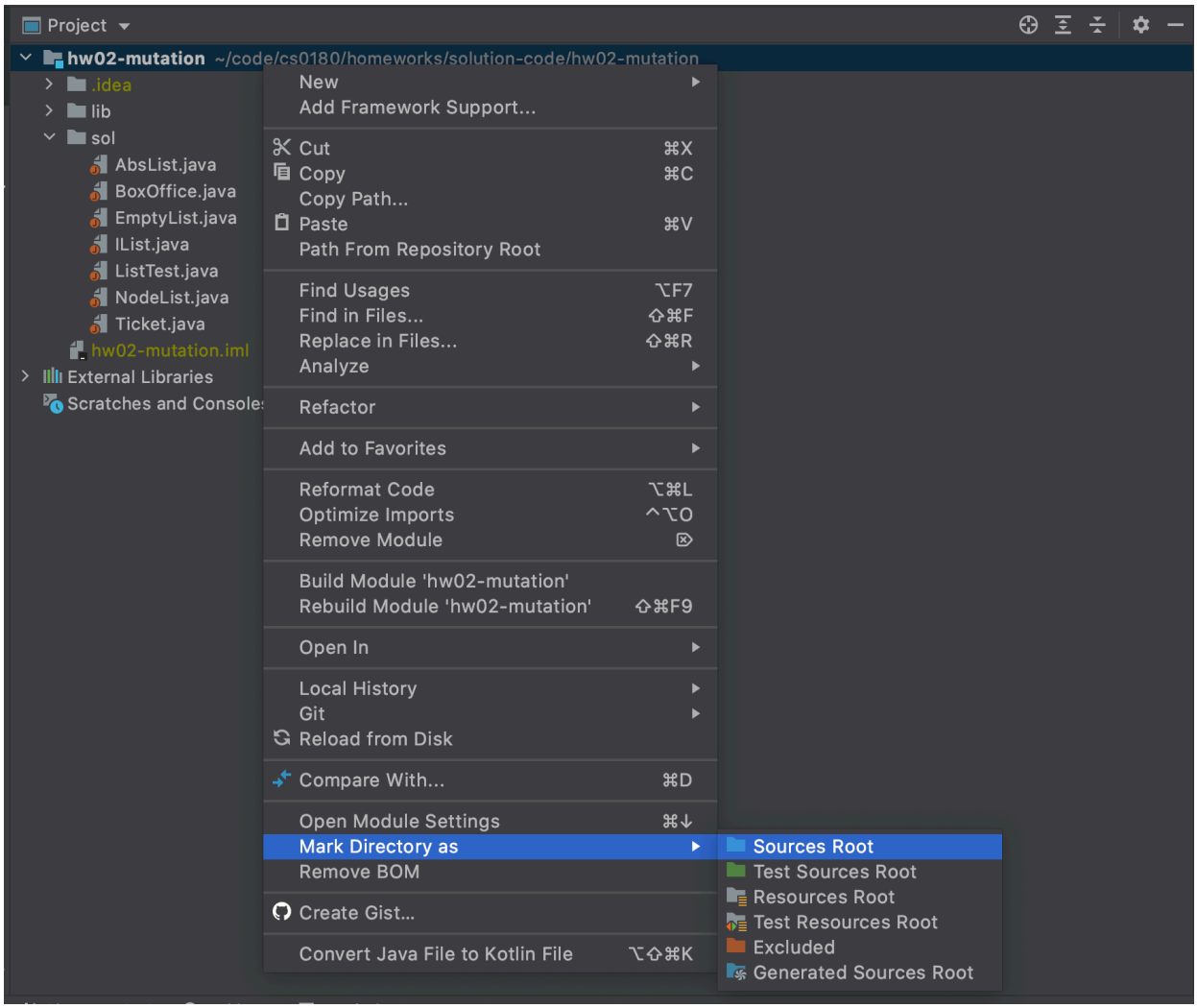


Above: correct project structure



Above: incorrect project structure (no "sources root" tag on top folder). Note the different icons on the .java files

If you are missing the sources root tag, right click on the top folder, and then select Mark Directory As > Sources Root:



- **I'm getting an error that says "Project JDK is not defined".** Click on "Setup SDK" and follow the prompts to select the Java SDK.

More Information

IntelliJ might look complicated at first, but it is an incredibly powerful tool once you get to know it. You can find more information with these helpful links:

- https://www.tutorialspoint.com/intellij_idea/intellij_idea_getting_familiar.htm
- https://blog.jetbrains.com/idea/2020/05/debugger-basics-in-intellij-idea/
- https://www.jetbrains.com/help/idea/running-applications.html

---

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS200 document by filling out the anonymous feedback form! (you do have to sign in but we don't see it)