

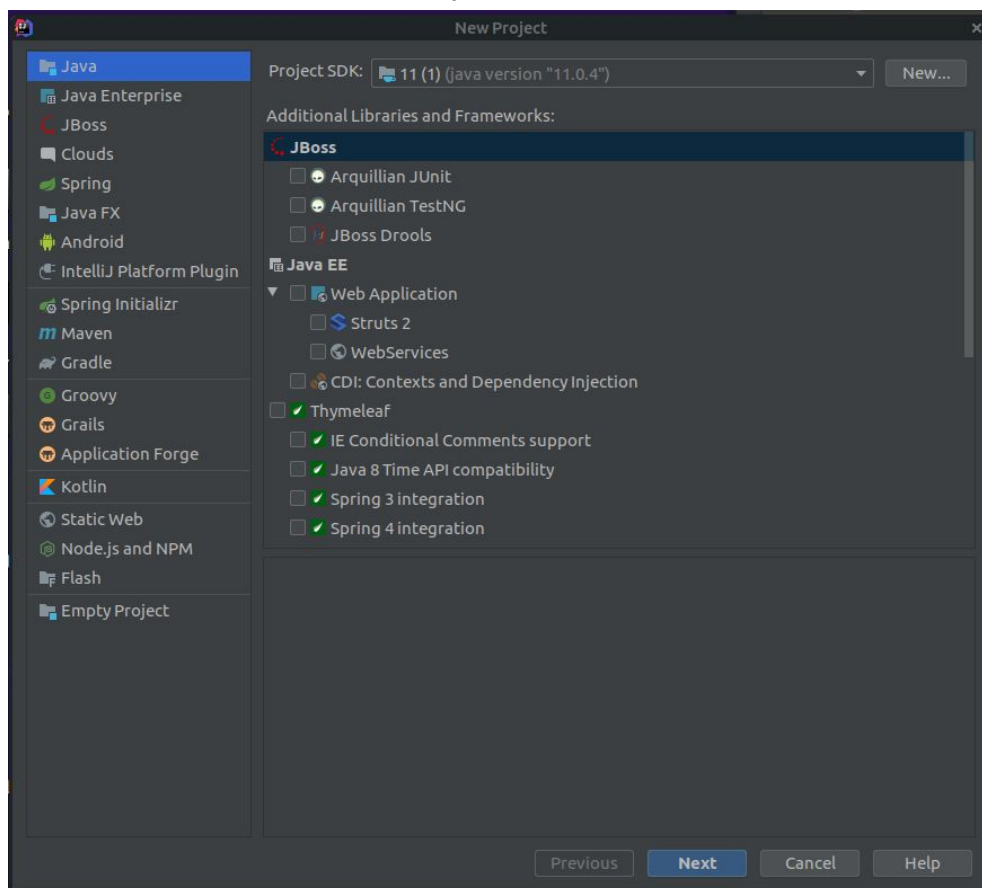
Setting Up Image Editor Unit Tests

Downloads

First you will need to visit the Projects section of the current CS 240 website and download the “Test Files (Zip)” folder. Inside this zip you will find 4 folders: jars, key_images, out_images, source_images, and test_files.

Create a Project

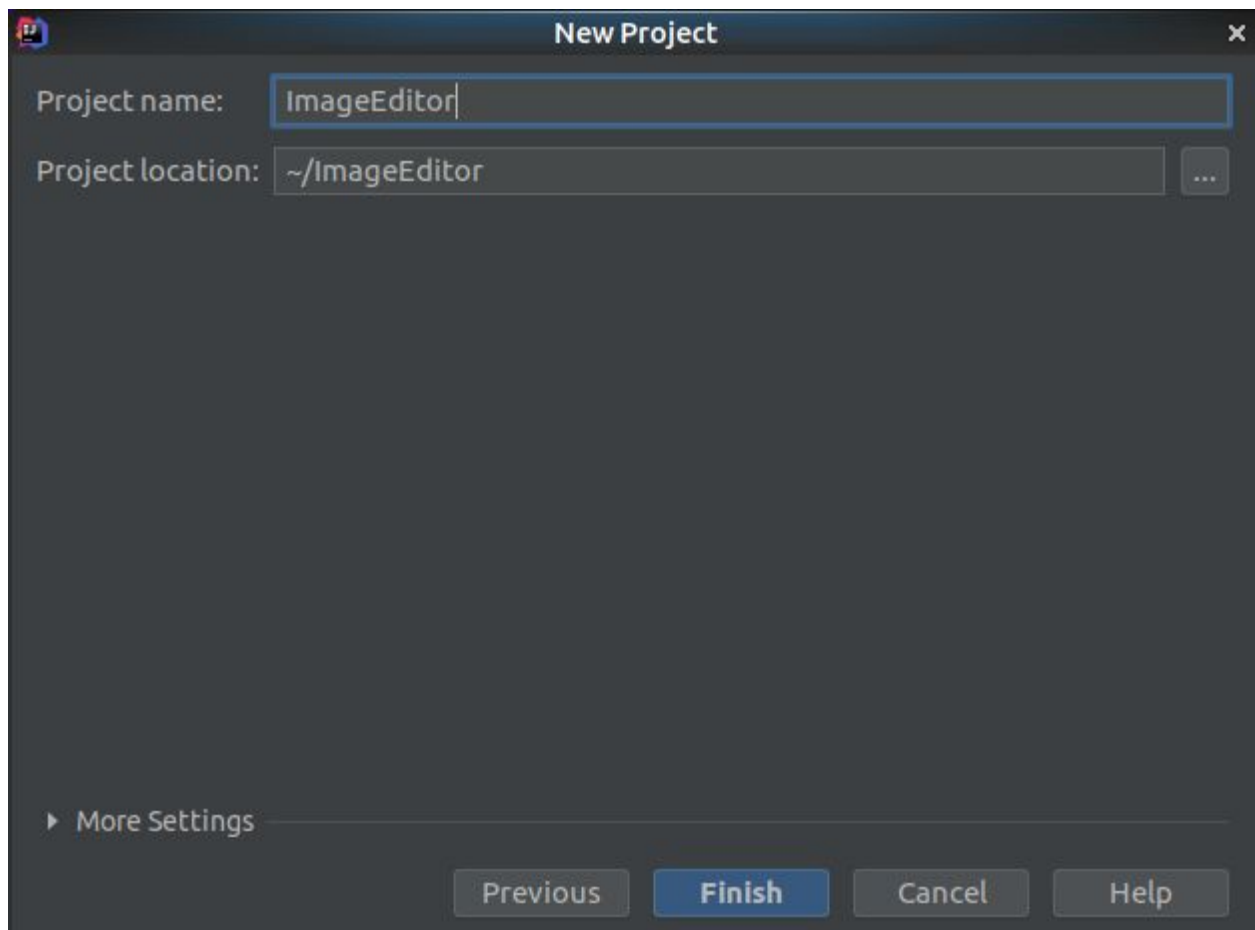
Open IntelliJ and create a new Project.



This tutorial will be setting up as a normal Java project. If you wish to create a Maven or Gradle project you will be responsible for setting it up on your own and getting dependencies working correctly.

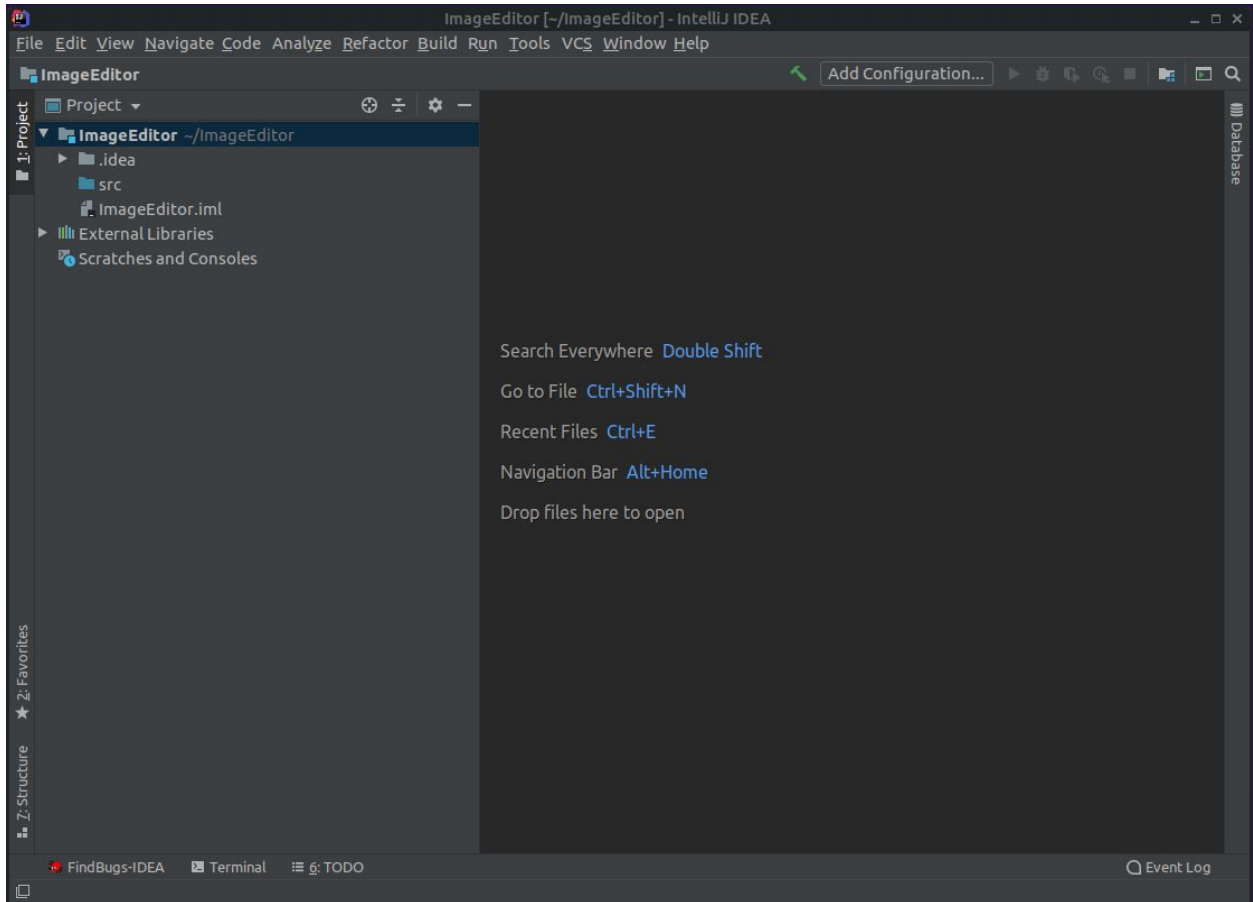
Select “Java” for your project type and in “Project SDK” select whatever version of Java you want (You should select whatever is the latest version of Java installed on your computer). You do not need to select any additional libraries or frameworks. Click “Next”.

Continue to click “Next” until you reach this window:

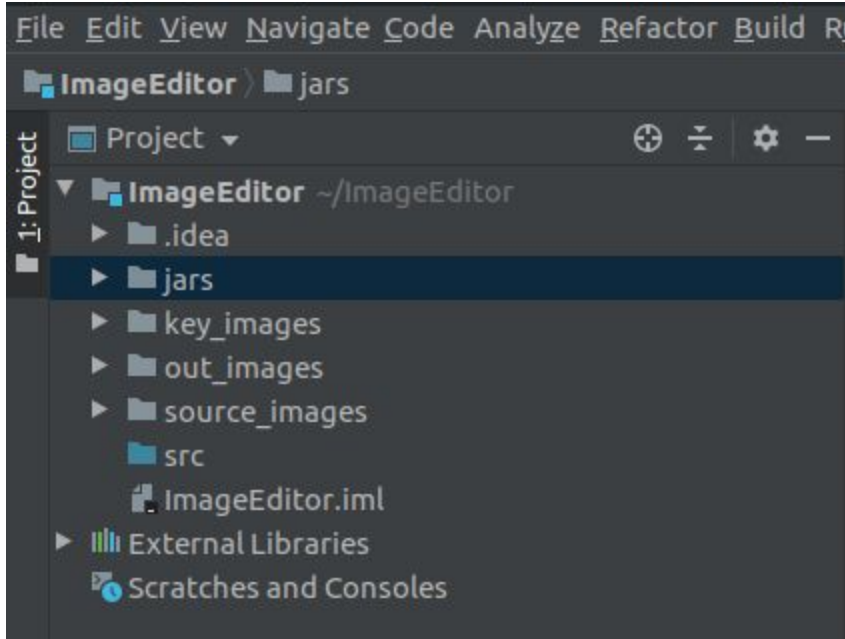


Name your project whatever you want though we suggest using ImageEditor. Click “Finish”

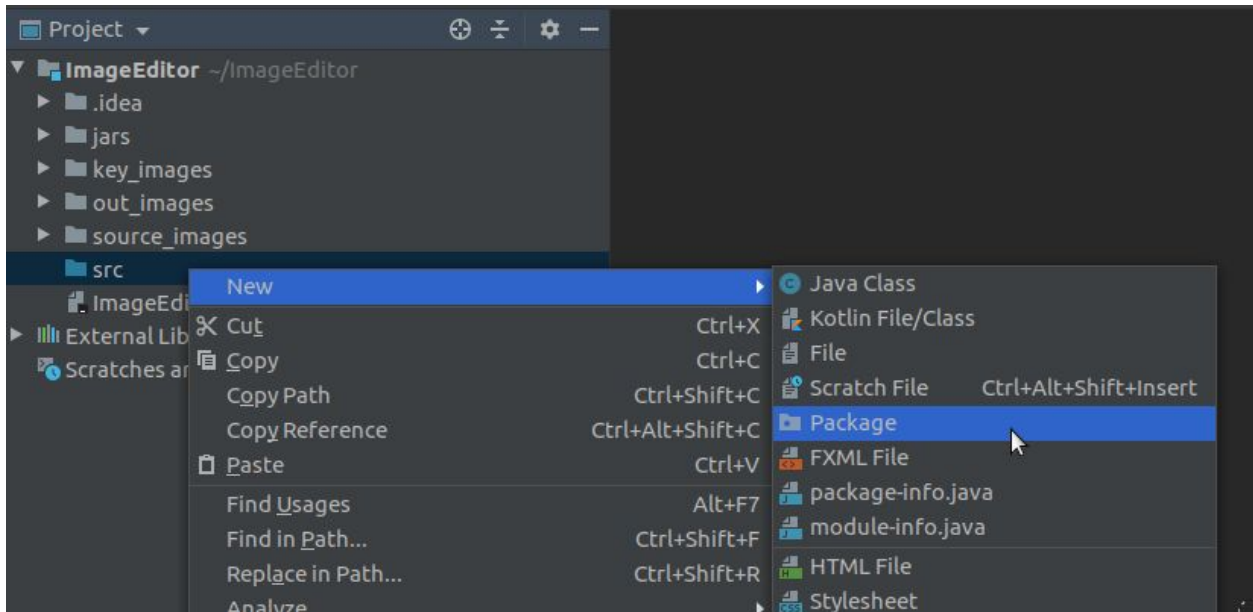
Now your project will build and will open:



Open the zip file you downloaded earlier and move jars, key_images, out_images, and source_images into the main folder of your project.



Right click on your src folder and add two new packages. One needs to be called passoff, the other needs to be called editor.



Move the files from test_files into passoff

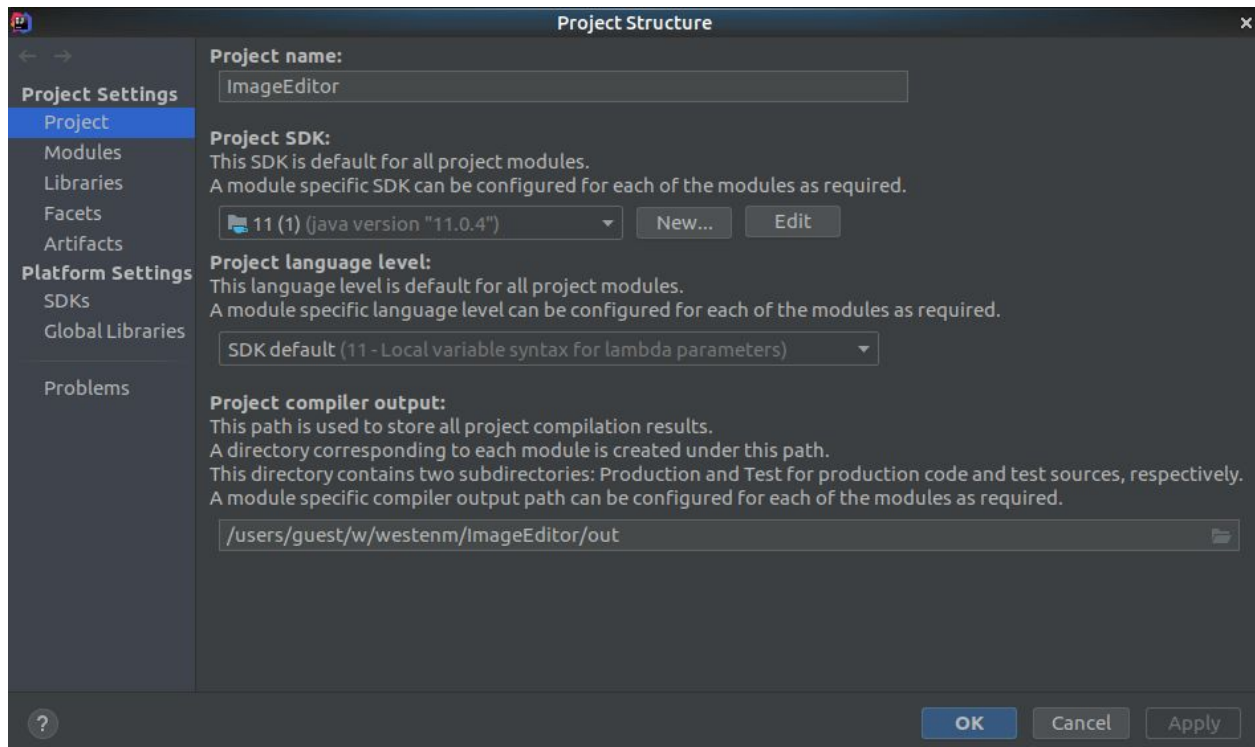


You'll see a lot of red errors, but that is okay for now.

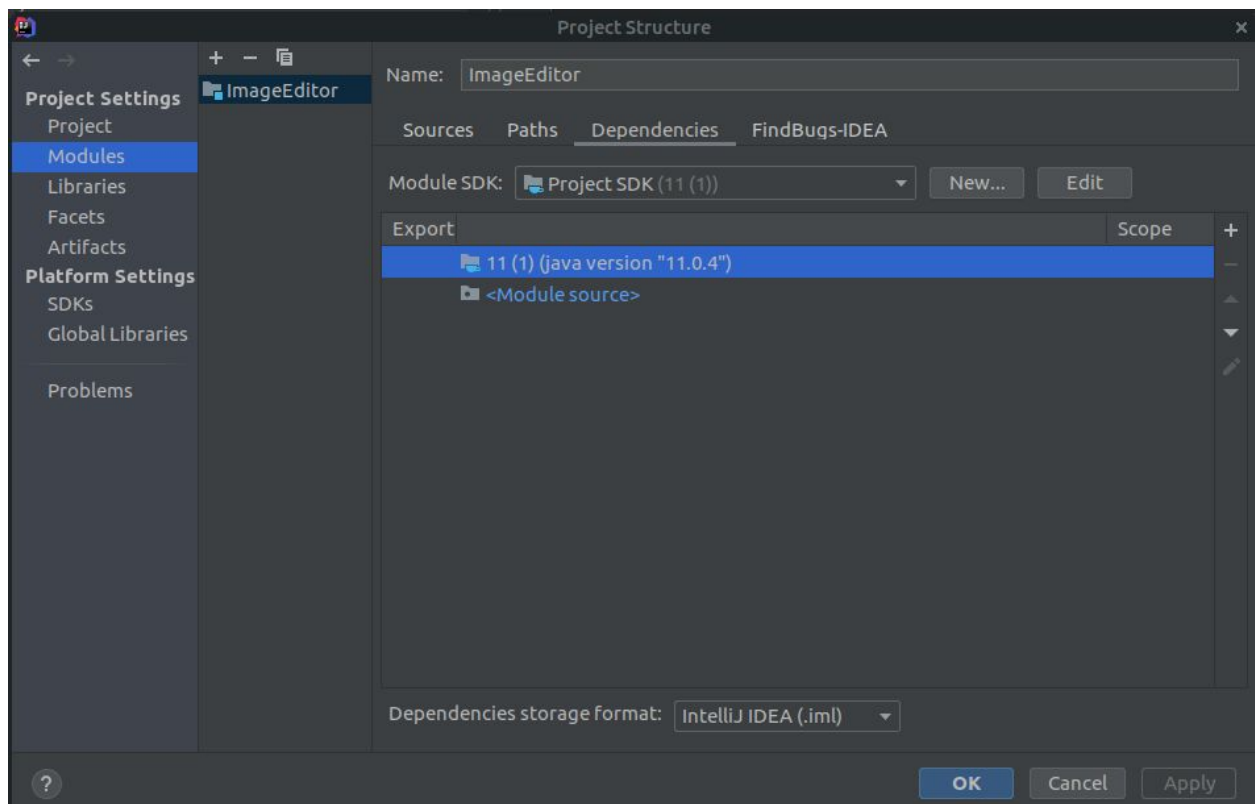
Dependencies

The last step we need to do is add dependencies to your project. Since these tests run on JUnit 5 you need to have the files for JUnit 5 so that IntelliJ can access them and use them. The folder that is labeled "jars" is where all this code is stored as .jar files. So we will add dependencies to your project for these files so that your project knows where to go in order to access the code.

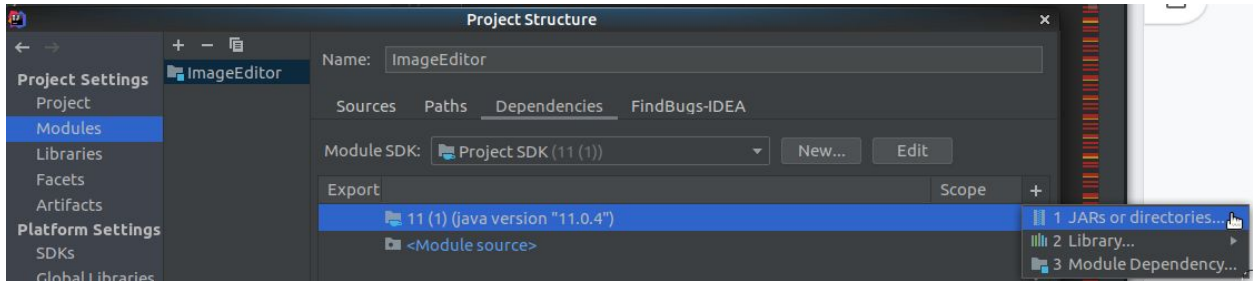
Select File > Project Structure (or use Ctrl+Alt+Shift+S) and you'll see this screen



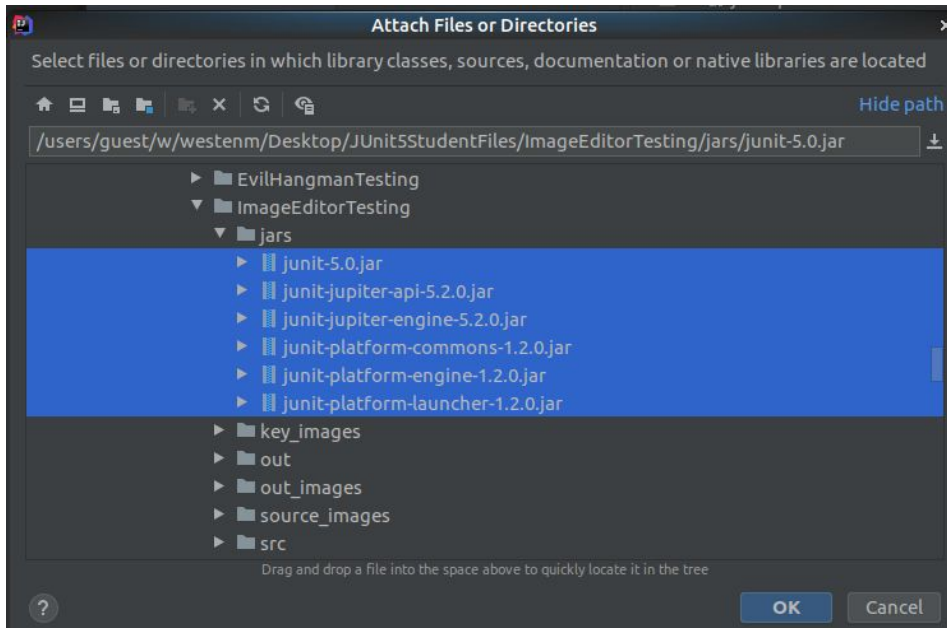
Double-click Modules and click the tab labeled Dependencies



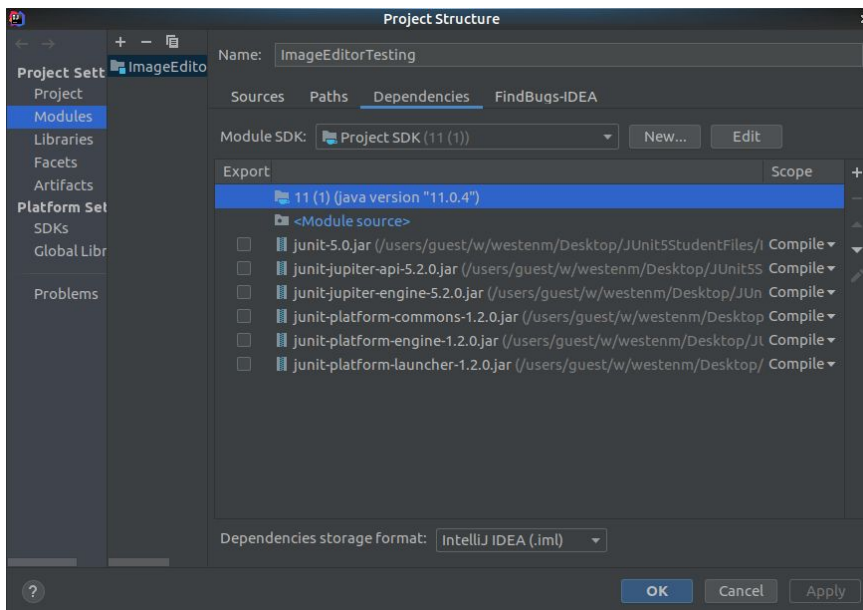
Now click the little “+” icon to the far right of the window and select “JARs or directories”



Navigate to the jars folder in your project and select all of the jar files then click “OK”

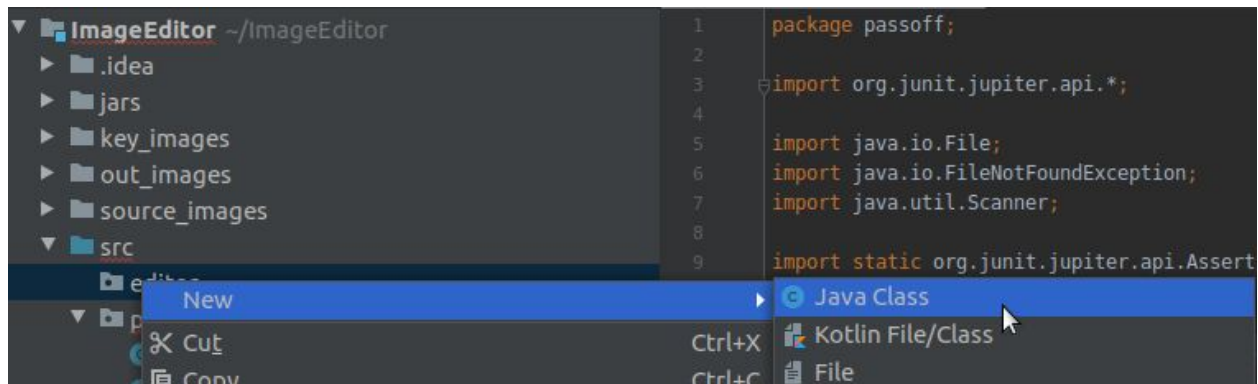


These will be added to the dependencies list. Click “OK”

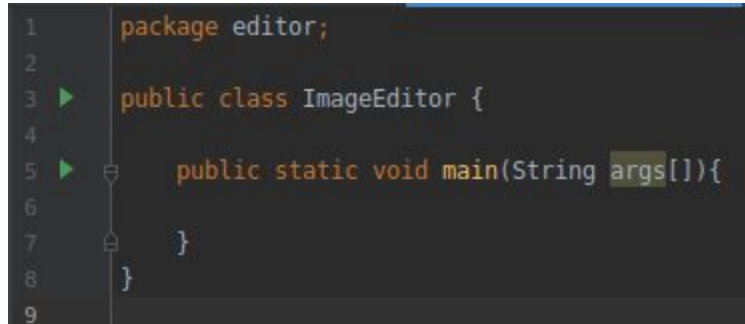


All of the red errors should disappear except for the ones related to `ImageEditor.main(command);`

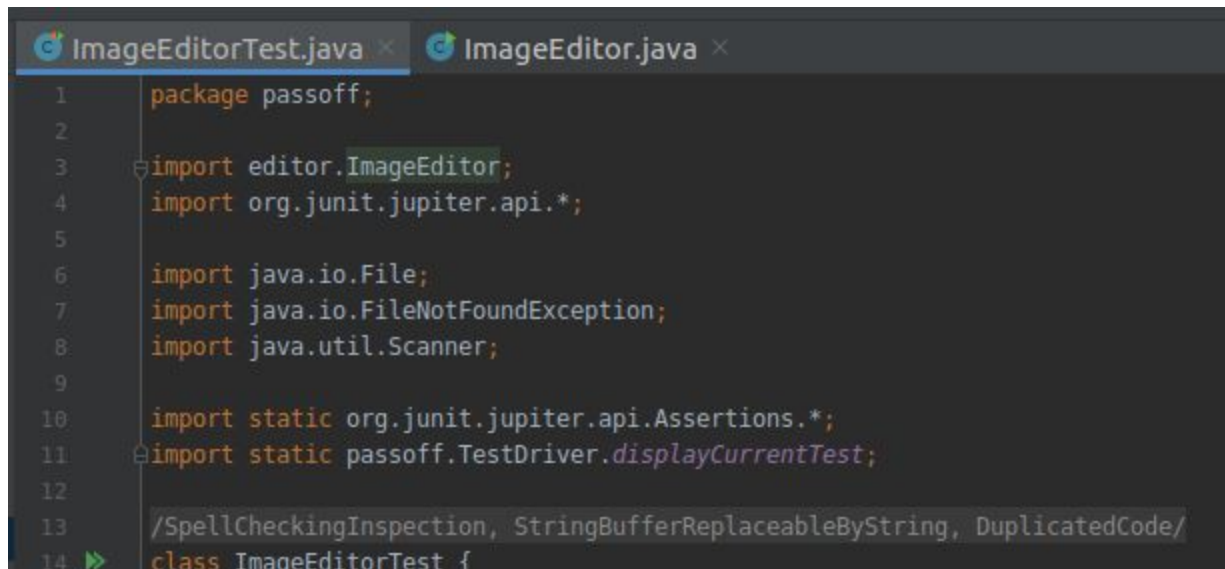
This is because you have not written your code yet. In your other package create a new Java class and call it `ImageEditor`



Give it a standard java main function which will look like this.

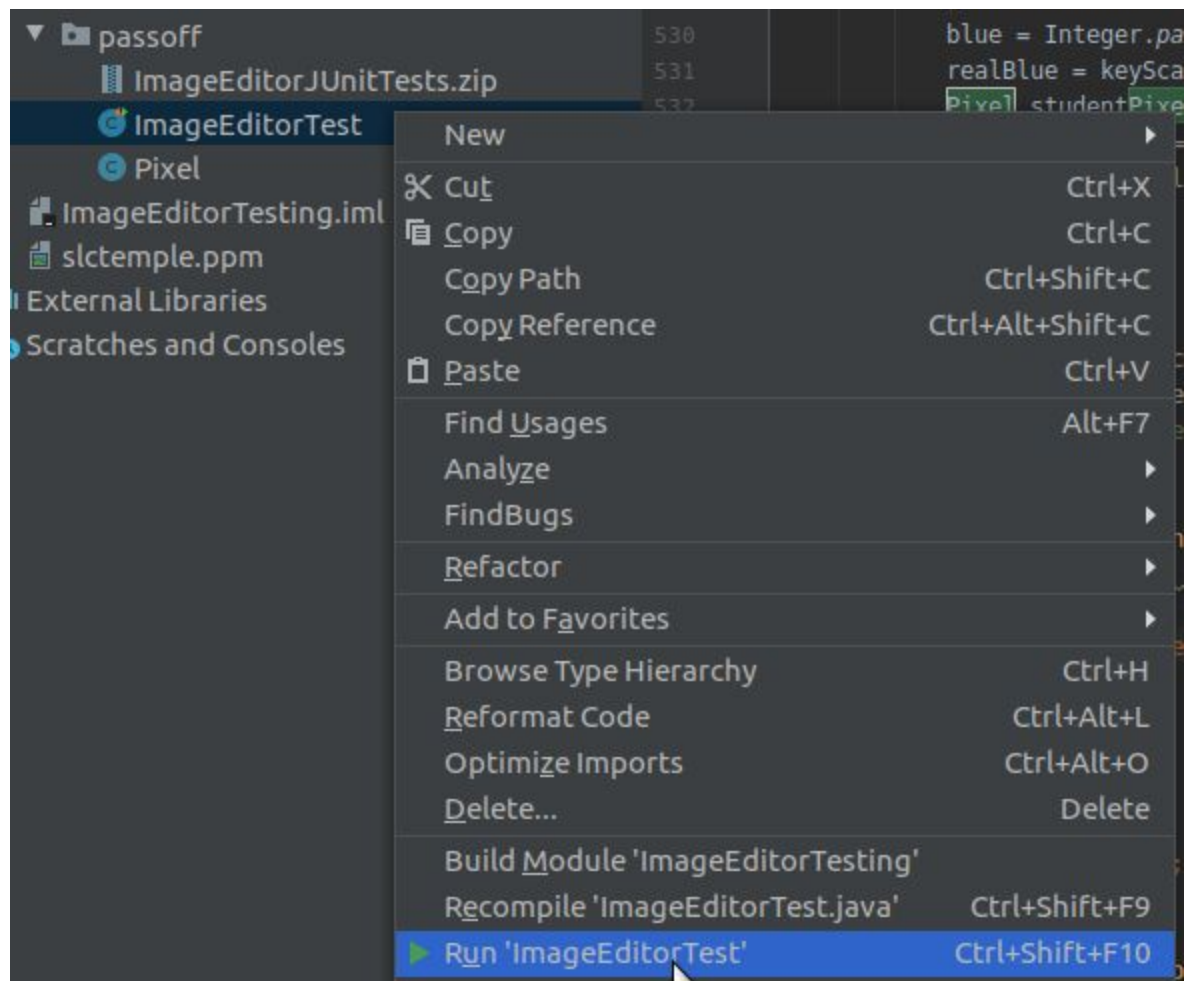


Now make sure that in your `ImageEditorTest` class it is importing from the correct location



All of the errors should be gone. Now right click on `ImageEditorTest` and select "Run"

ImageEditorTest” or click on the green arrow next to the class declaration in the file itself




```
1 package passoff;
2
3 import editor.ImageEditor;
4 import org.junit.jupiter.api.*;
5
6 import java.io.File;
7 import java.io.FileNotFoundException;
8 import java.util.Scanner;
9
10 import static org.junit.jupiter.api.Assertions.*;
11
12 class ImageEditorTest {
13     private static final String SOURCE_IMAGE_DIRECTORY = "source_
14     private static final String OUT_IMAGE_DIRECTORY = "out_images
15     private static final String KEY_IMAGE_DIRECTORY = "key_images/
16     private static final String FEEP = "feep.ppm";
17     private static final String ONE_DOES_NOT = "one-does-not-simp
18     private static final String PENGUINS = "Penguins.ppm";
19     private static final String SUNSET = "sunset.ppm";
20     private static final String TINY = "tiny.ppm";
21     private static final String INVERT = "invert";
22     private static final String GRAYSCALE = "grayscale";
23     private static final String EMOSS = "emboss";
24     private static final String MOTION = "motionblur";
25
26     @Test
27     @DisplayName("Test Invert Tiny")
28     void invertTinyTest(){
29         System.out.println("Invert Tiny");
30         String[] command = {SOURCE_IMAGE_DIRECTORY + TINY, OUT_I
31         try{
32             ImageEditor.main(command);
33         }
34         catch(Throwable t){
35             fail("Running main threw " + t.getClass());
36         }
37     }
38 }
```

This will run the tests. The tests will obviously fail because you have not written your code yet, but now you have the tests all ready to go so you can test as you code.