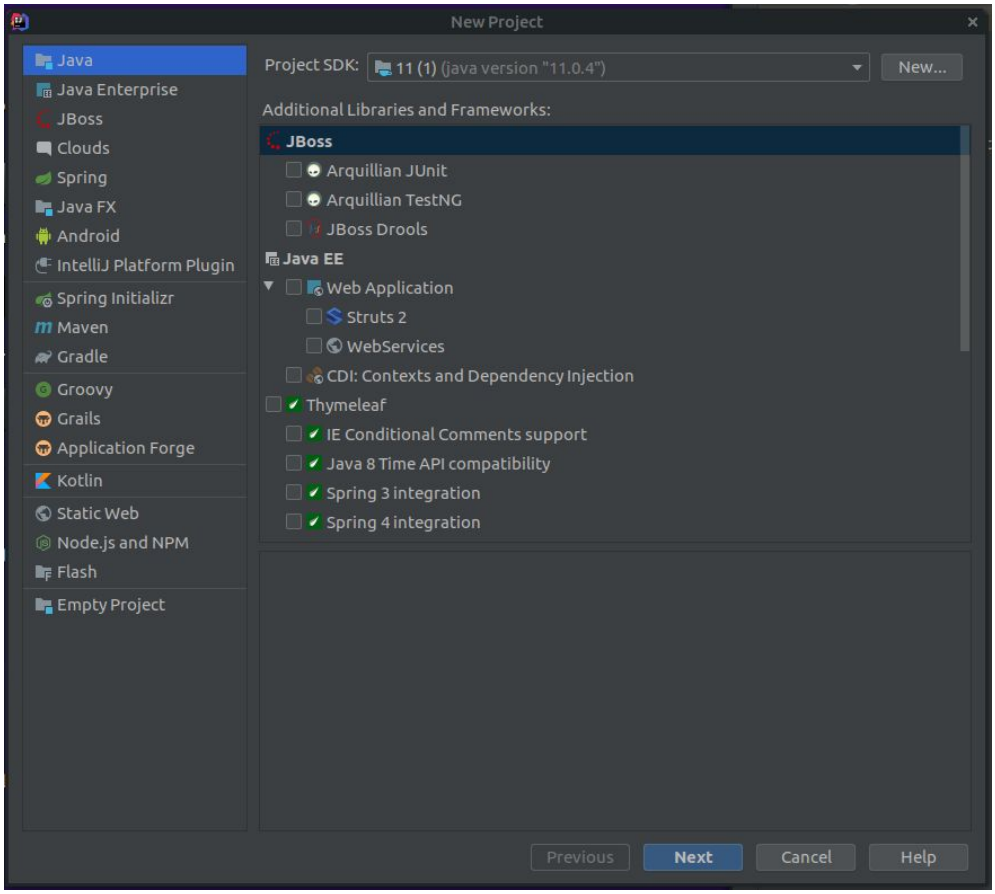
Setting Up EvilHangman Unit Tests

Downloads

First you will need to visit the Projects section of the current CS 240 website and download the “Files (Zip)” folder as well as the “Test Files (Zip)” folder. Inside this “Test Files (Zip)” you will find 4 dictionary files and two folders labeled “test_files” and “jars”.

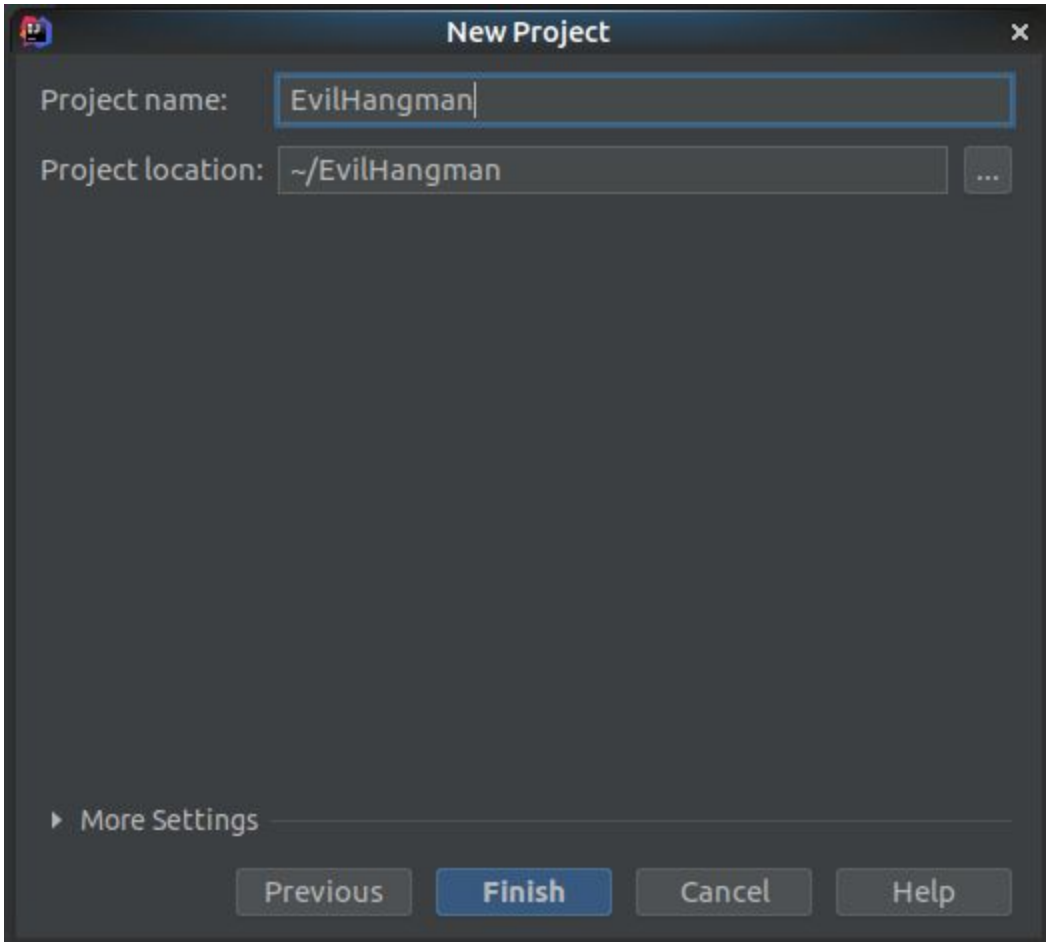
Create a Project

Go ahead and open IntelliJ and create a new Project.



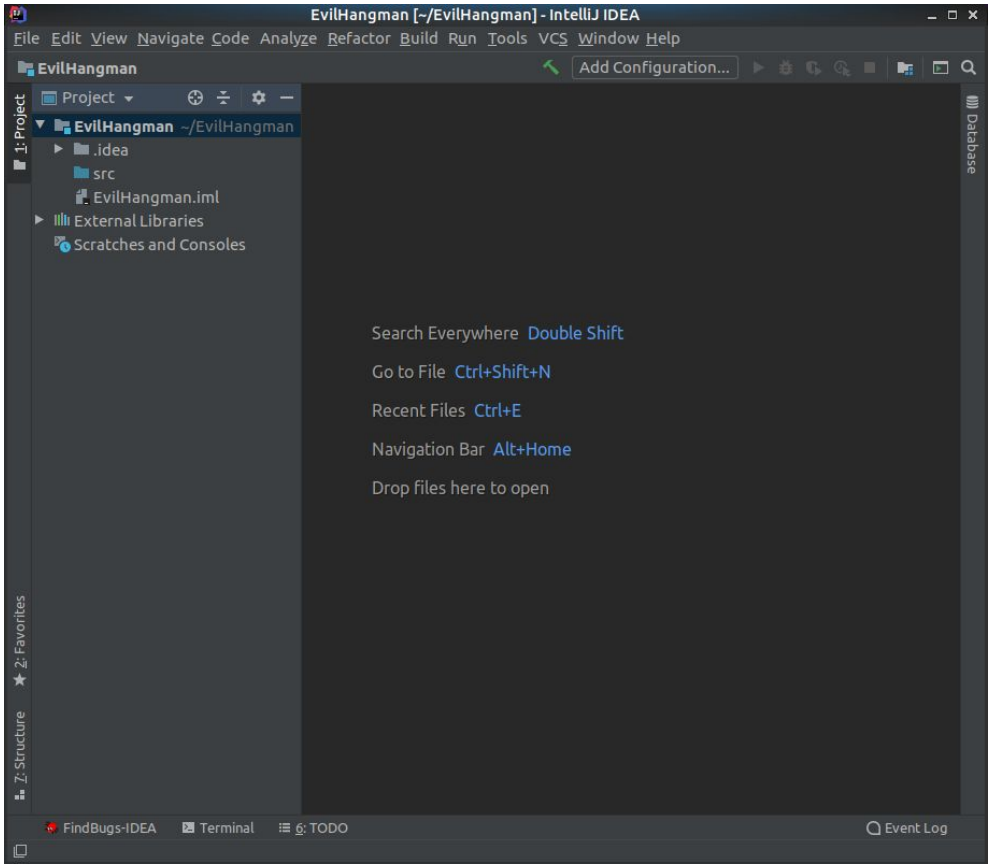
This tutorial will be setting up as a normal Java project. If you wish to create a Maven or Gradle project you will be responsible for setting it up on your own and getting dependencies working correctly.

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:

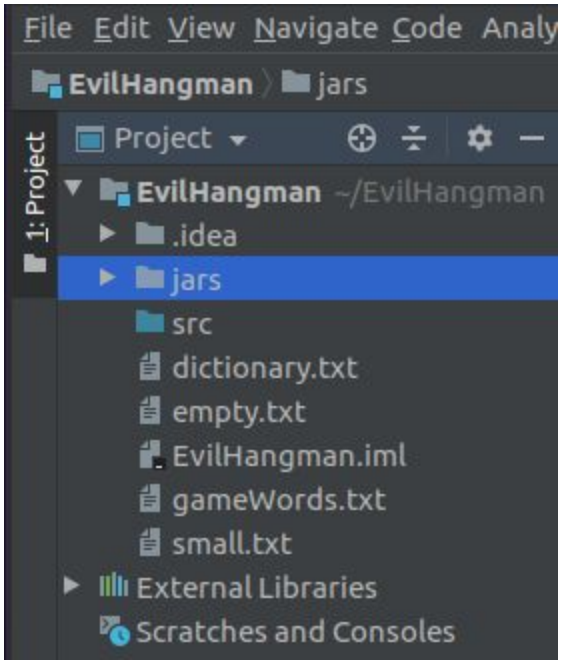


Name your project whatever you want though we suggest using EvilHangman. Click “Finish”

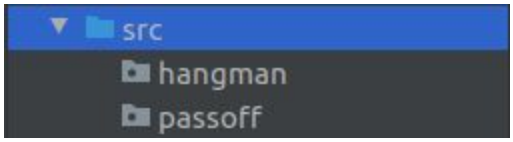
Now your project will build and will open:



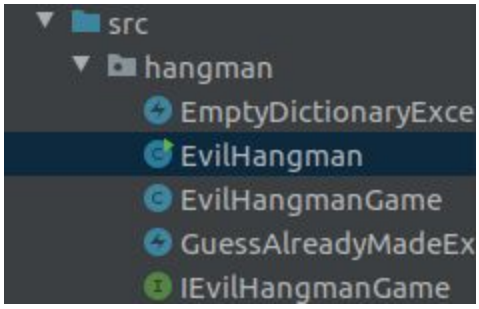
Open the “Test Files (zip)” file you downloaded earlier and move the dictionary files and “jars” folder into the main folder of your project.



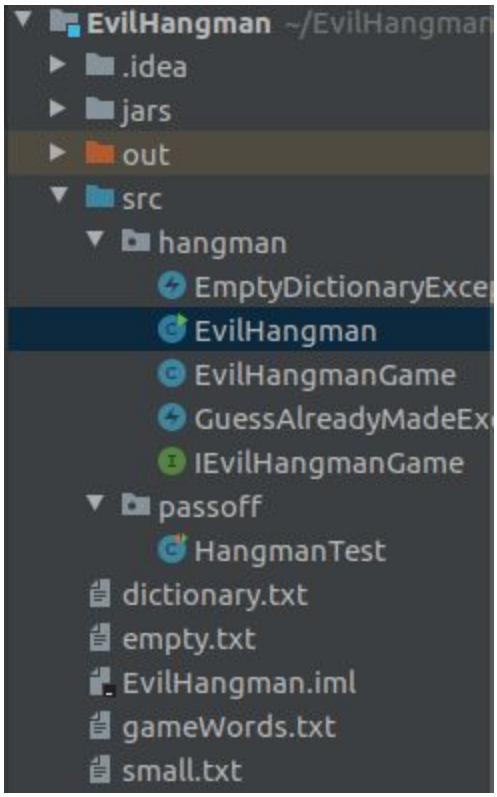
Right click on your src folder and add two new packages (New > Package). One needs to be called passoff, the other needs to be called hangman.



Take all the files you downloaded from “Files (Zip)” and move them into the hangman folder



Move the files from test_files into passoff

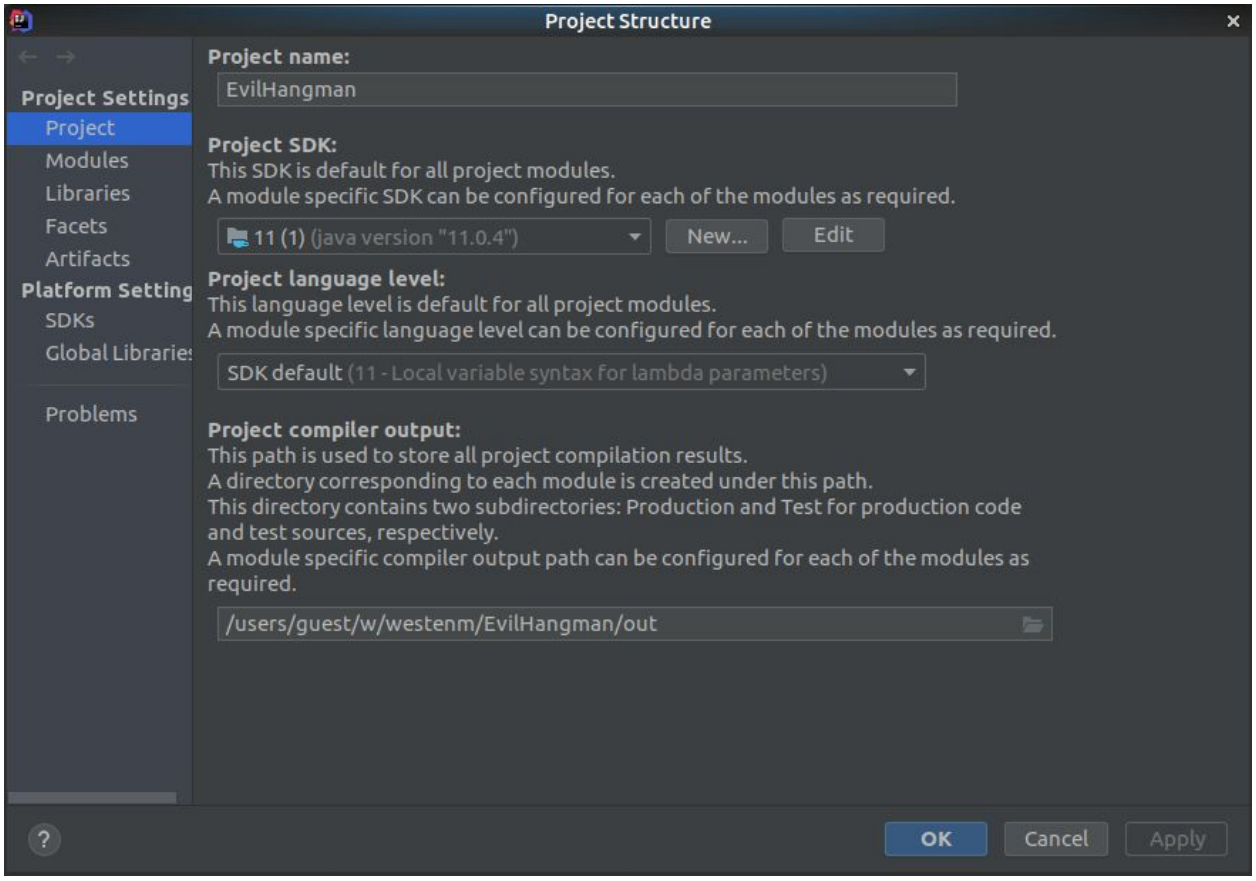


You’ll see a lot of red errors, but that is okay for now.

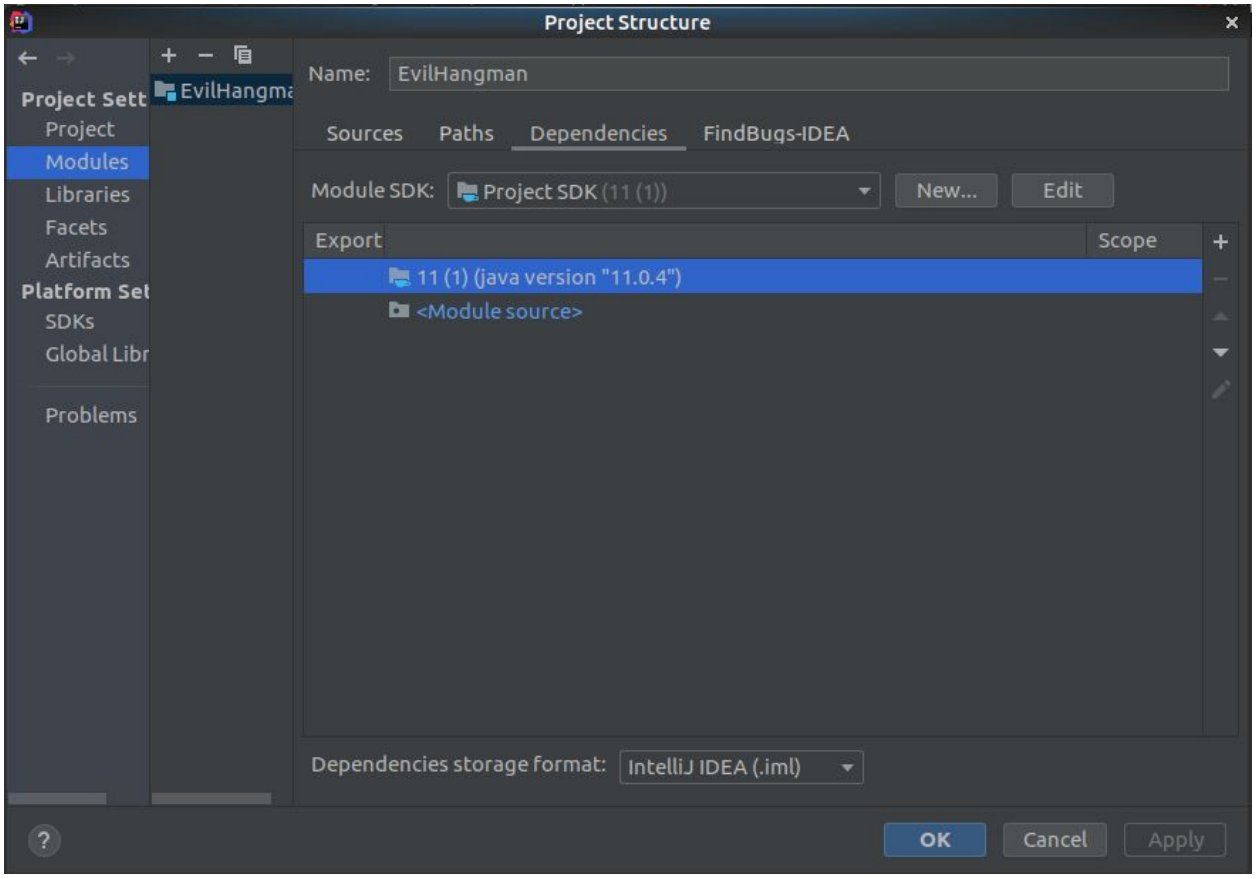
Dependencies

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.

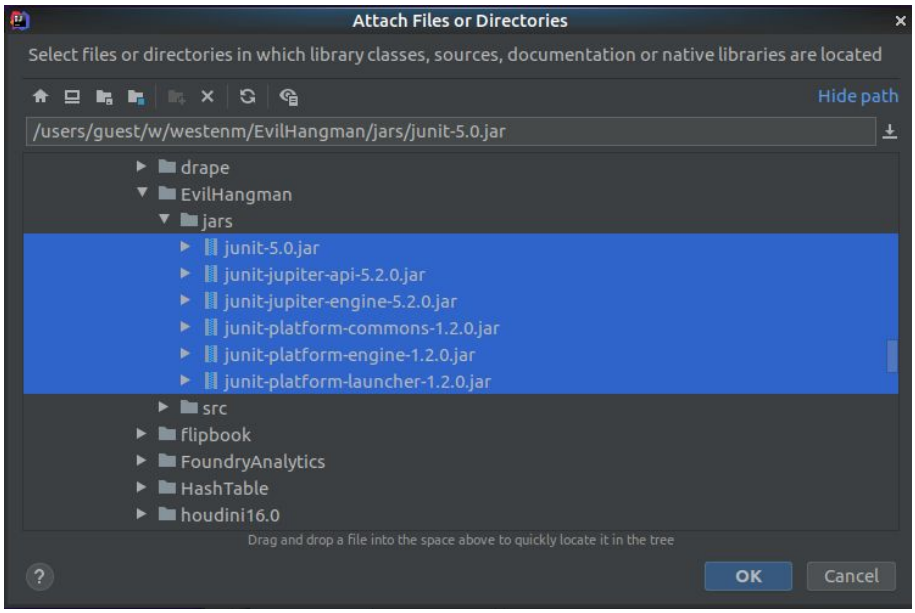
Select File > Project Structure (or use Ctrl+Alt+Shift+S) and you’ll see this screen



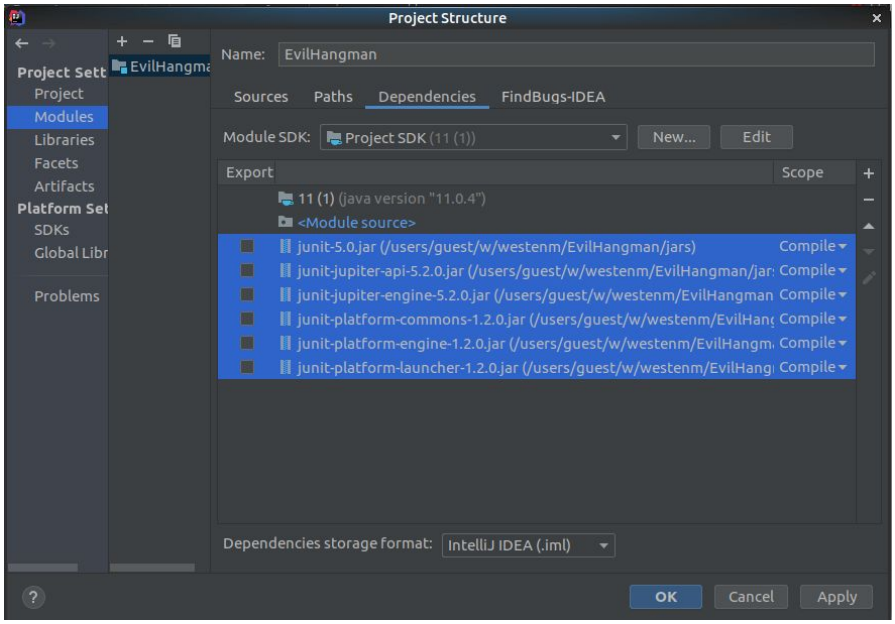
Double-click “Modules” and click the tab labeled “Dependencies”



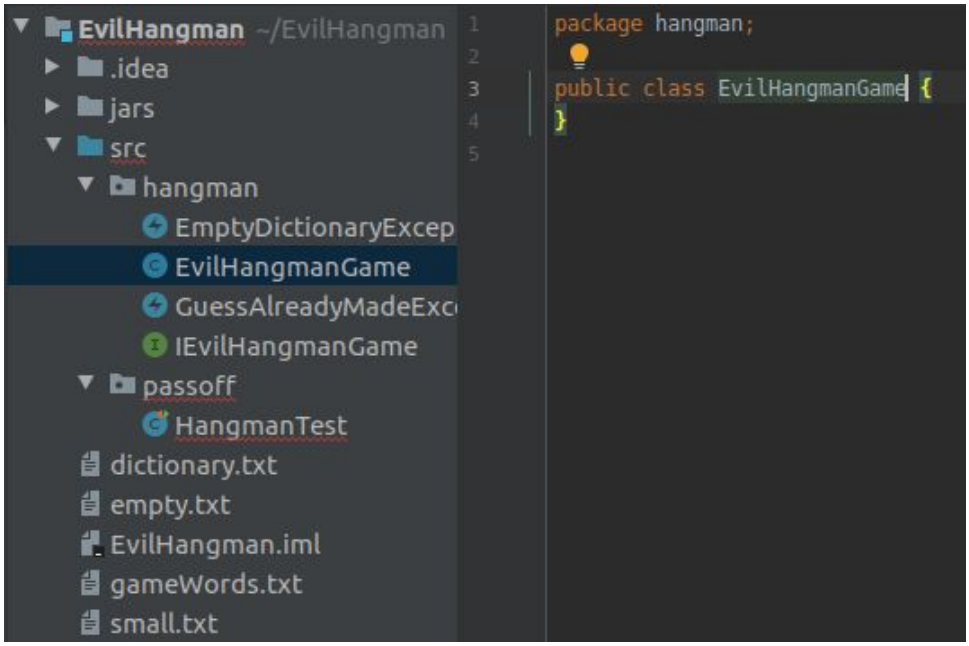
Now click the little “+” icon to the far right of the window and select “JARs or directories”
Navigate to the jars folder in your project and select all of the jar files then click “OK”



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

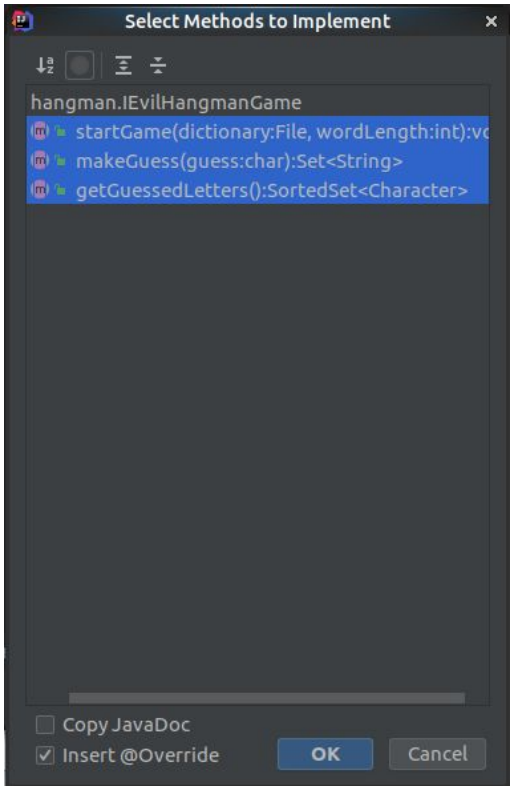


All of your errors relating to dependencies will disappear, but you should still have several errors. This is because we don't yet have our EvilHangmanGame class. Right click on your hangman package and create a new Java class and call it EvilHangmanGame.

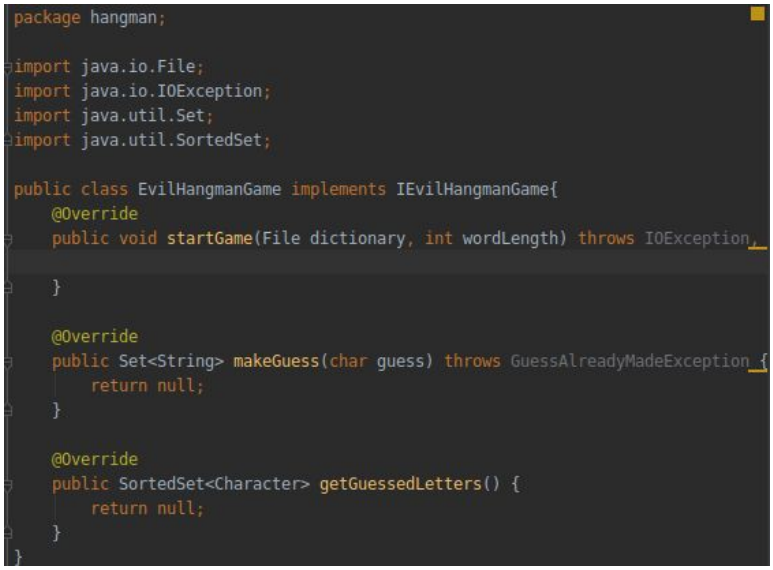


Have this class implement the "IEvilHangmanGame" interface.

Click the red light bulb or click Alt+Enter to pull up suggestions and select Implement Methods. Select all the methods and click OK



IntelliJ will automatically generate the needed code



```
package hangman;

import java.io.File;
import java.io.IOException;
import java.util.Set;
import java.util.SortedSet;

public class EvilHangmanGame implements IEvilHangmanGame{
    @Override
    public void startGame(File dictionary, int wordLength) throws IOException,

    }

    @Override
    public Set<String> makeGuess(char guess) throws GuessAlreadyMadeException {
        return null;
    }

    @Override
    public SortedSet<Character> getGuessedLetters() {
        return null;
    }
}
```

Now all of your error should be gone.

Now right click on HangmanTest and select “Run Test” or click on the green arrow next to the class declaration in the file itself.

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.

IMPORTANT

The HangmanTest will only test your implementation of IEvilHangmanGame. This will not test your game loop (which you will write in EvilHangman) and the non-logical elements of playing your game (such as displaying the guessed letters) This is something that you will have to test by yourself and that the TAs will be testing manually when you come in to passoff.