Installing and Using Visual Studio Code for Java

1. Installing Visual Studio Code

Before downloading and installing Visual Studio Code, ensure that you have downloaded and installed the Java SE Development Kit (referred to as the JDK or SDK) from the Oracle site.

The latest version can be downloaded from here:

https://www.oracle.com/uk/java/technologies/downloads/

Once you have downloaded and installed the JDK you can then download and install Visual Studio Code from the following site:

https://code.visualstudio.com/download

Once you have installed both the JDK and Visual Studio Code you will be in a position to run all the programs in this book, except those that involve JavaFX. Running JavaFX programs requires some additional steps - this is dealt with in section 3.

2. Using Visual Studio Code

2.1 Starting a new project

When you first open Visual Studio Code you will see the home screen as shown in Figure 1.

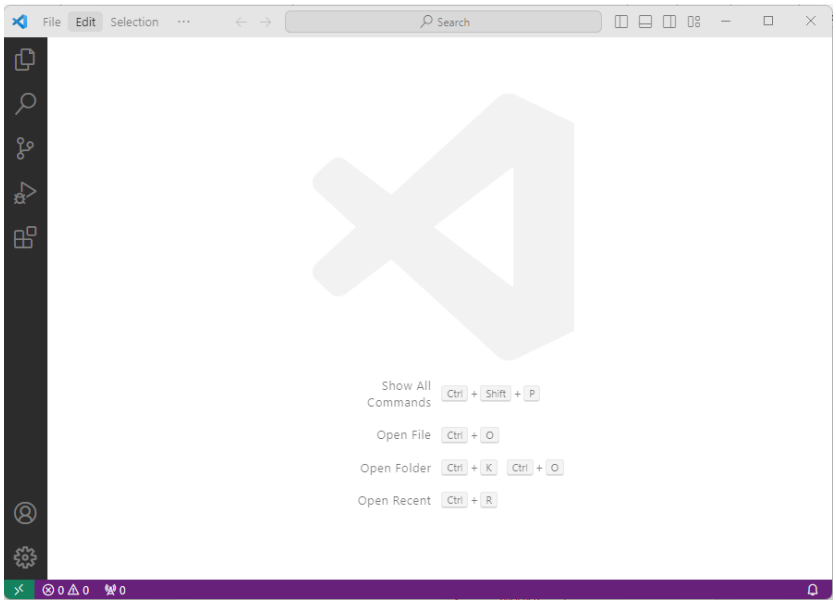
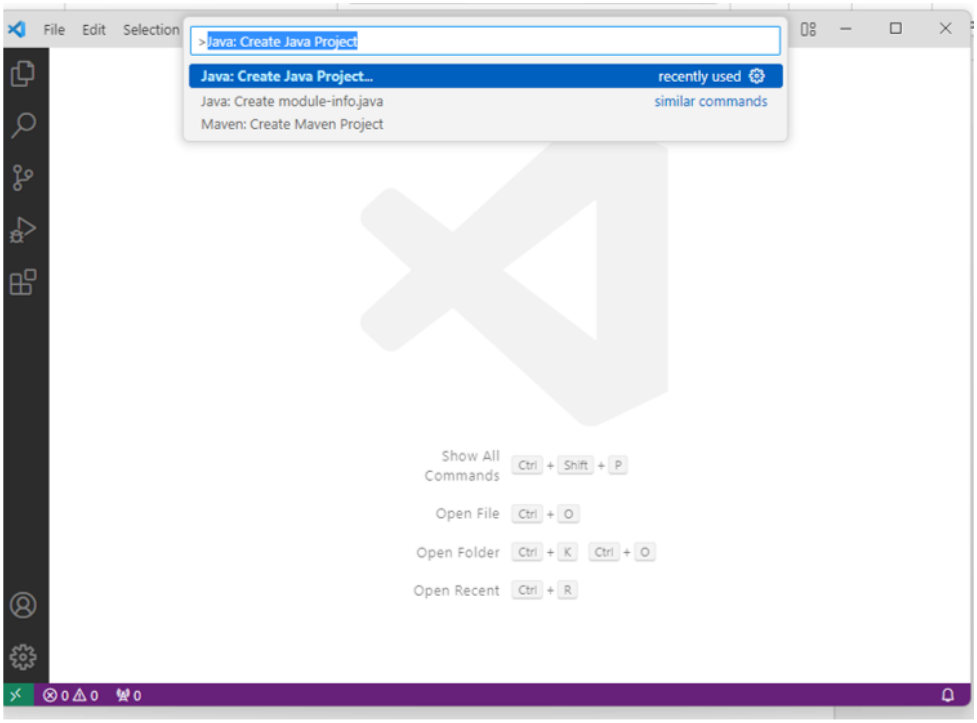


Figure 1

Each project in Visual Studio Code consists of a number of classes and sometimes additional files - for example image files. While you are getting started, it might be a good idea to start a new project for each chapter of the

book. When you move on to larger applications, such as the case study in chapters 11 and 12, or applications that you are creating yourself (such as a class assignment), you should create a single project for your application.

To create a project, press *Ctrl+ Shift + P.* Then, in the space provided, type **Java: Create Java Project**, (followed by *Enter*), as shown in Figure 2.



You will now see the screen shown in Figure 3. Choose the **No build tools** option.

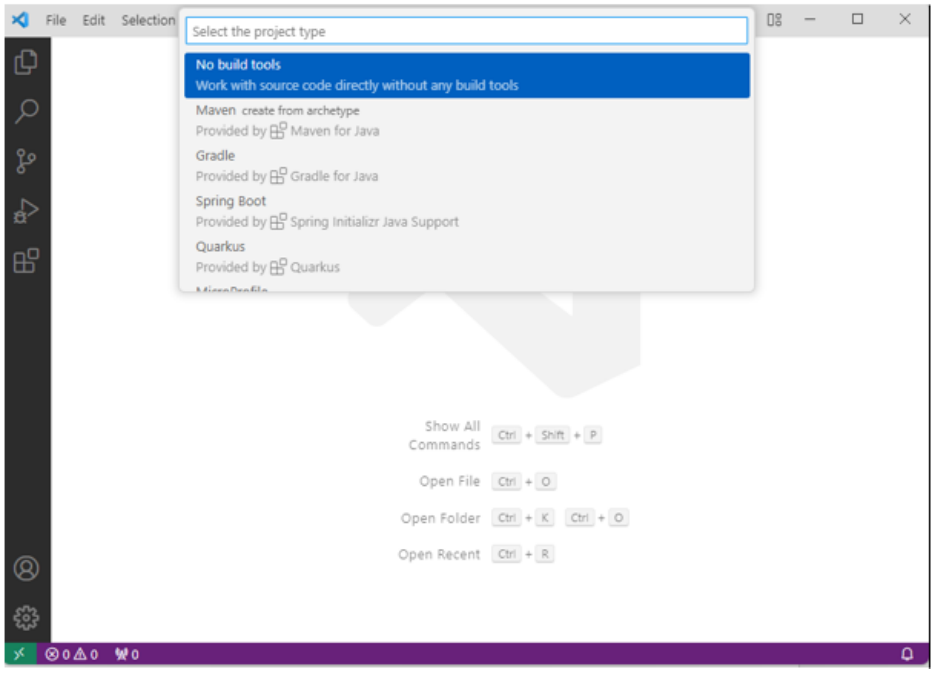


Figure 3

You will then be prompted to choose the location of your project, and then to choose a name for the project. We have called ours FIRSTPROJECT. You will see that your project is now listed on the left-hand pane as shown in Figure 4.

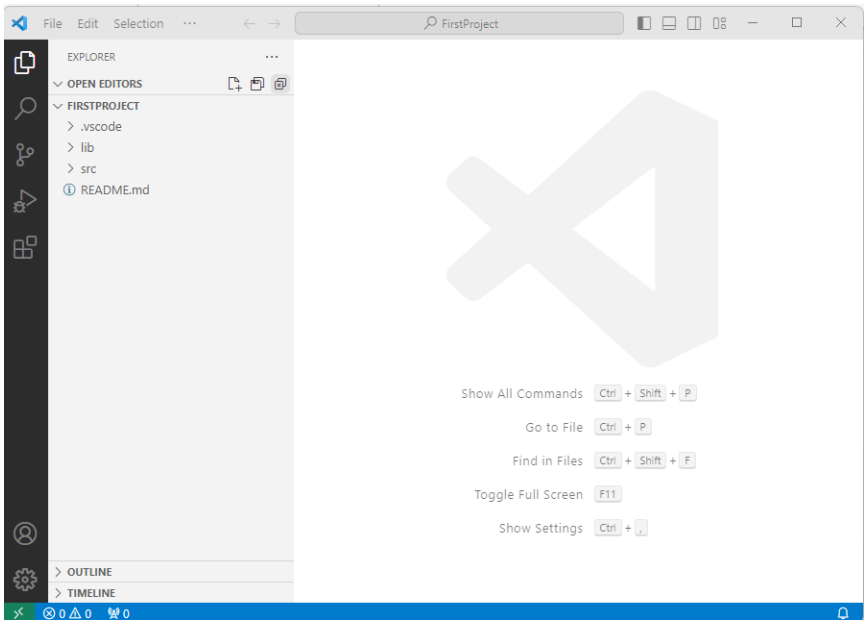


Figure 4

2.2 Adding new files to the project

Click on the **File** tab and choose **New File**, as shown in Figure 5.

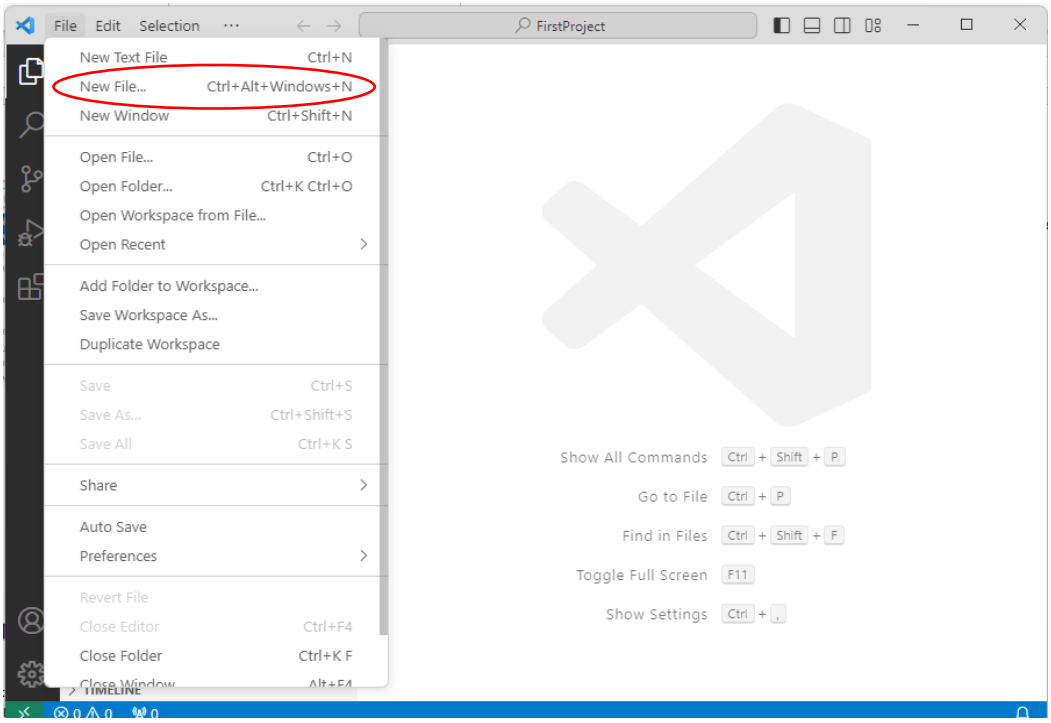


Figure 5

You will then be asked to select a file type. Choose **New Java File** from **Project Manager for Java** (Figure 6).

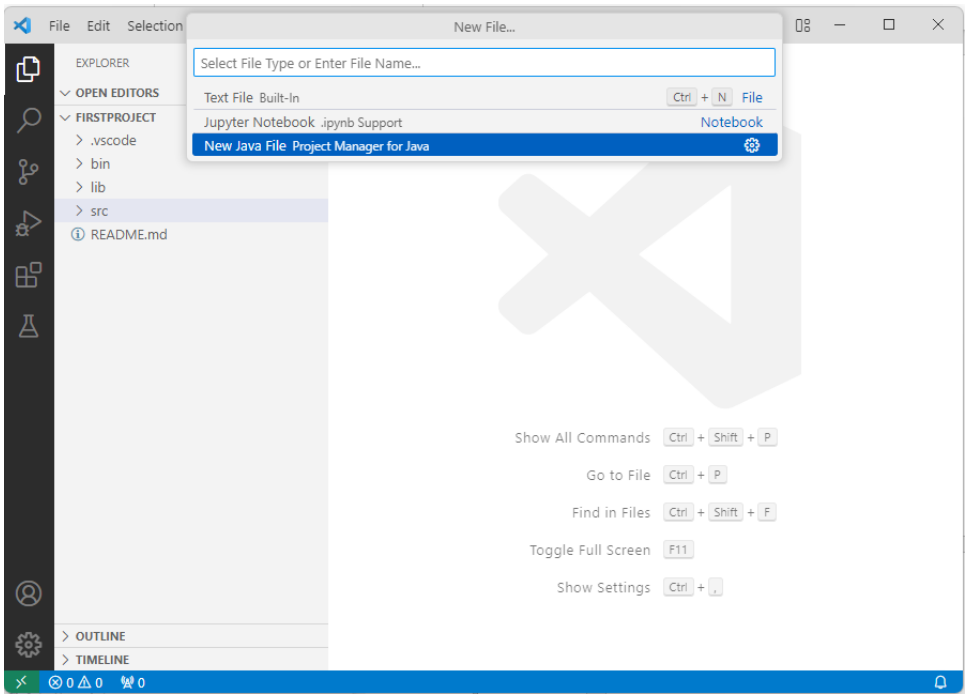


Figure 6

You will now be asked to select the Java type you want to create. Choose **Class** as shown in Figure 7.

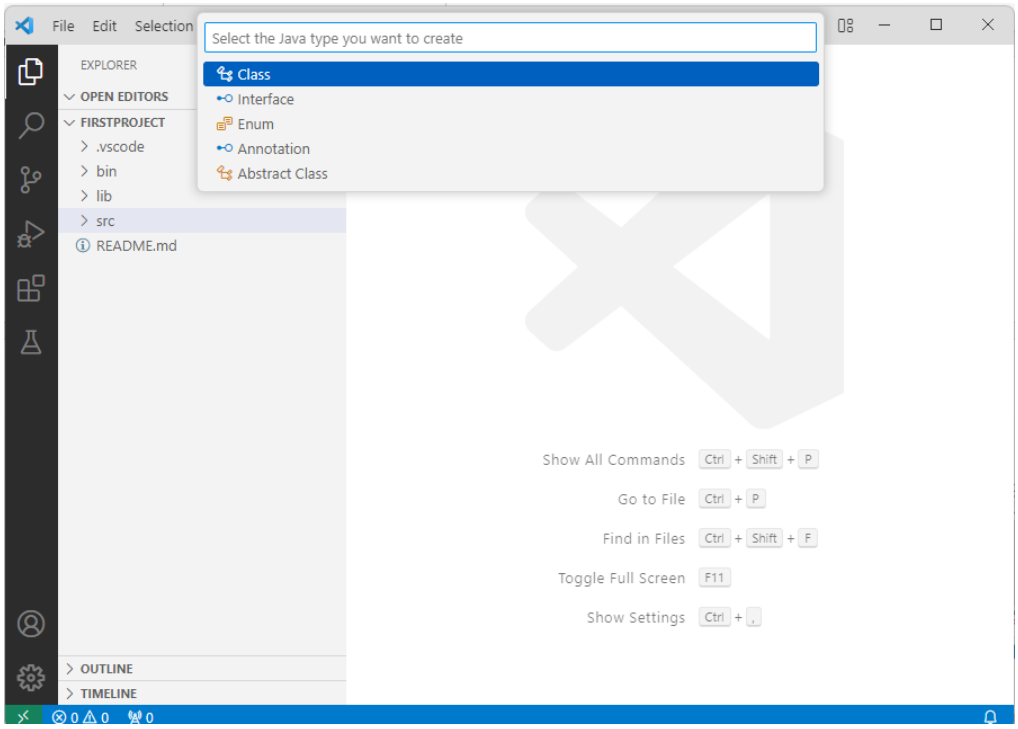


Figure 7

When we add a new file to the proje ct, it is important that the file is added in the correct folder. When writing your first programs, you will not want to specify a package, and the file should therefore be added to the *src* folder. The easiest way to achieve this is to right-click on the *src* (source) folder, then choose **New** followed by **Java Class** (see Figure 8).

At this point you will be asked to supply a name for your class. As you can see from Figure 8, we have called our class `HelloWorld`.

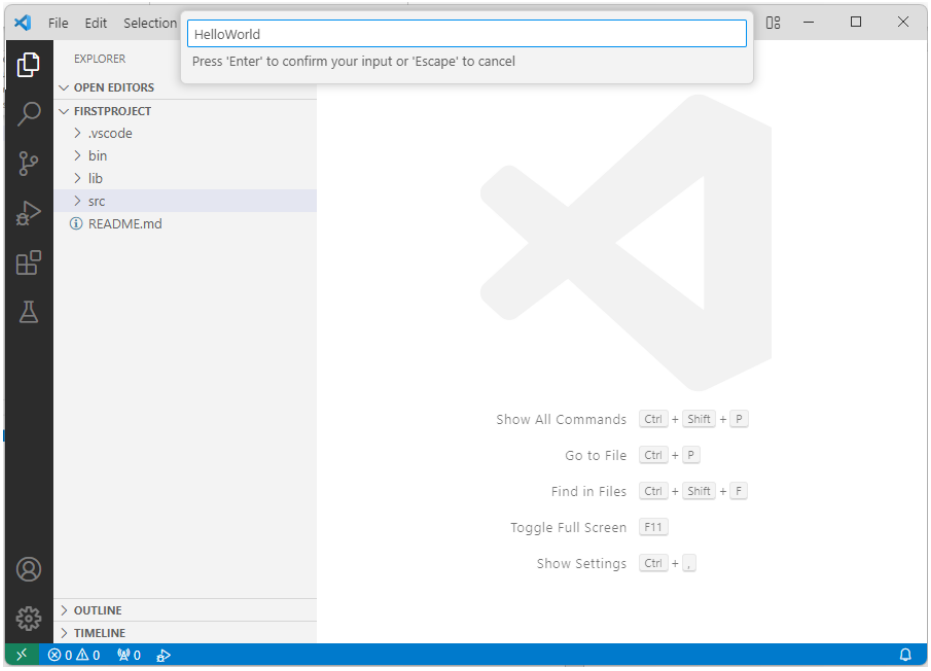


Figure 8

The screen shown in Figure 9 now appears. As you can see, the file that you have created is saved to the **src** folder. This folder, which is short for **source**, is where Visual Studio Code keeps the java source code files.

As you can see there is also a file called App.java - this is supplied to you automatically, and you can adapt it if you want, rather than creating a new file. But it can also be ignored or deleted.

You can also see that the code itself for our new file appears in a window on the right - an outline is automatically created for you.

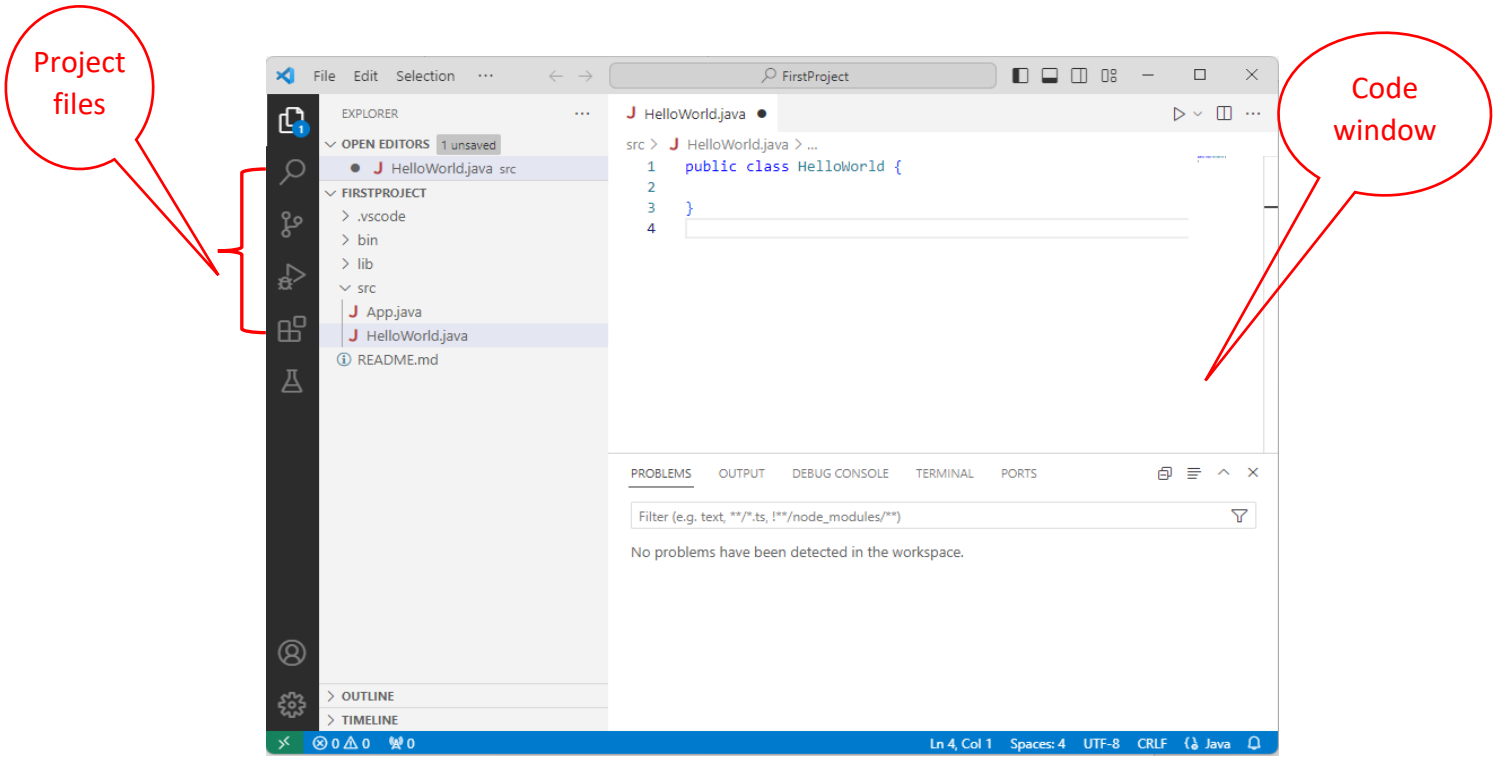


Figure 9

You can now go ahead and complete the code, as we have done in Figure 10.

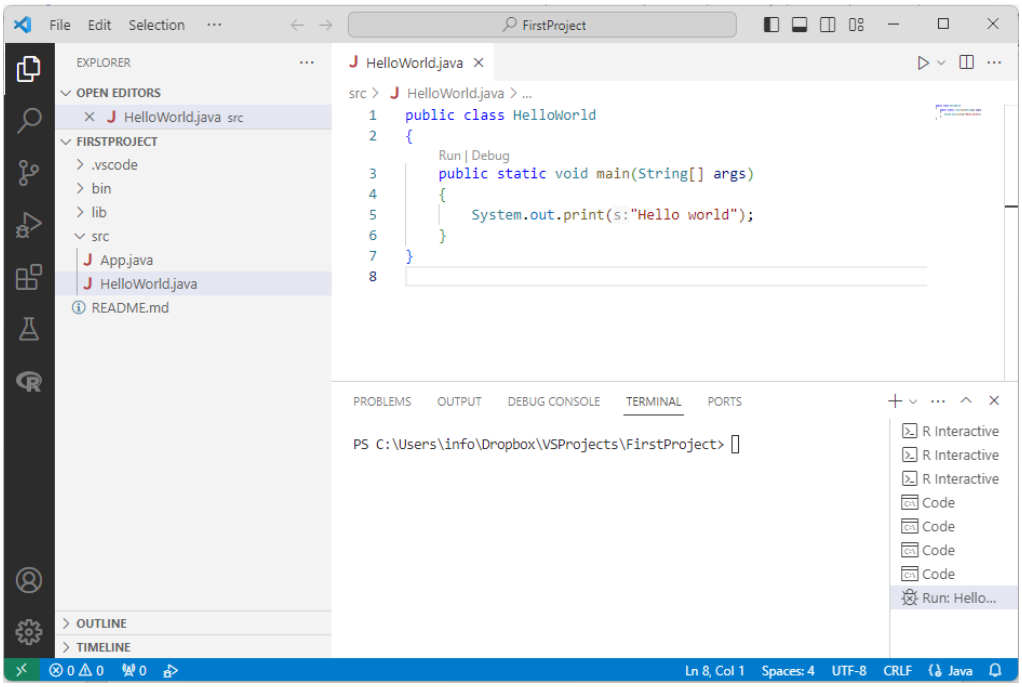


Figure 10

2.3 Compiling and running applications

Classes are automatically compiled as soon as they are written. Error messages are listed under the **Problems** tab in the bottom window, as shown in Figure 11, where a semi-colon has been omitted.

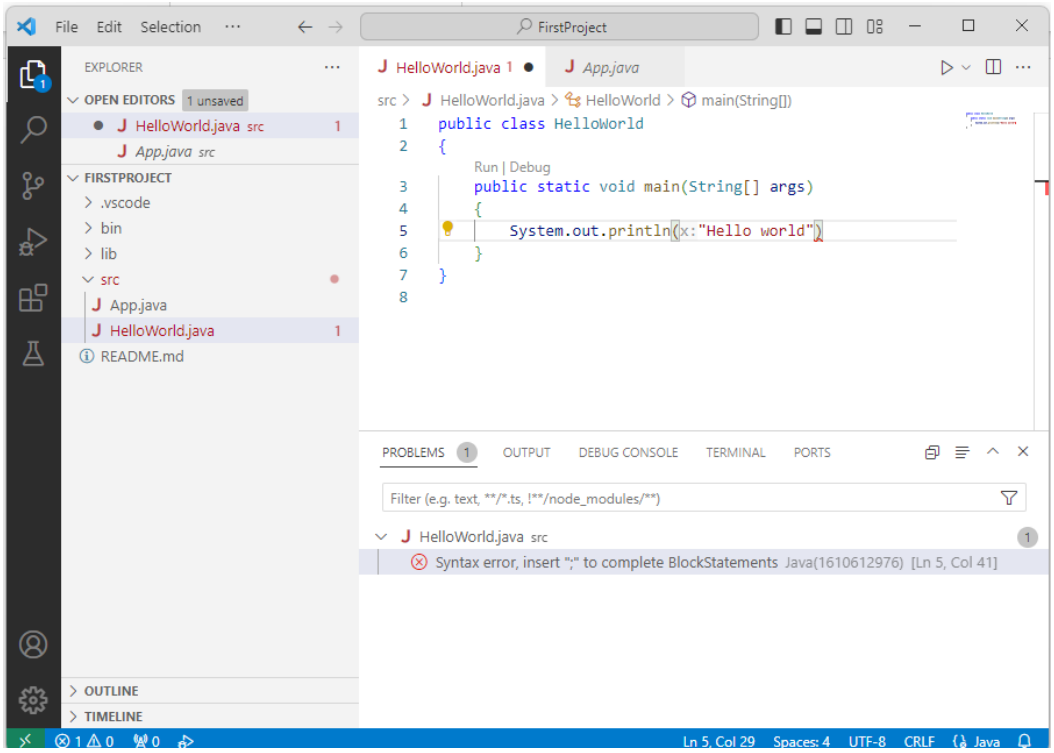


Figure 11

There are two ways in which you can run your program. You can click on the arrow at the top right hand corner - this will run the program that is open in the current window. Or you can right click on the file name in the left hand pane and then click **Run Java**. This is illustrated in Figure 12.

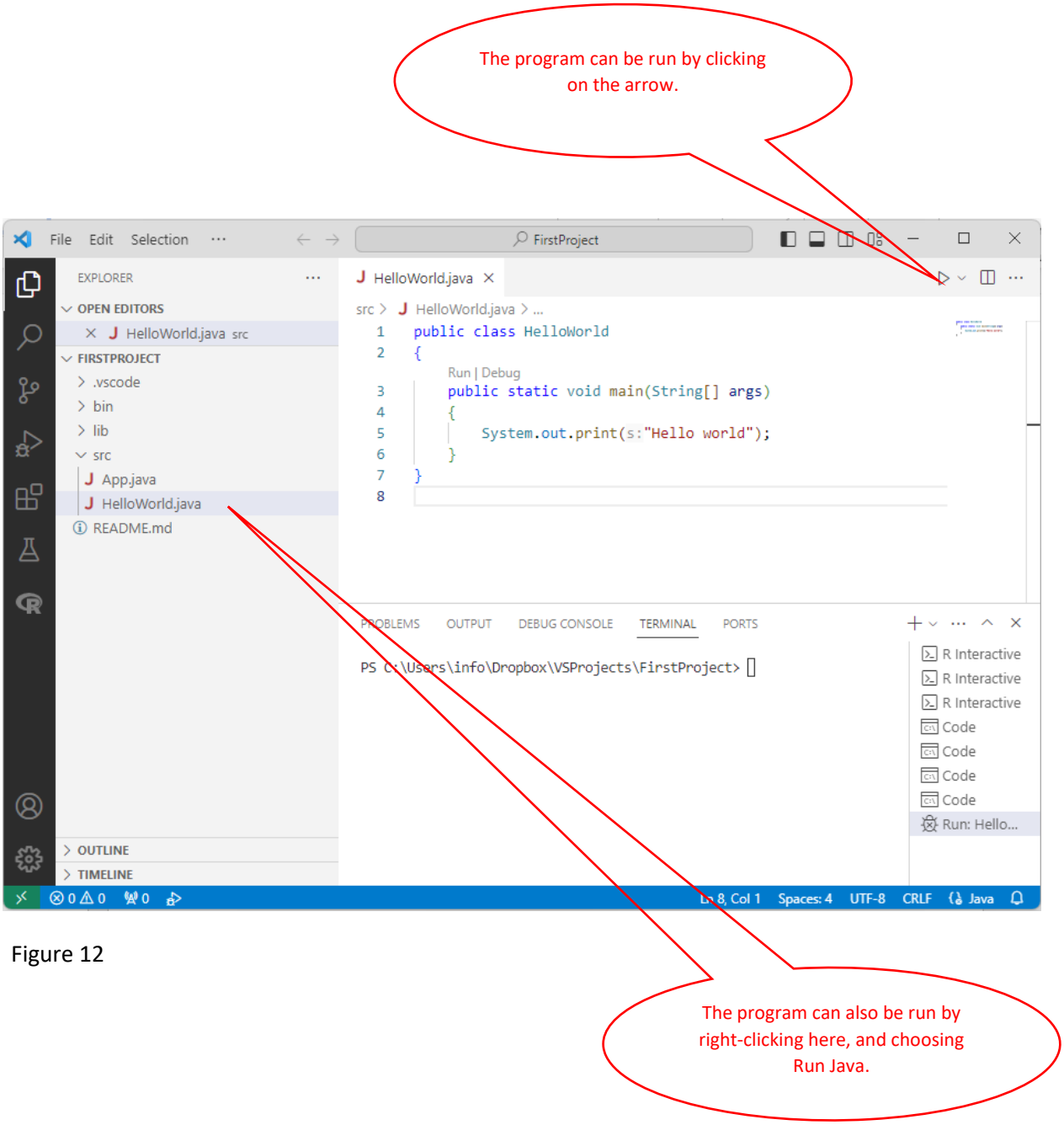


Figure 12

The program output appears in the bottom pane under the TERMINAL tab as shown in Figure 13. As you can see, the output is somewhat cluttered, but with longer, interactive programs, this is less of an issue.

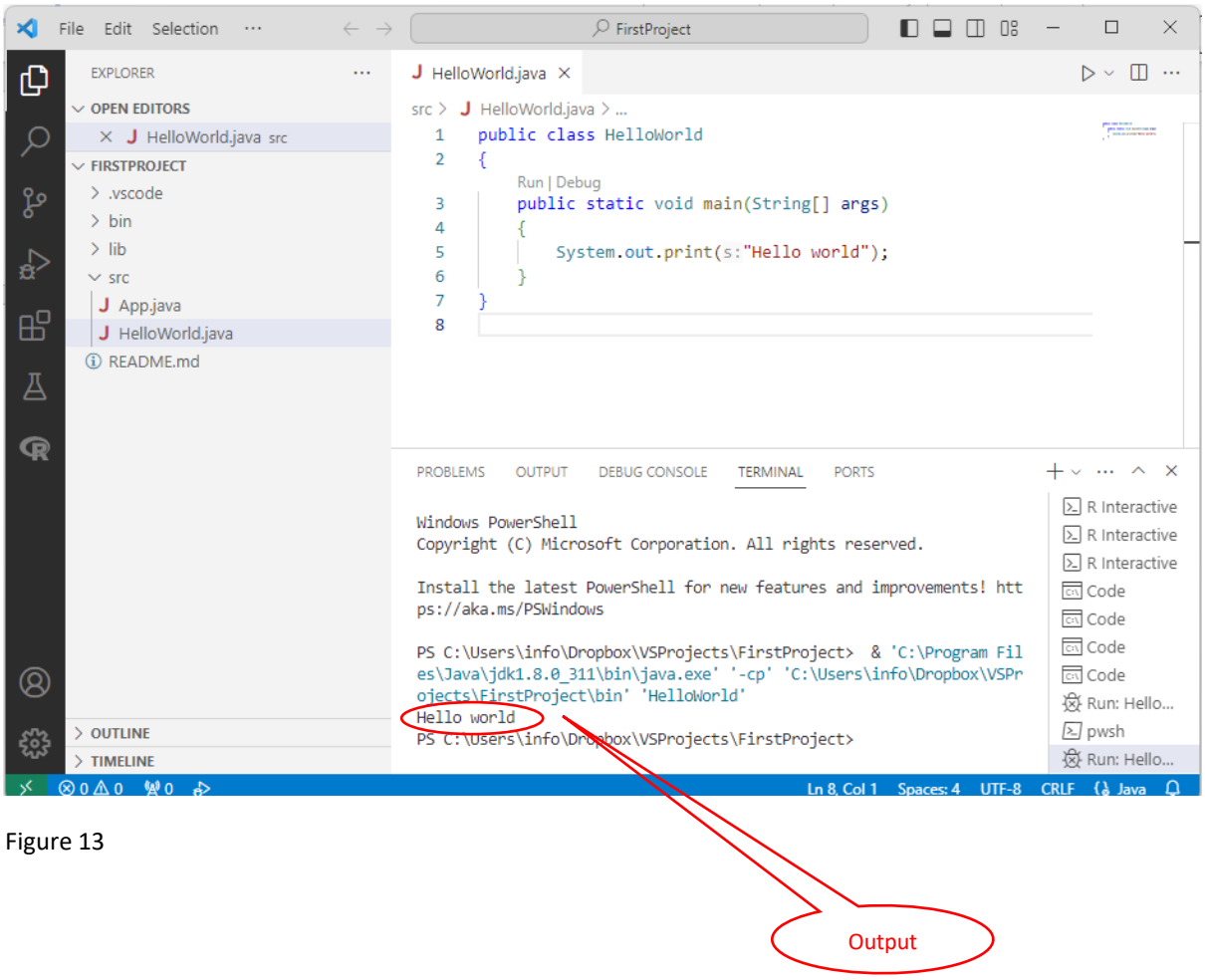


Figure 13

2.4 Adding existing classes to projects

Adding existing classes (that is to say, the source code) to a project is an easy matter - simply drag them or copy them to the **src** folder.

3 Creating JavaFX projects

Download the latest version of the JavaFX SDK (which comes in the form of a zipped file) from the link below.

https://gluonhq.com/products/javafx/

Extract the files to a location of your choice. Here we will assume they are in a folder called:

C:\JavaFX

The above assumes the use of a Windows system. If you are using another operating system such as Linux or MacOS, note that such systems use the forward slash instead of the backslash, and use the mount point (/) instead of a drive letter.

Within the folder where you have unzipped your files, there will be another folder called *lib* which contains the required .jar files. There will also be a folder called *bin* and a folder called *legal*.

The above step only has to be done once. However the following steps have to be followed for each new project.

Start a new project. Press **Shift + Ctrl + P** and type Java: Open Project Settings, followed by the *Enter* key (Figure 14).

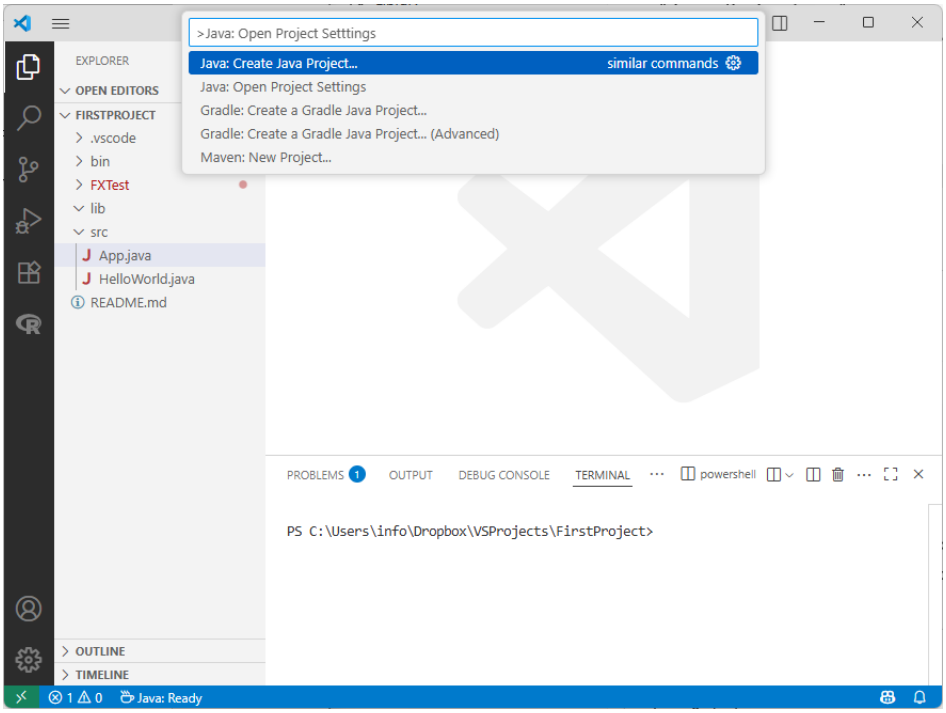


Figure 14

Select the **Libraries** tab (Figure 15).

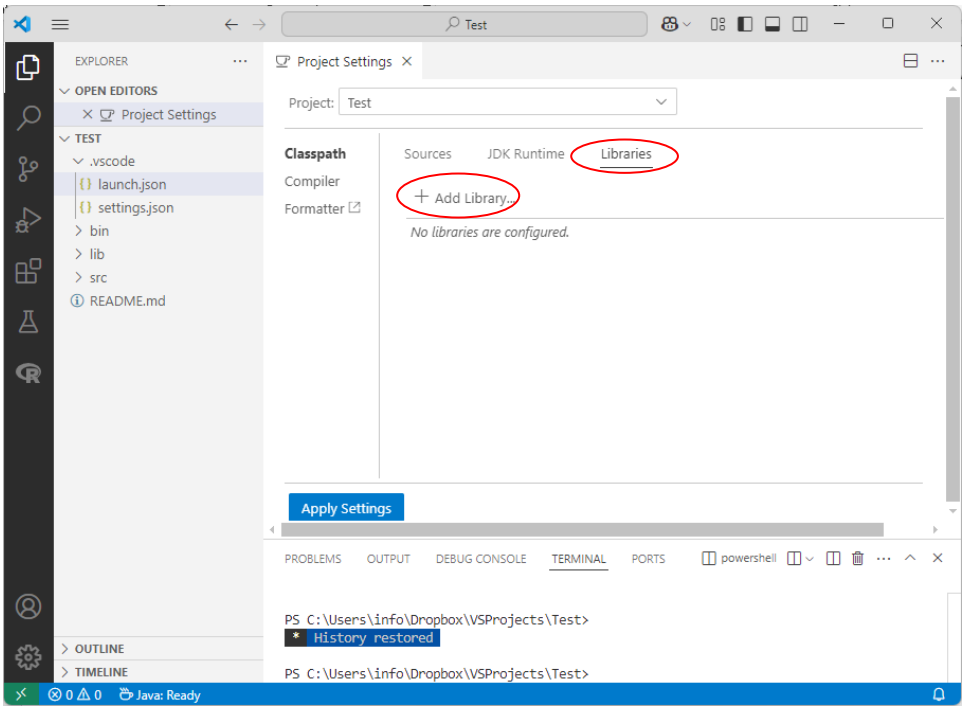


Figure 15

Click on **+ Add Library…** and navigate to the JavaFX *lib* folder that contains all the `.jar` files you downloaded earlier. Highlight these and press the *Select Jar Files* button - the files will be added shown in Figure 16.

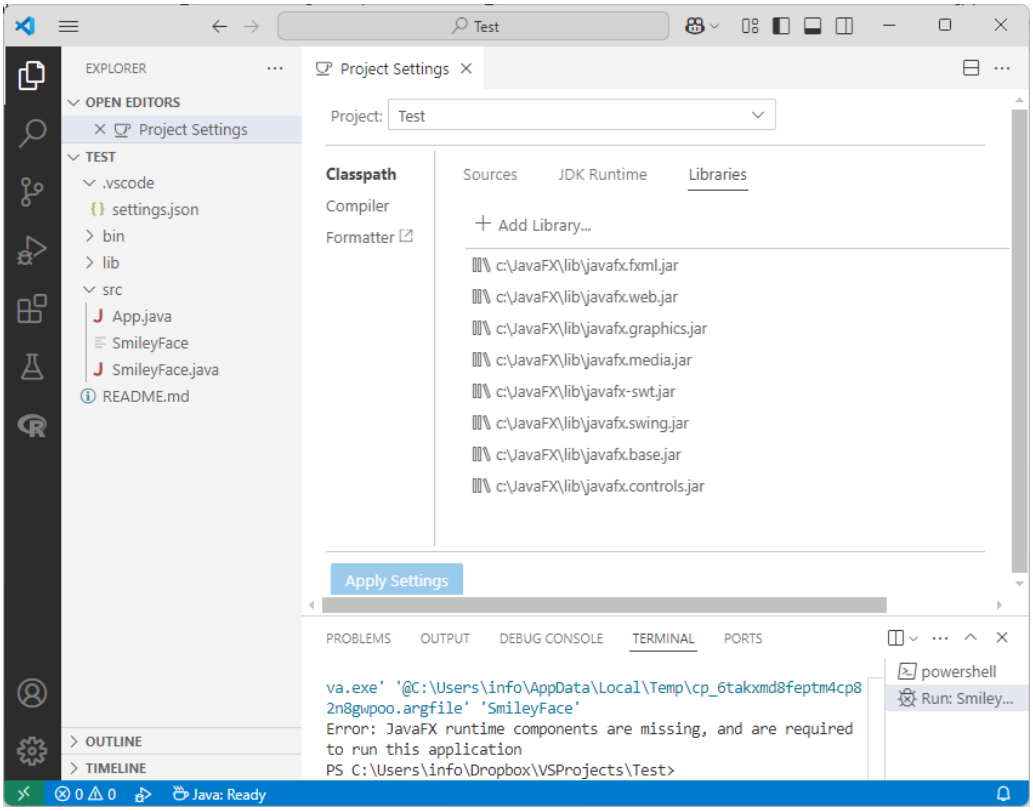


Figure 16

You should now find that your JavaFX classes compile successfully.

However if you try running the program you will get a runtime error. This happens because VS Code does not add all the modules to the module path. To fix this, press the *Run and Debug* arrow - circled in red in Figure 17.

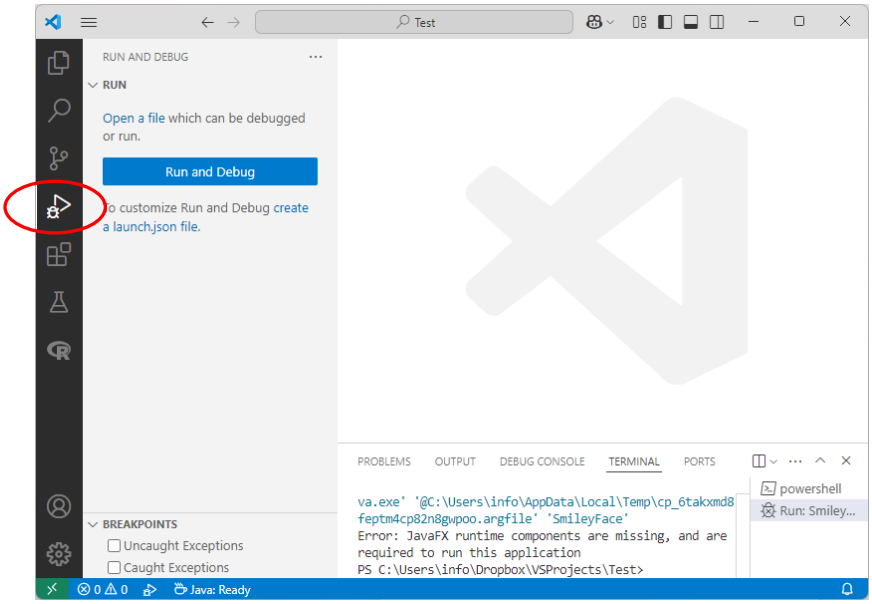


Figure 17

Just to the right of this arrow you will see the option to *create a launch.json file*. Select this option.

If the screen that appears looks like that in Figure 18, click outside of the grey window in the centre to reveal the script shown in Figure 19.

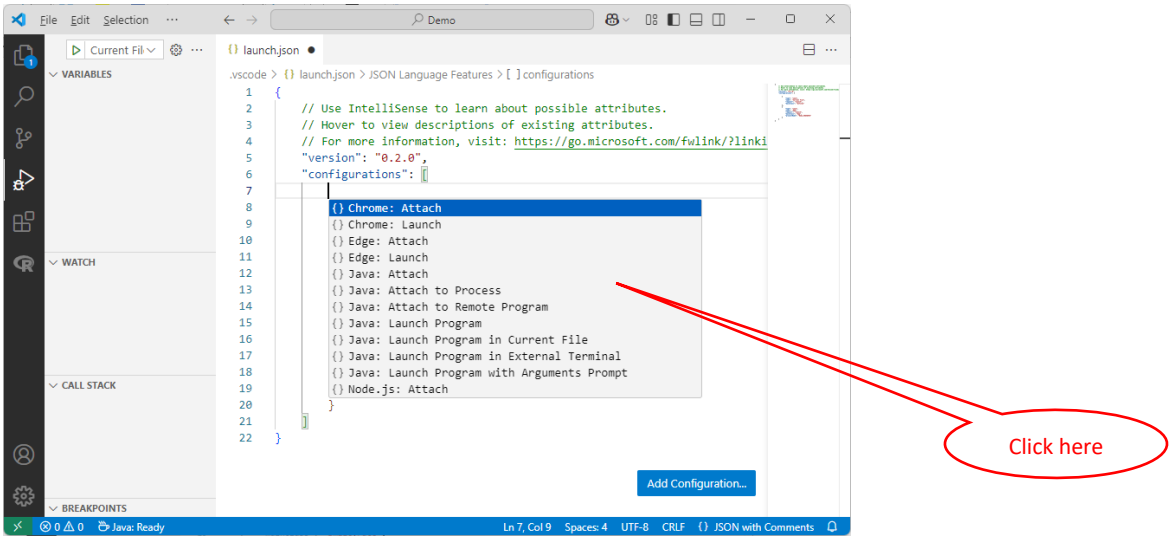


Figure 18

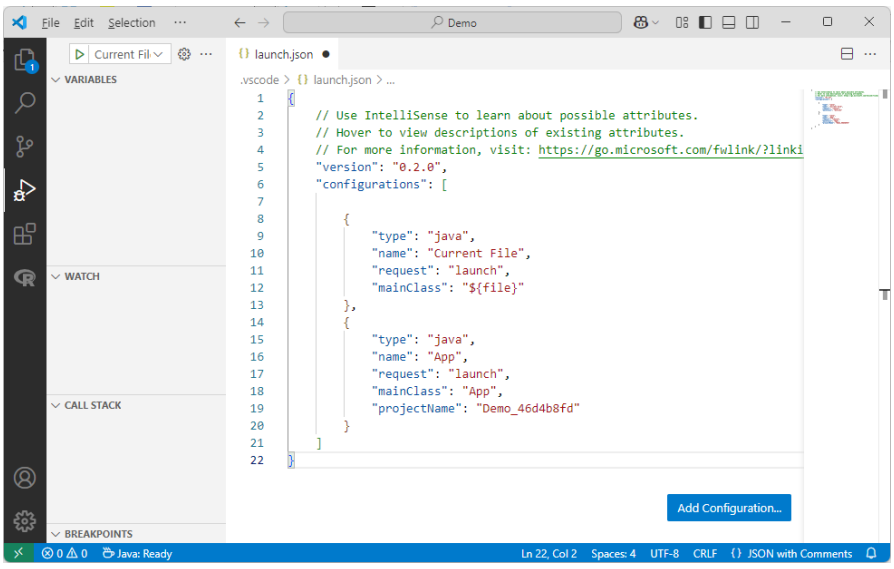


Figure 19

You will need to add a comma to the last line and then add the following line (assuming a Windows environment), if necessary replacing the path to your JavaFX SDK with your own path name.

```
"vmArgs": "--module-path C:\\JavaFX\\lib --add-modules javafx.controls,javafx.fxml"
```

Note that in a windows environment it is necessary to include an additional backslash (JSON files follow a JavaScript syntax).

In Linux or Mac environment, your pathname will look something like: **/home/user/javafx/lib**.

This is illustrated in Figure 20.

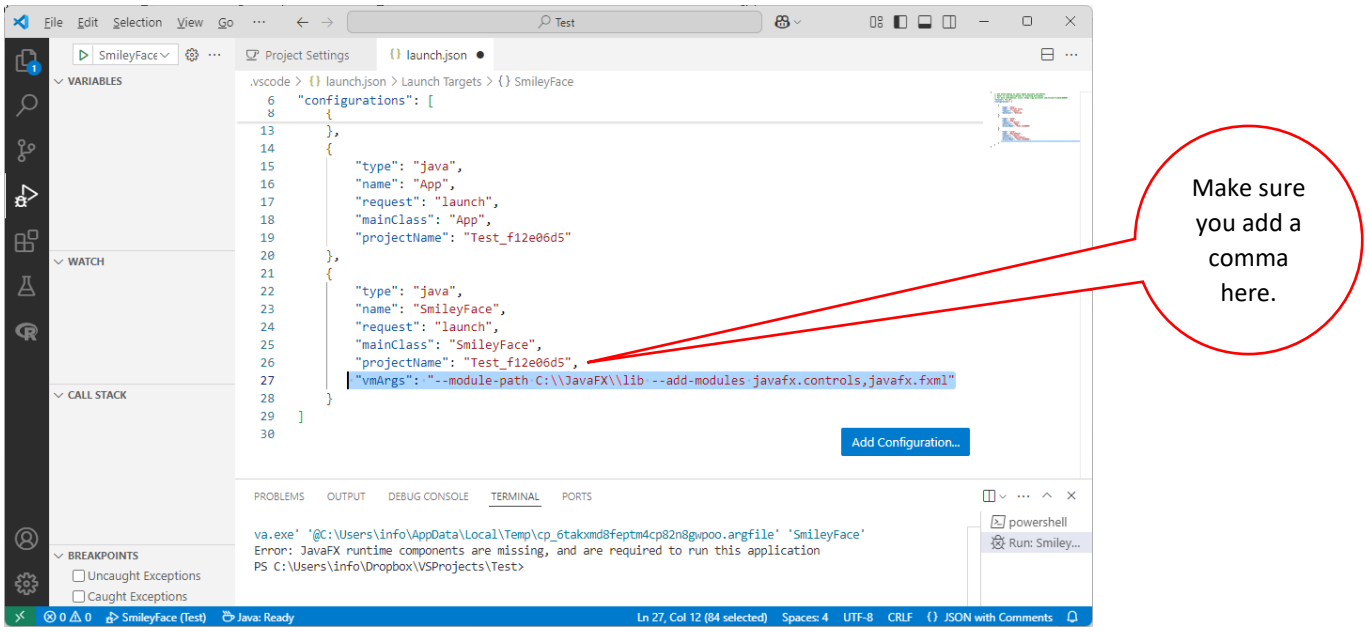


Figure 20

Finally go to the **File** menu and save this file. Your JavaFX programs should now run successfully.