Installing and Using IntelliJ

1. Installing IntelliJ

Before downloading and installing IntelliJ, 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 IntelliJ from the following site:

https://www.jetbrains.com/idea/download

The Community edition is suitable for students.

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

2. Using IntelliJ

2.1 Configuring IntelliJ

When you first open IntelliJ you will see the home screen as shown in Figure 1. You can make some initial changes to the configuration at this stage.

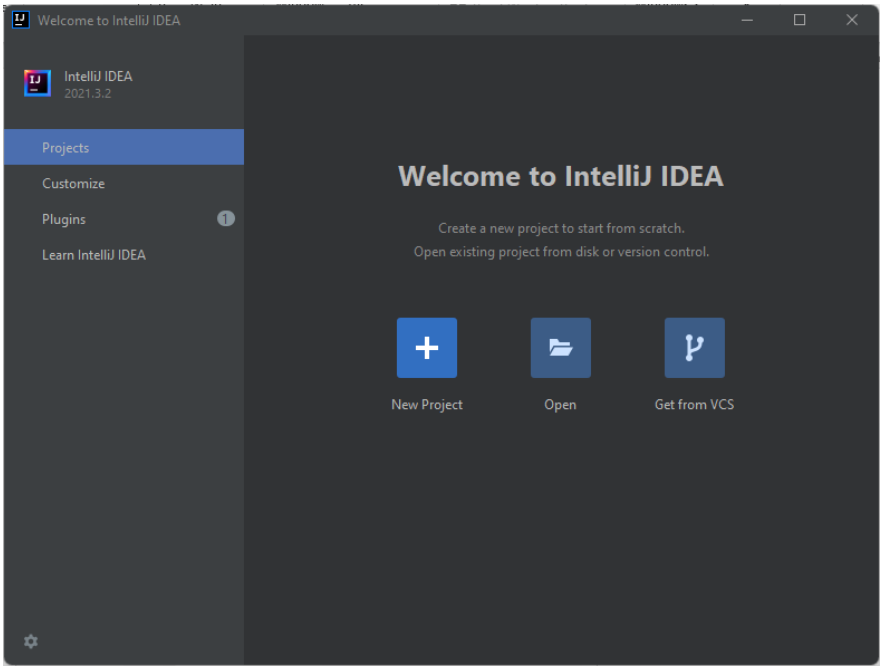

Figure 1

One of the changes you might wish to make is to change the colour theme. The initial theme, shown in Figure 1, is called *Darcula*.

By selecting **Customize** as shown in Figure 2, you can choose a new theme – we have chosen IntelliJ Light.

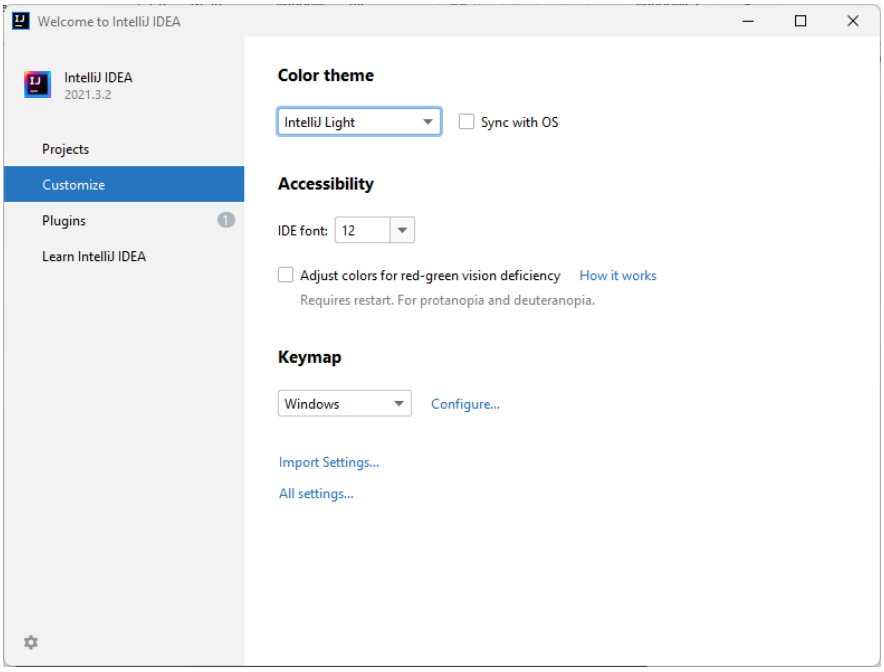

Figure 2

It is a good idea to change the code style so that the braces are lined up in a similar way to that of the book. Again choose *Customize* from the home screen, then choose **Code Style > Java** . Under *Braces Placement* make sure that *Next Line* is chosen for each option (Figure 3).

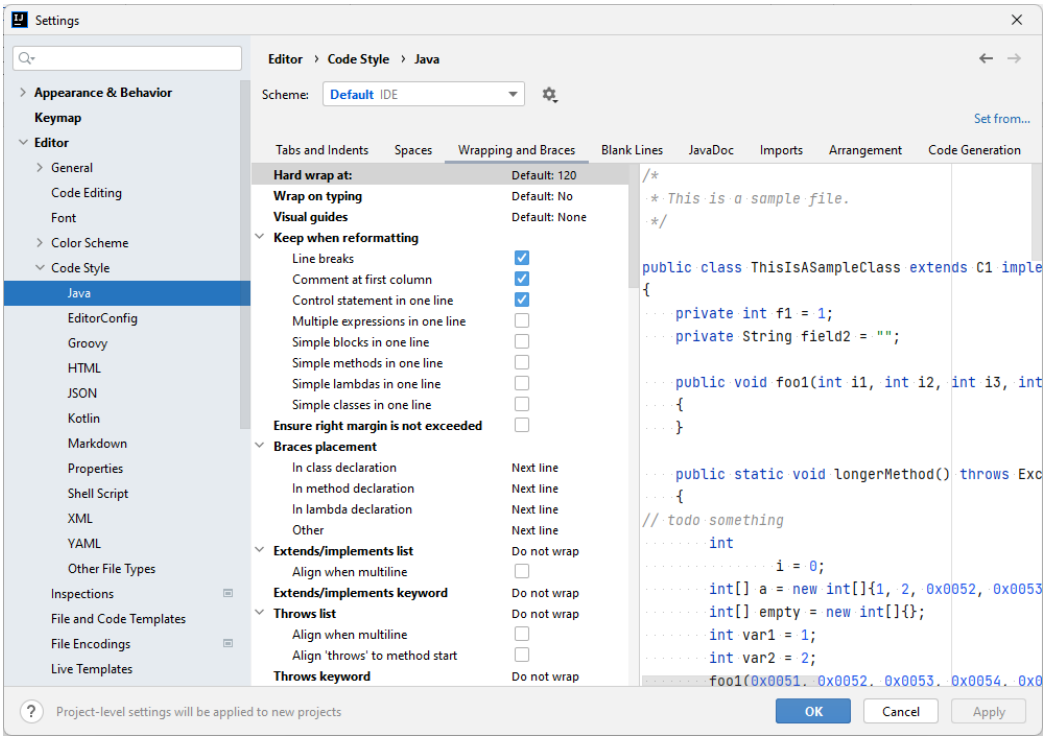

Figure 3

2.2 Starting a new project

Each project in InelliJ consists of a number of classe,s and sometimes additonal files - for examople 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, choose New Project as shown in Figure 4.

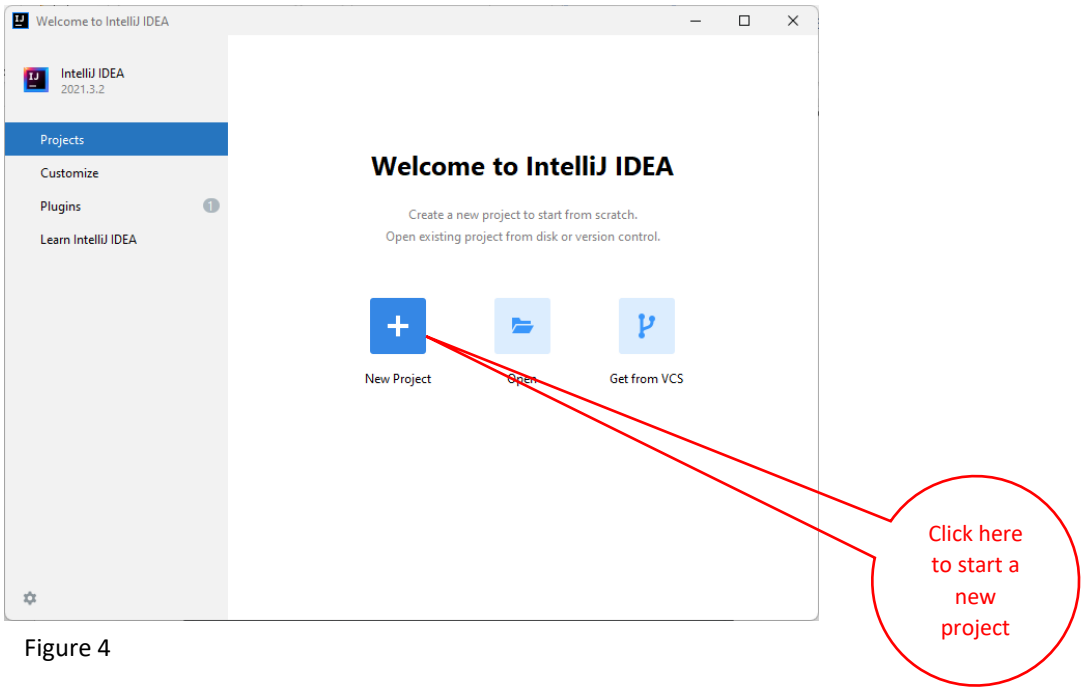

Figure 4

You will now see the screen shown in Figure 5. Choose Java from the left-hand window. As you can see, as long as you have downloaded a Java SE development kit as explained in section 1, then the JDK (or SDK as IntelliJ calls it) for the project will be selected for you. By pressing the down arrow, you can select a different one if you have downloaded more then one JDK.

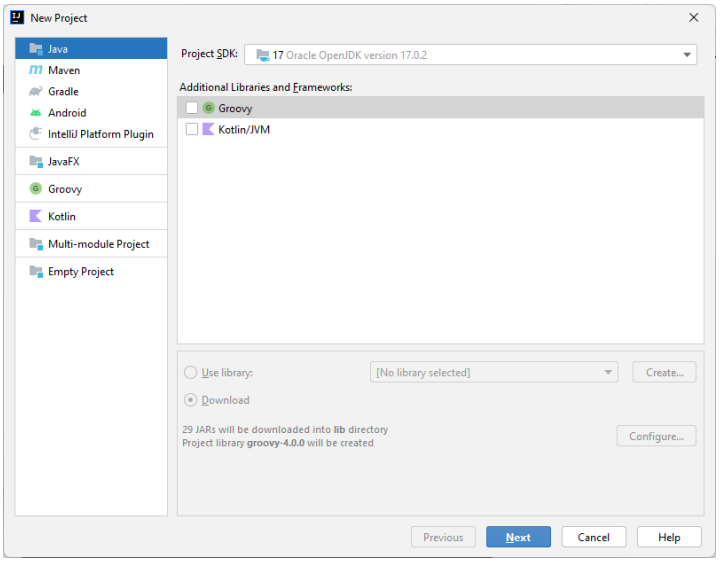

Figure 5

As long as a suitable JDK is selected, you simply continue by pressing *Next*. You will now see the screen shown in Figure 6.

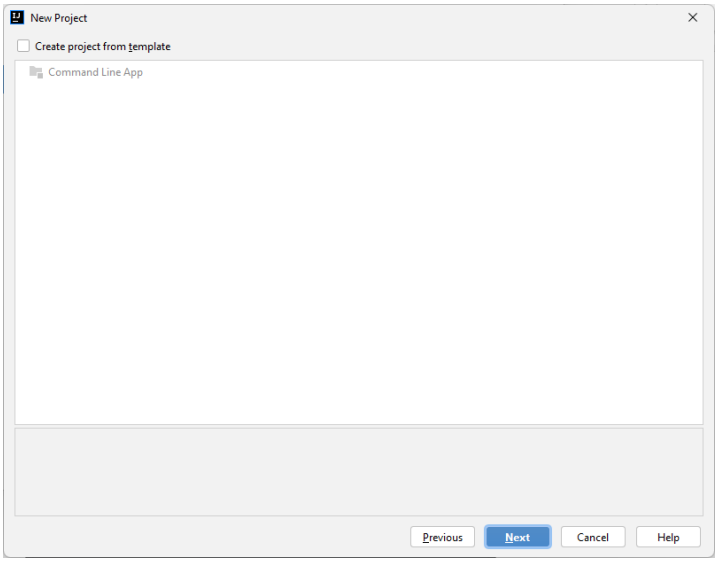

Figure 6

You will not be creating a project from a template, so you can simply continue by pressing *Next*.

You will see the screen that appears in Figure 7. Enter a suitable name for your project (we have called it *FirstProject*). You can also choose a location if you are not happy with the default location shown. Now press *Finish*.

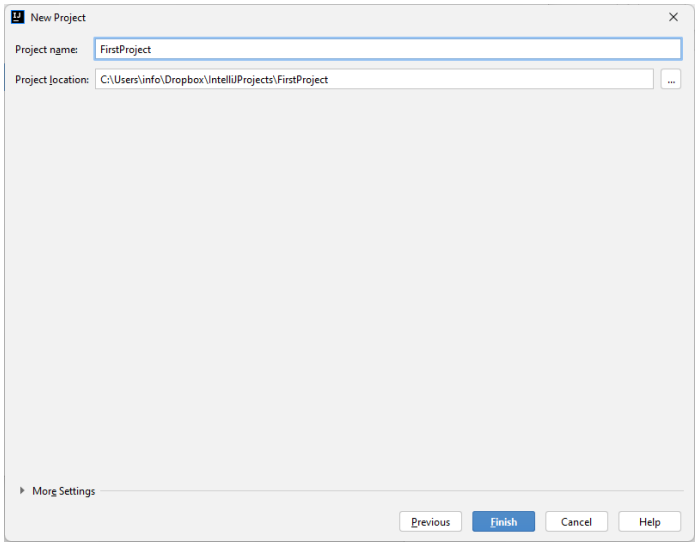

Figure 7

Your screen will now appear as in Figure 8, with *FirstProject* listed as your only open project.

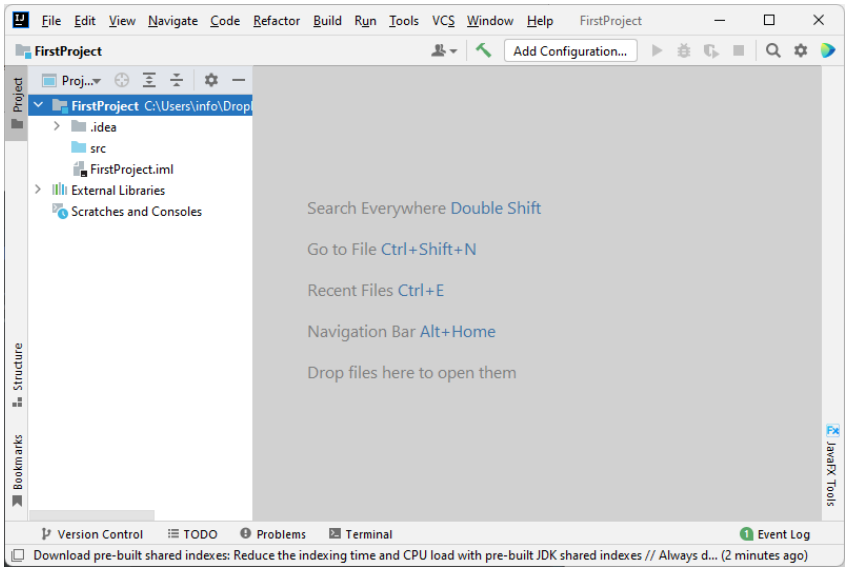

Figure 8

2.3 Adding new classes to the project

When we add a new file to the project, 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 9).

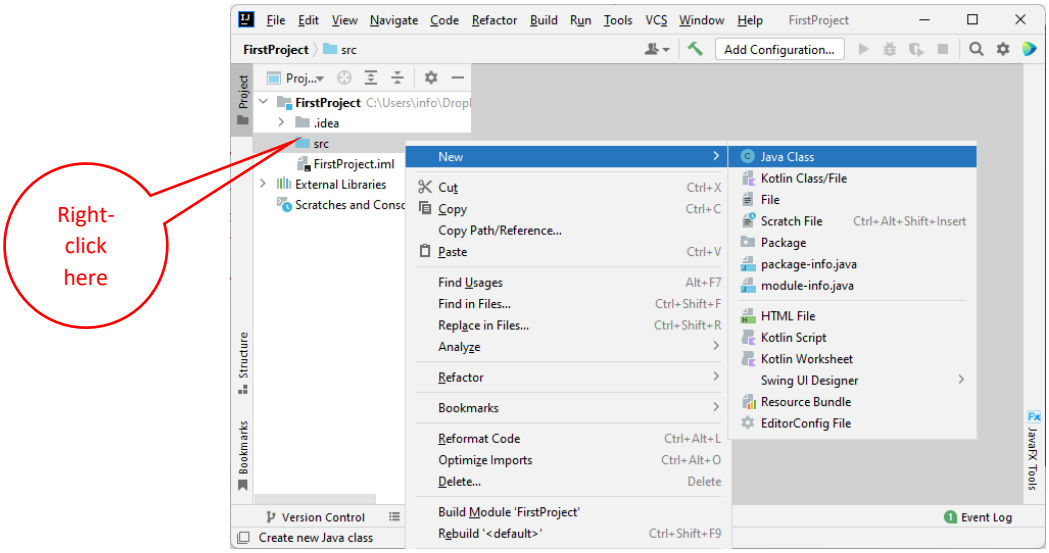

Figure 9

You will be prompted to enter the name of the class as shown in Figure 10. Type a class name of your choice, and press *Enter*.

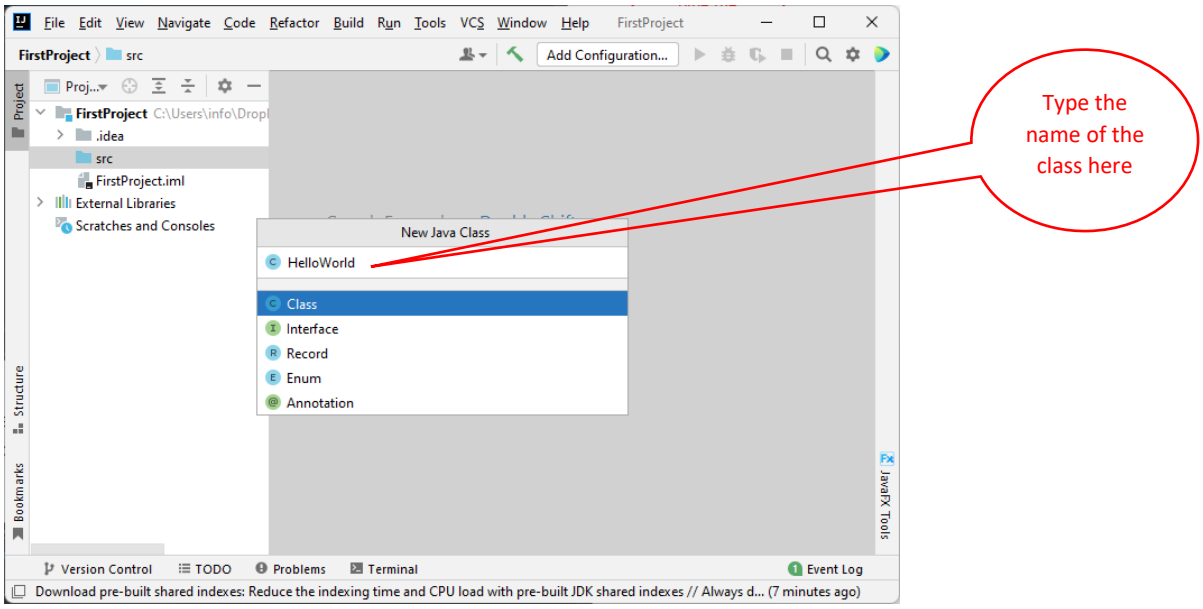

Figure 10

The screen shown in Figure 11 now appears. Your new class will be listed in the *src* folder (make sure this is expanded), and the outline code will appear as shown.

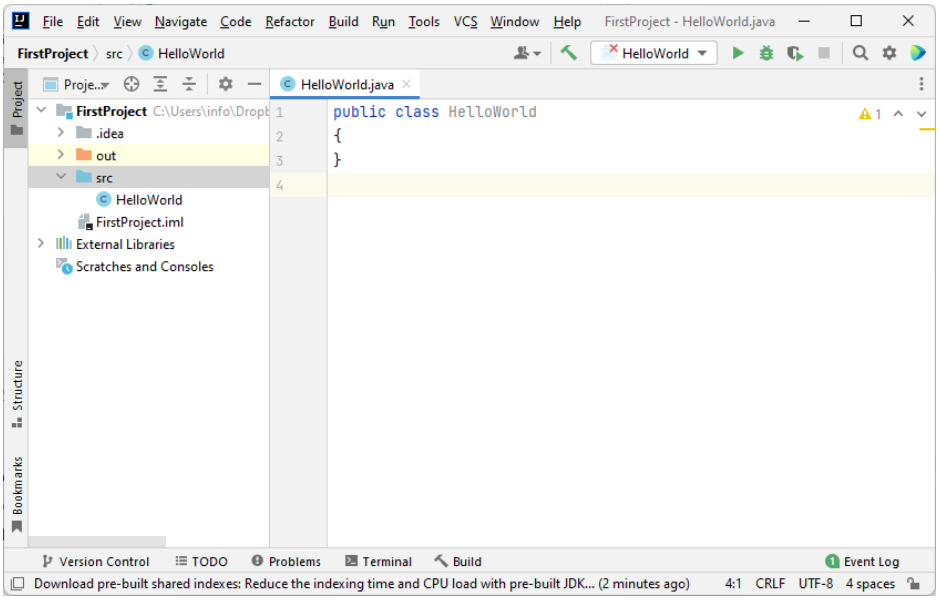

Figure 11

You can now proceed to complete the code for your class as shown in Figure 12. IntelliJ offers some useful auto-complete features, and you will see that simply typing *main* in the appropriate place, followed by the *Enter* key will automatically provide you with the header and braces for a main method.

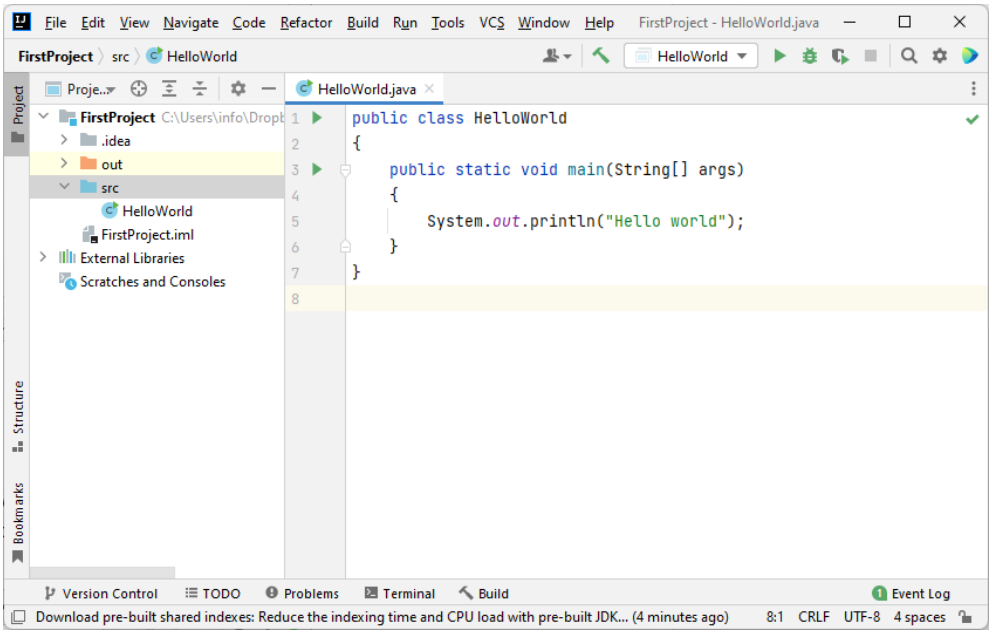

Figure 12

2.4 Compiling and running applications

Classes are automatically compiled as soon as they are written. Error messages are indicated in red and hints can be revealed as shown in Figure 13, in which the semi-colon has been omitted.

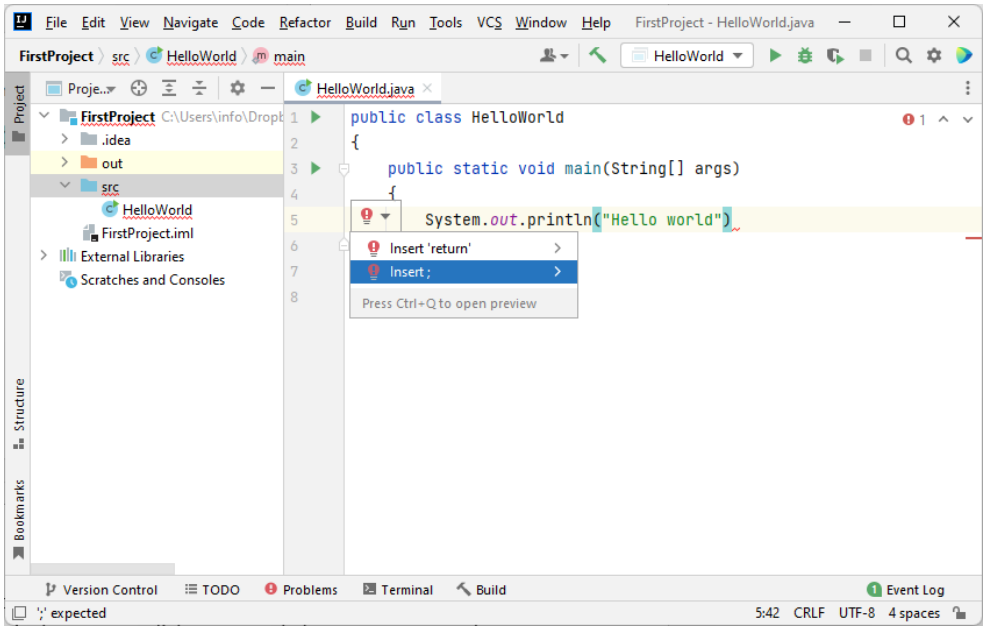

Figure 13

There are a number of different ways to run an application. Probably the easiest way is to press *shift+ctrl+F10* on the active window containing the main class. Alternatively you can click on one of the green arrows that appear in the gutter (as you can see in Figure 12) – or right-click on the class in the list and choose the *Run* option from the drop down menu (or again press *shift+ctrl+F10*).

Running a non-graphics program will cause the output to be displayed in an output window as shown in Figure 14. This window can be made to float, if you prefer (this can be achieved simply by dragging it).

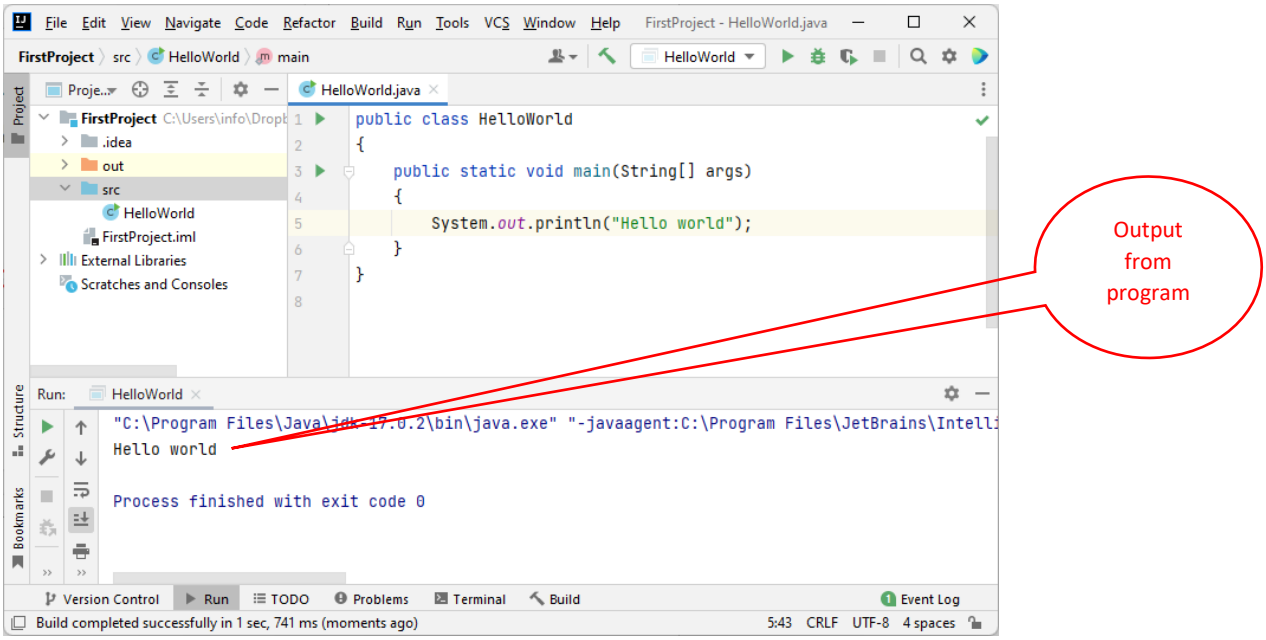

Figure 14

2.5 Adding existing classes to projects

Existing classes (that is to say, the source code) can be added to the project simply by placing them in the correct folder. The easiest way to do this is to copy or drag them into the folder within the IntelliJ environment. For example they can be placed into the src folder. All the classes will then appear in the project as shown in Figure 15.

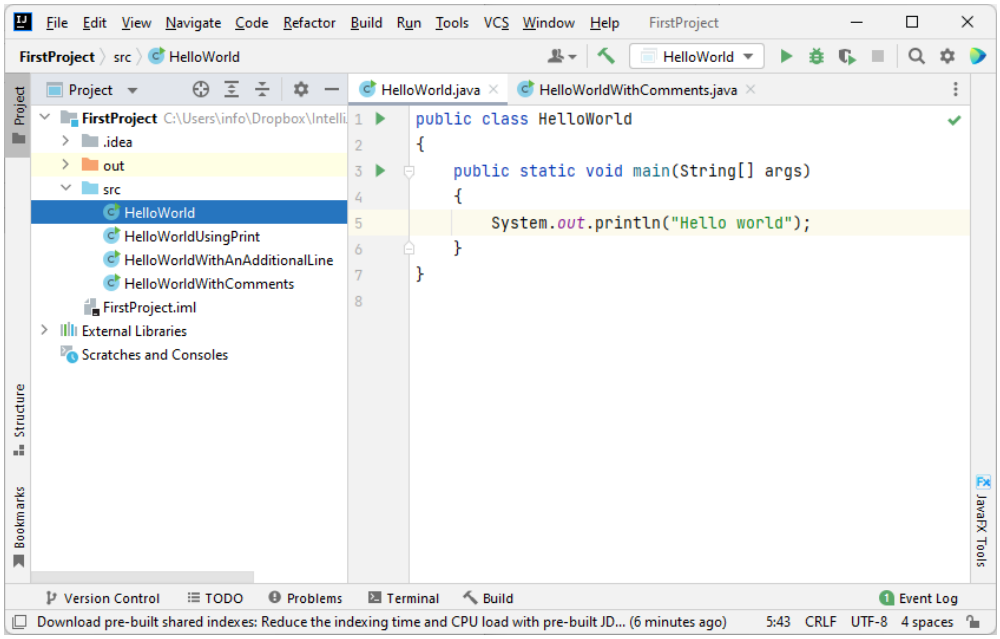

Figure 15

3 Creating JavaFX projects

There are two ways we can do this. The first is to download and use the JavaFX SDK - At the time of writing the latest stable version is JavaFX17. The second is to use the IntelliJ JavaFX plugin. Both of these are described below.

3.1 Using the JavaFX SDK

Download 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 operating 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 - do not choose a JavaFX project, just choose a regular Java project as before.

With the project open, choose File > Project Structure. Highlight Libraries, as shown in Figure 16.

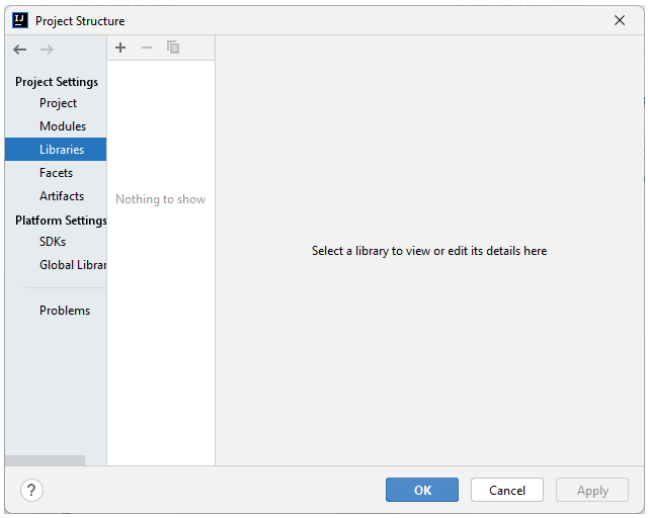

Figure 16

Now click on the + sign to create a new project library. Choose Java (Figure 17).

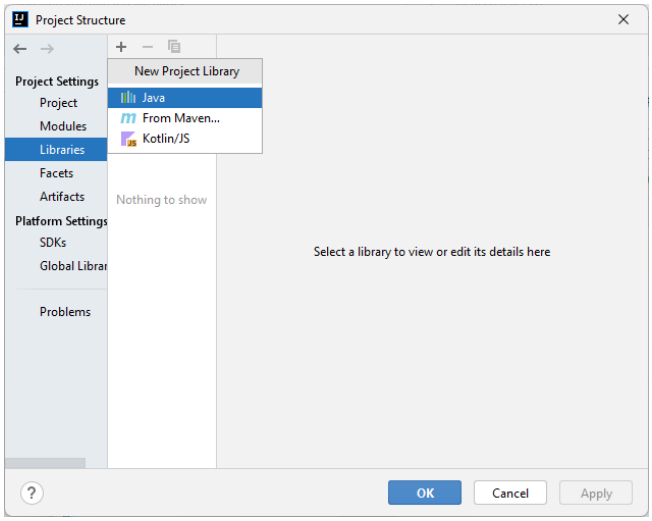

Figure 17

You will now be presented with the window shown in Figure 18. Browse to the location of the JavaFX\lib folder (or equivalent if you are using macOS, Linux etc) that you created earlier (in our case c:\javafx\lib) and select all the files from that folder as shown.

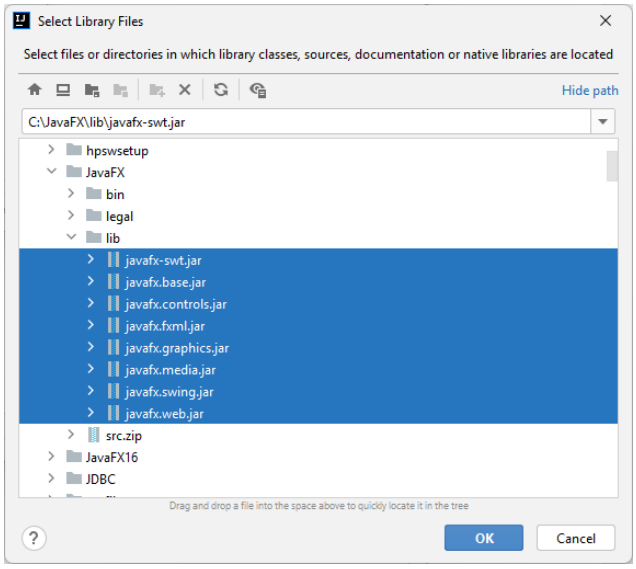

Figure 18

You can now go ahead and create a new JavaFX application. Add your new file to the *src* folder in the usual way - if you have carried out the above steps correctly, you will find that your program compiles successfully.

However if you try running the program you will get a runtime error. This happens because IntelliJ does not add all the modules to the module path. To fix this, choose **Run > Edit Configurations**. The screen shown in Figure 19 will appear.

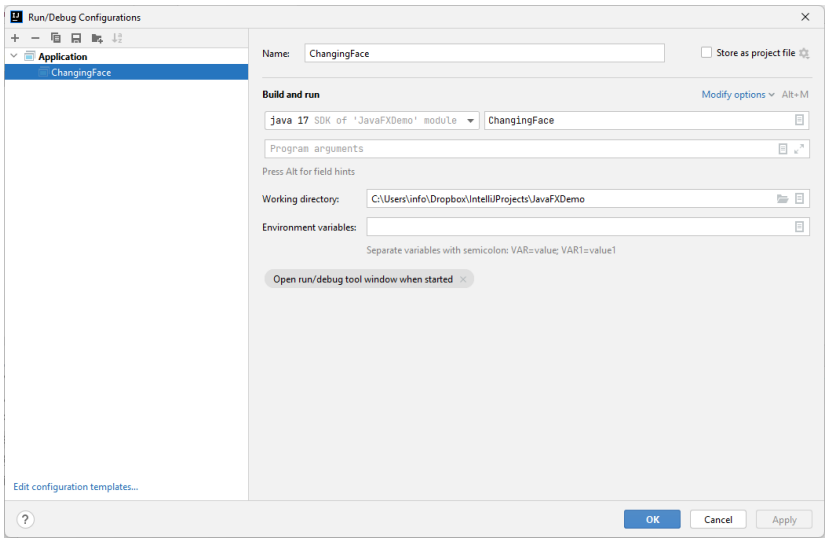

Figure 19

Click on **Modify options** (in the top right hand corner).

You will now see that that there is a box into which you can enter the VM Options (Figure 20).

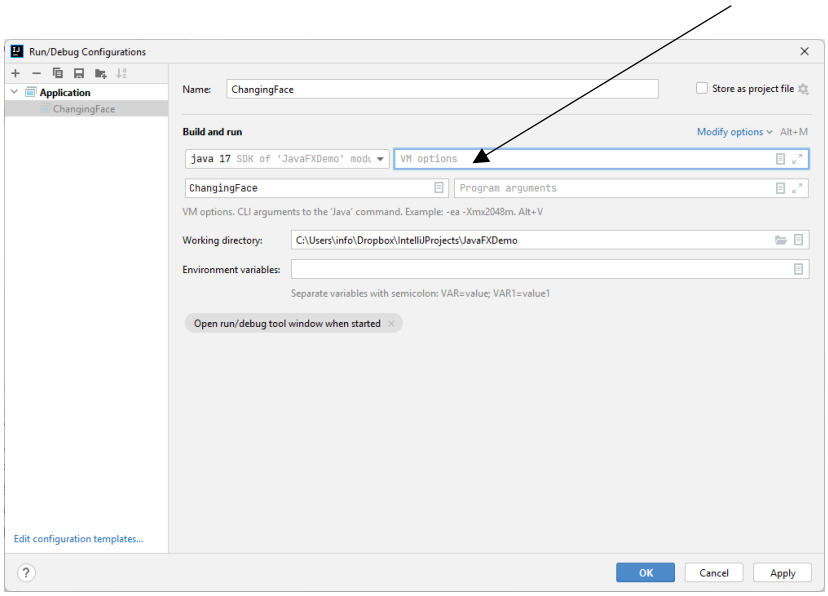

Figure 20

In this box enter the following, if necessary replacing the directory (in red) with your own location for the JavaFX files (see Figure 21) - don't forget that the path name should be enclosed in quotes if it contains spaces.

```
--module-path c:\javafx\lib --add-modules javafx.controls,javafx.fxml
```

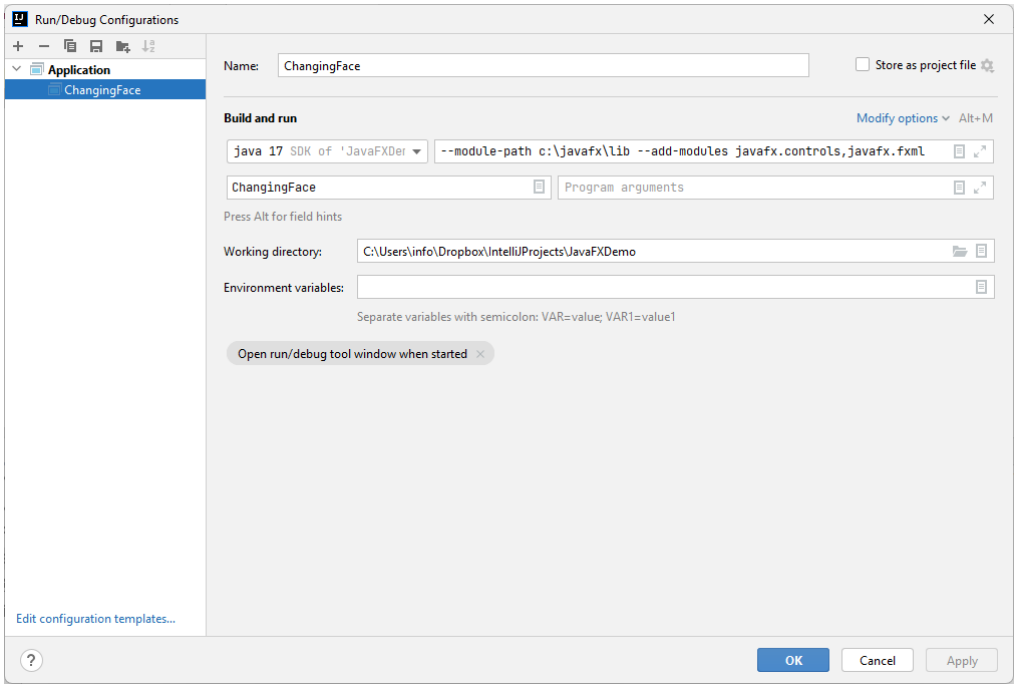

Figure 21

Your JavaFX program should run without any problems.

Don't forget that the above steps need to be carried out for each new project.

3.2. Using the JavaFX plugin

The first thing you should do is to check that the JavaFX plugin is installed. You can do this by selecting *Plugins* from the home screen as shown in Figure 22. If it is not, then make sure it is ticked.

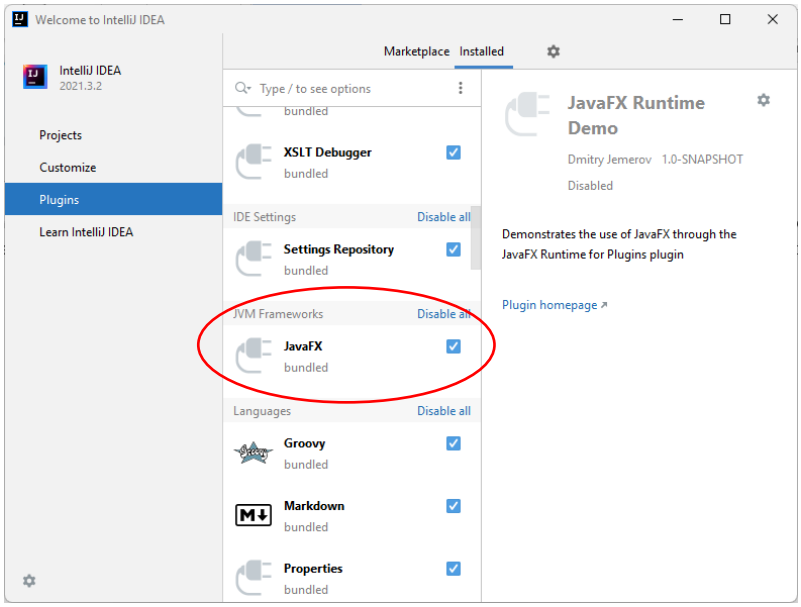

Figure 22

Now you can start the project. This time, choose a JavaFX project, rather than a Java project as shown in Figure 23 and give it a name as usual – we have chosen *FirstJavaFXProject*.

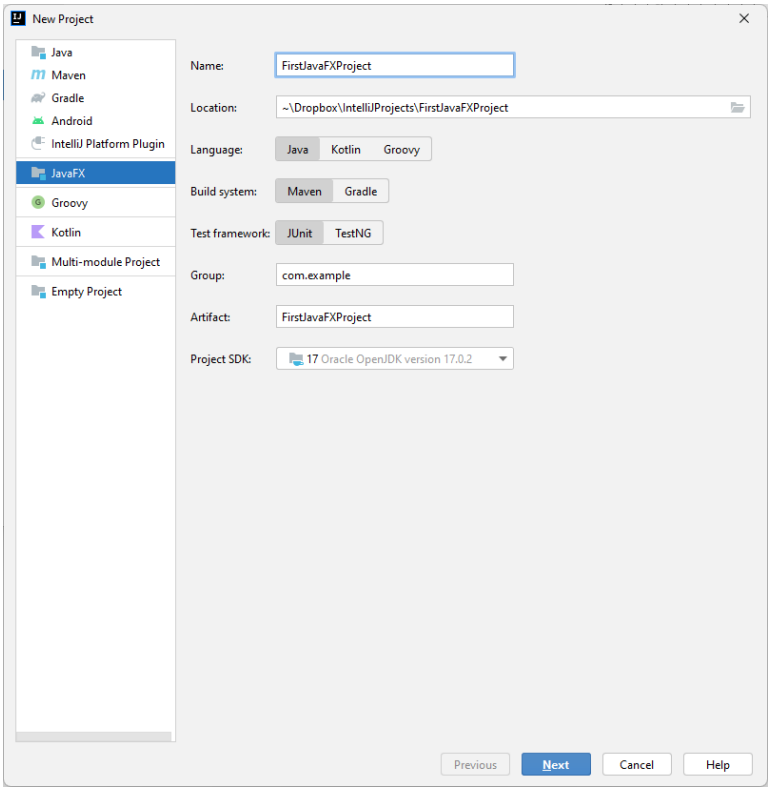

Figure 23

The screen shown in Figure 24 will appear. You do not need any of the dependencies at this stage, so you can simply proceed by pressing *Next*.

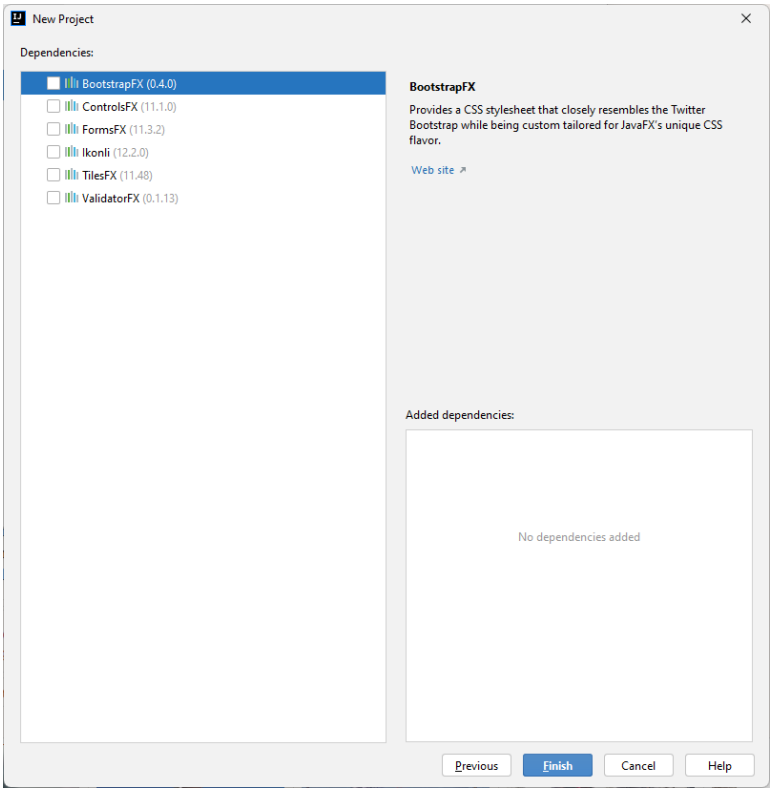

Figure 24

Your project will now be listed in the left-hand window as shown in Figure 25.

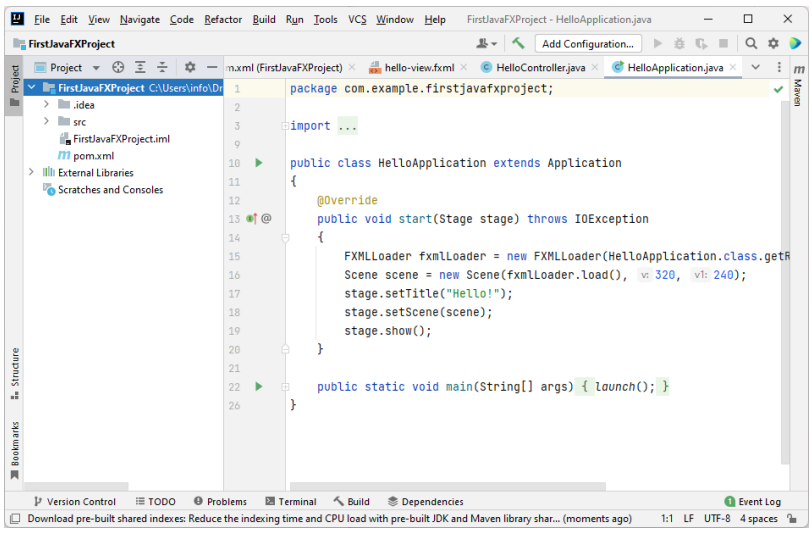

Figure 25

As you can see, a sample class called *HelloApplication* is provided for you – this can be deleted later. Initially the folders are collapsed, so it is a good idea to expand them as shown in Figure 26.

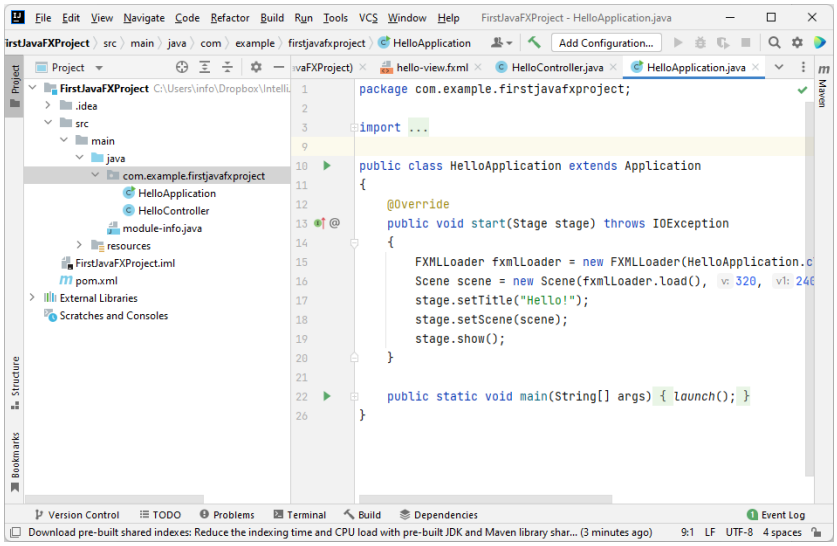

Figure 26

You can now create a new class. JavaFX projects will not allow your class to be in an unnamed package, and must be in a named package, corresponding to a folder. The easiest way to proceed is to create your class in the sample package, which will be named according to the name of the project – in our case this will be *com.example.firstjavafxproject*. We can rename this later if we want.

So right-click on this folder, choose **New > Java Class** and provide a name (*ChangingFace* in our case) – see Figure 27.

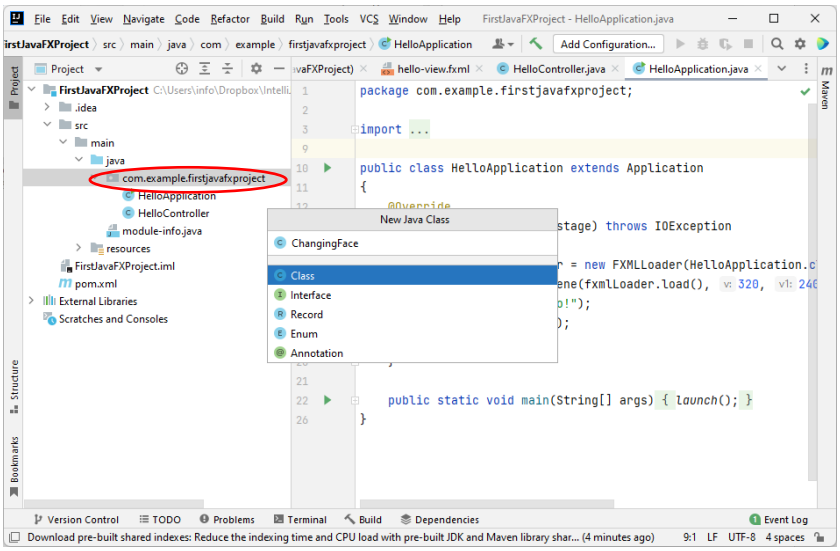

Figure 27

Figure 28 shows that your class is now listed in the correct folder and that the following required line of code has automatically been inserted for you:

```
package com.example.firstjavafxproject;
```

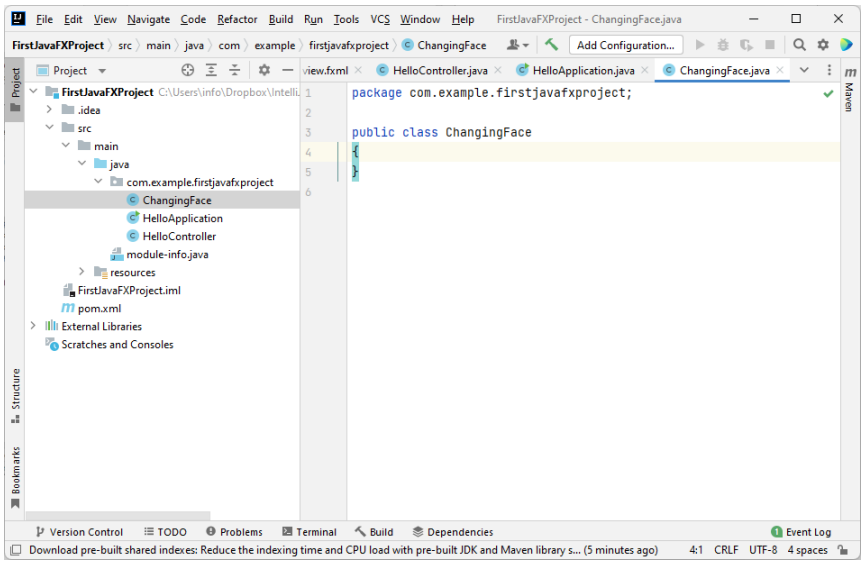

Figure 28

At this stage you can, if you wish, rename your project. Right-click on the folder as shown in Figure 29 and choose **Refactor > Rename**.

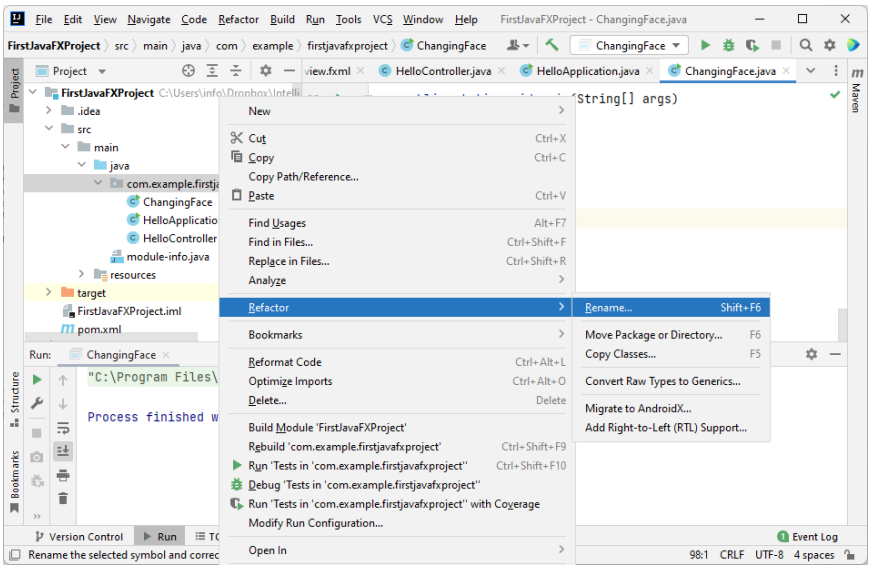

Figure 29

You will be presented with the dialogue shown in Figure 30. It is recommended that you choose *In Whole Project*, which will simplify the directory structure.

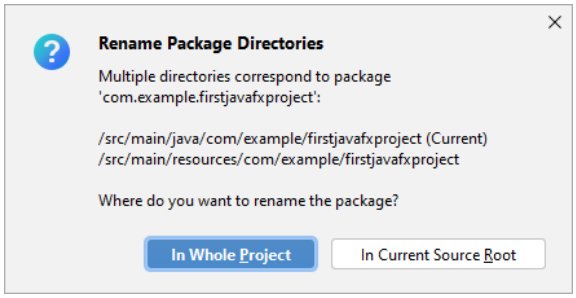

Figure 30

You will now be able to choose a new name for your package (Figure 31).

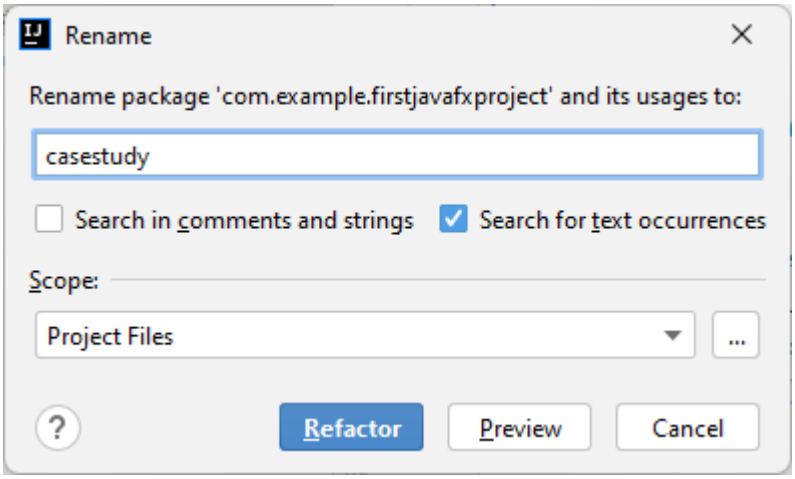

Figure 31

If you enter a simple package name such as *casestudy* without a full domain-style structure, IntelliJ will relocate the package accordingly, as shown in Figure 32. As you can see the package instruction in the class has also been changed.

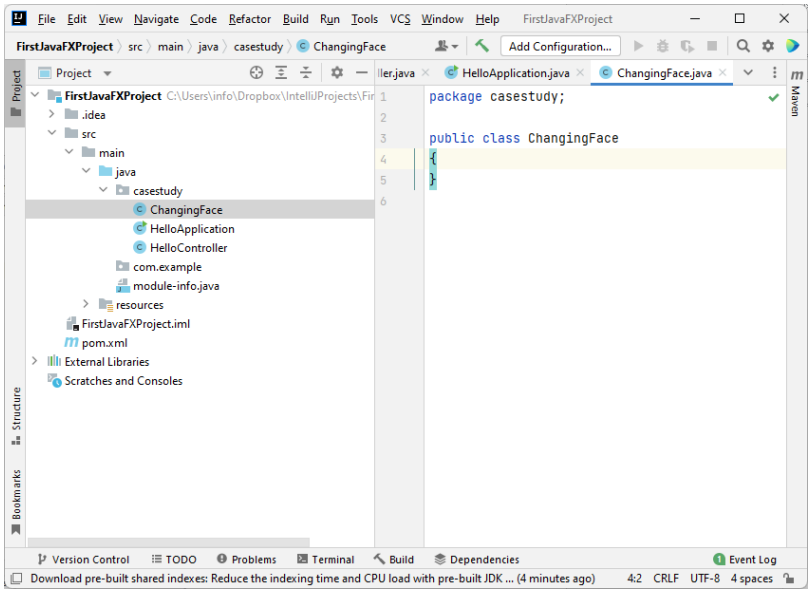

Figure 32

You can now go on to write the code for your class, and to run your application as before. You can also delete the sample classes *HelloApplication* and *HelloController* if you wish.