

Programming in Two Semesters

Charatan and Kans

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. To install the latest release, use the following link:

<https://www.oracle.com/technetwork/java/javase/downloads/>

IntelliJ™ makes it very easy to switch between different versions of the JDK, so you might also want to download Java 8 (the JDK is referred to as 1.8), as this version makes running JavaFX programs (chapters 23 and 24) a little more straightforward – see section 3. Java 8 can be downloaded from here:

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

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.

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.

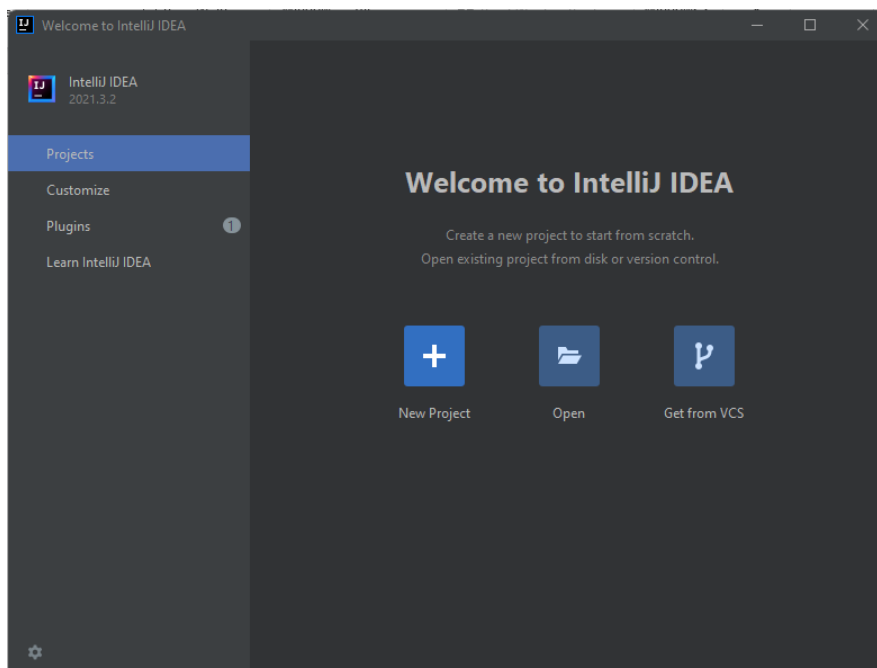


Figure 1

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.

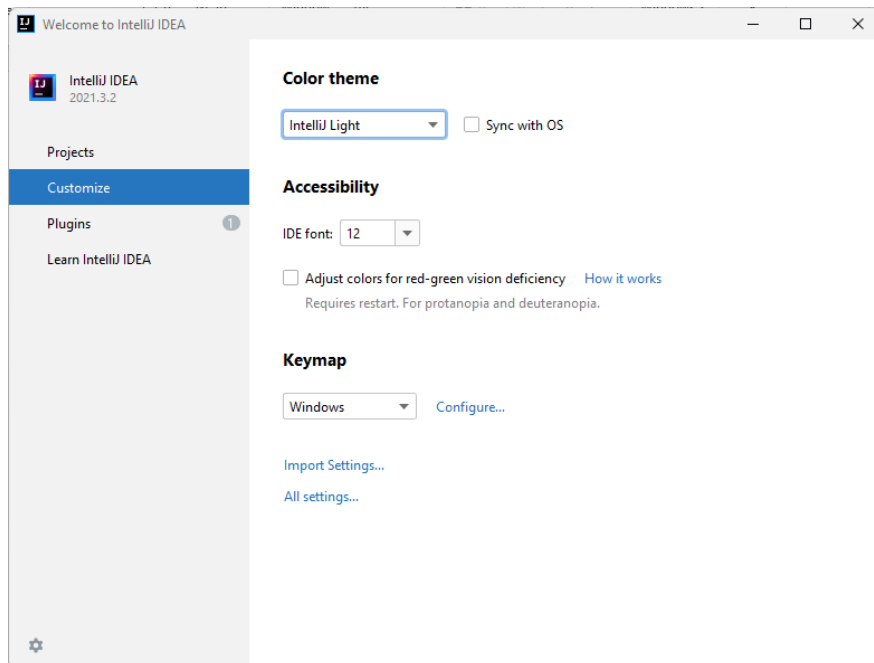


Figure 2

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).

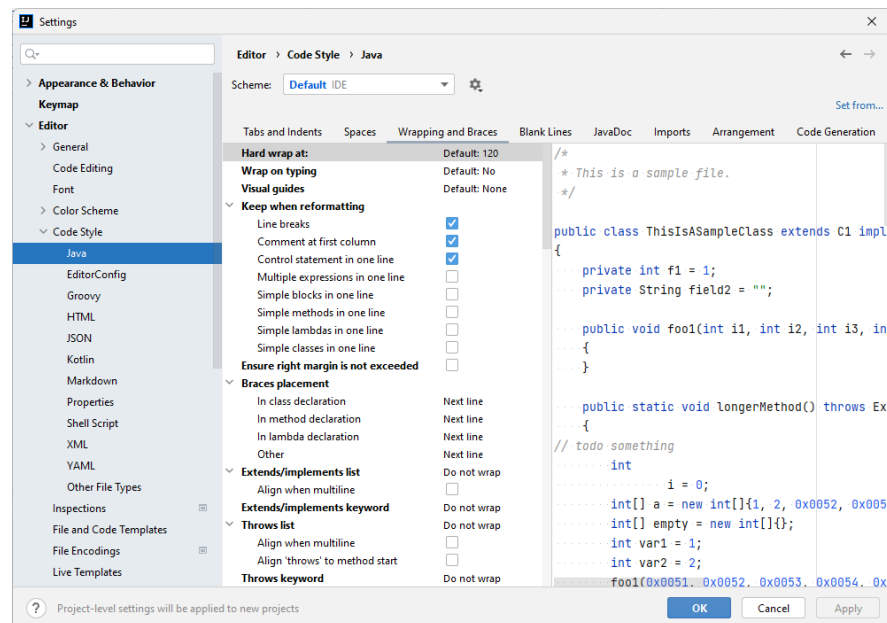


Figure 3

2.2 Starting a new project

All the files associated with an application should be kept in a single project. It is recommended that you create a separate project for each application that you are creating. To create a project, chose *New Project* as shown in figure 4.

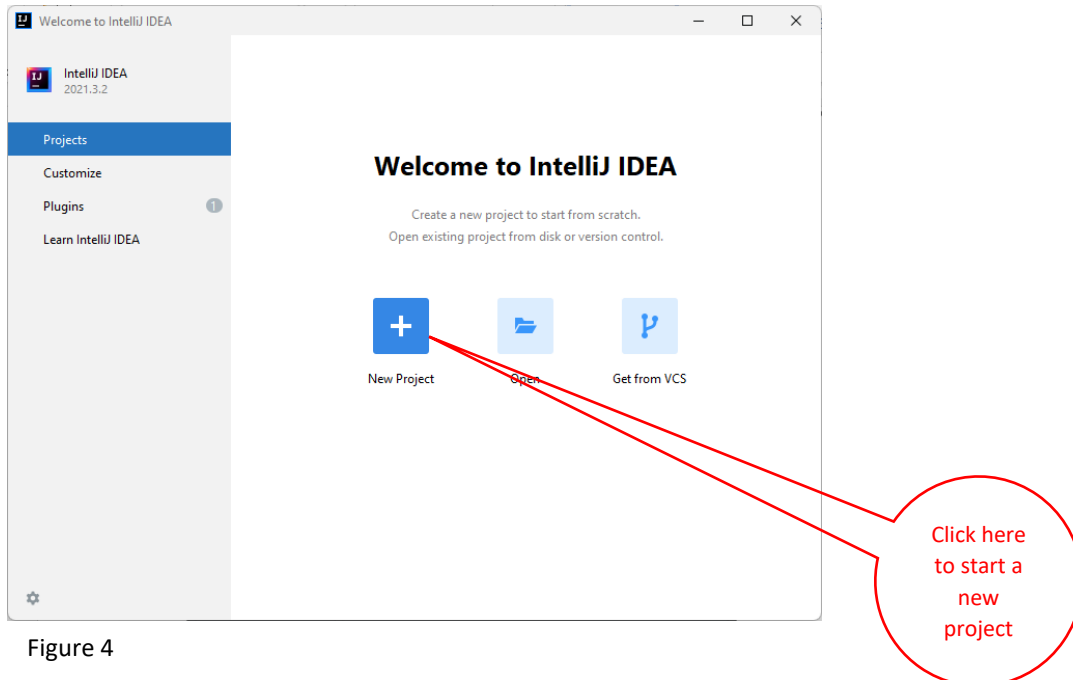


Figure 4

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. As we go on to explain in section 3.1, not only does IntelliJ™ make it very easy to choose a particular JDK when creating a project, it also makes it easy to change the JDK for that project at any time.

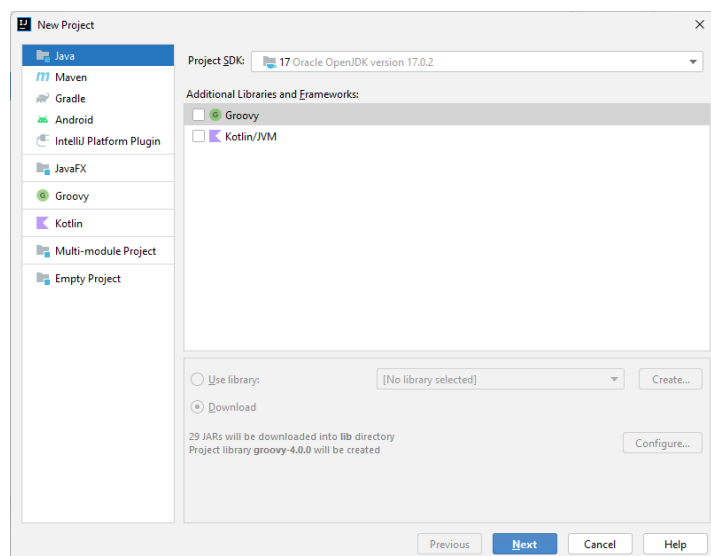


Figure 5

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

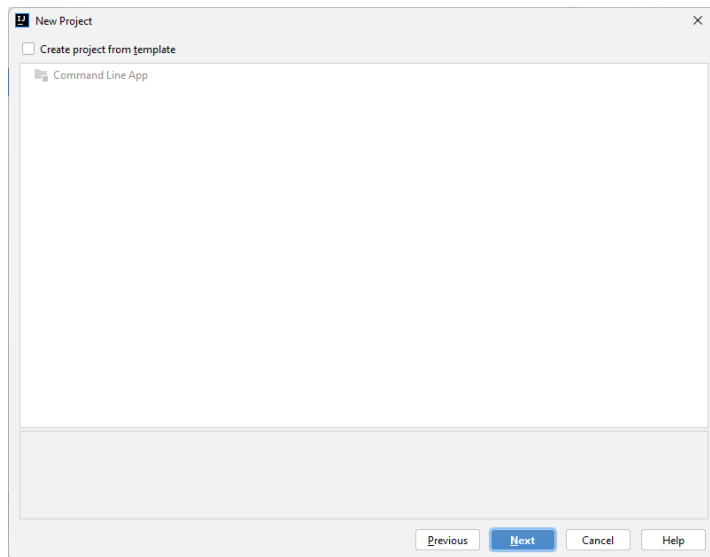


Figure 6

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 do not want the default location shown. Now press *Finish*.

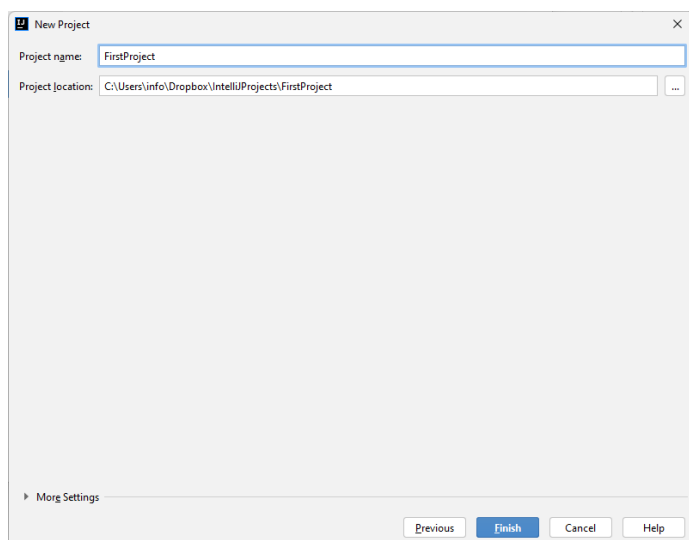


Figure 7

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

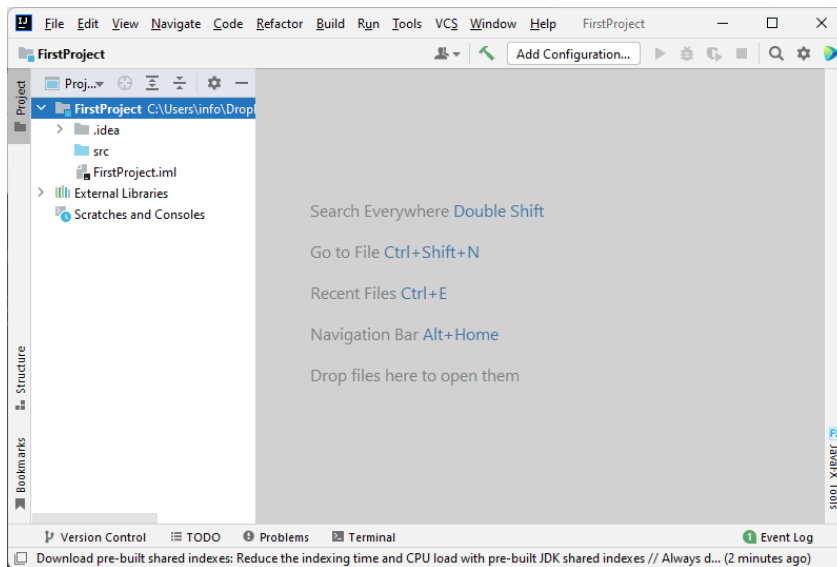
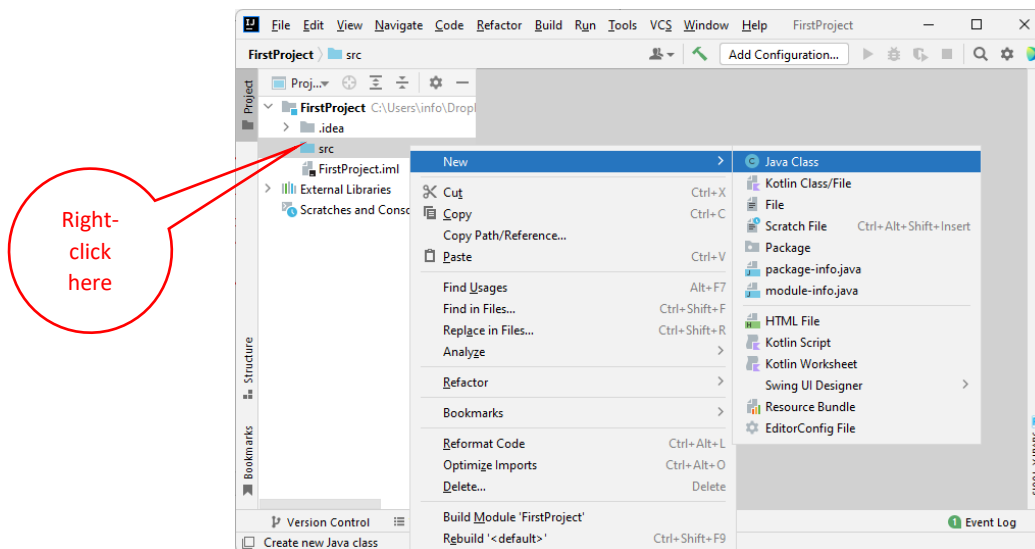


Figure 8

2.3 Adding new files 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).



You will be prompted to enter the name of the class as shown in figure 10. Enter the name of your choice, followed by the *Enter* key.

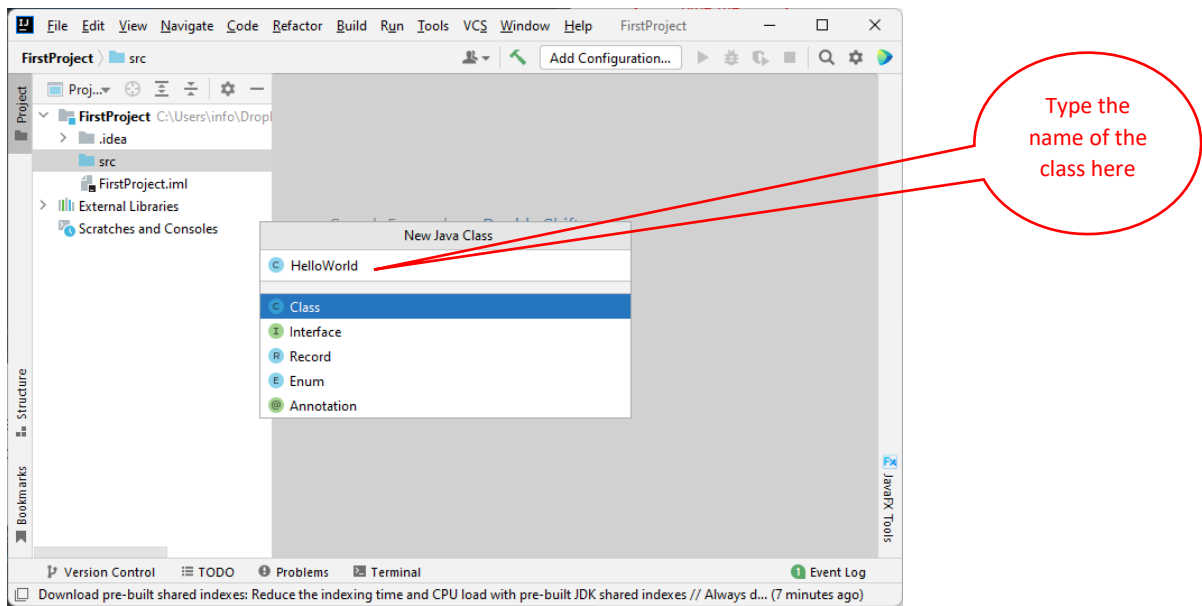


Figure 10

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.

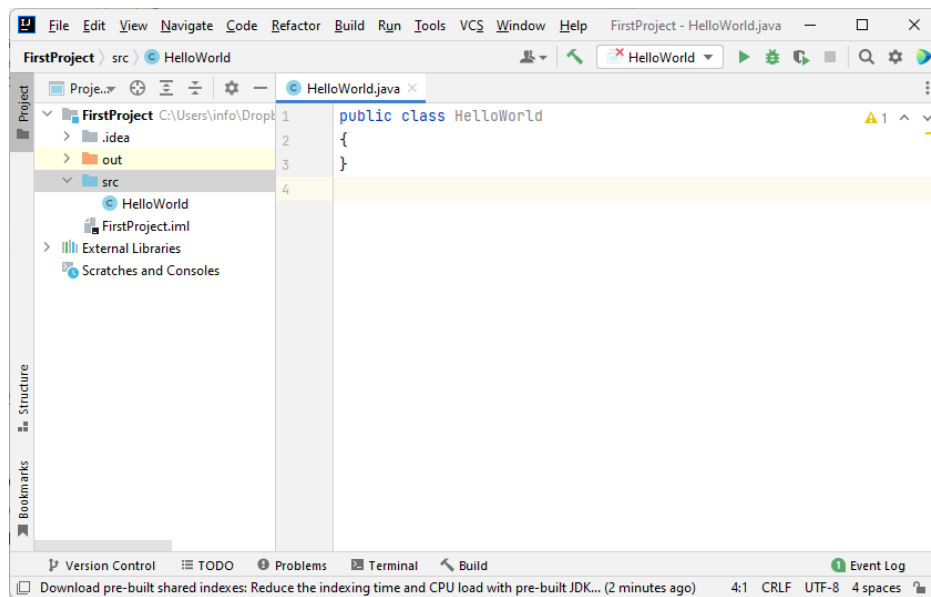


Figure 11

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.

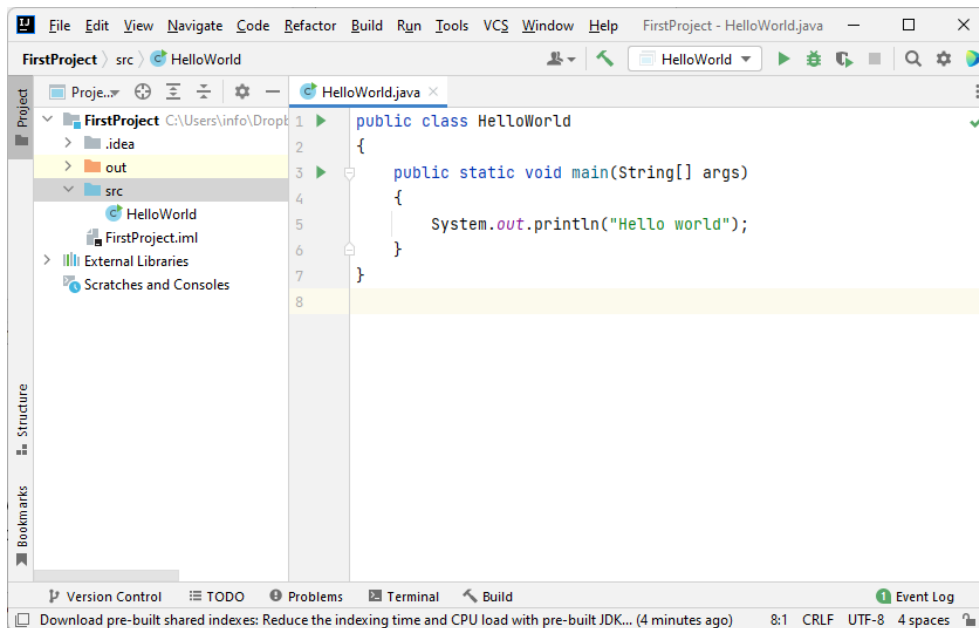


Figure 12

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.

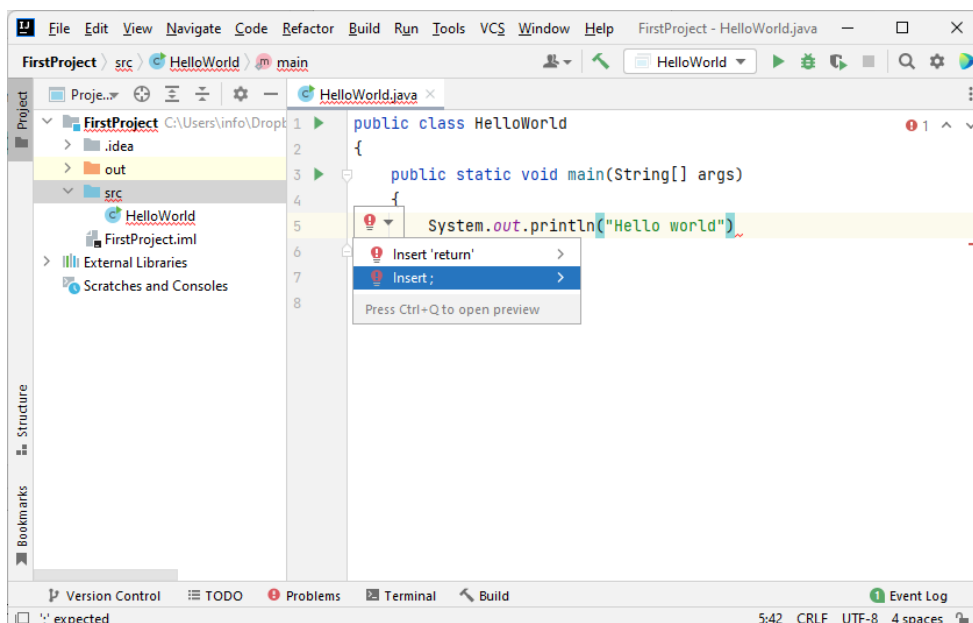


Figure 13

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

3 Creating JavaFX projects

3.1 Creating JavaFX projects with Java 8 (very easy for beginners)

As we explained in section 2.2, it is very easy to choose a particular SDK (SDK) when creating a project. It is also an easy matter to change the JDK for an existing project.

With the project open, choose **File | Project Structure**. You will see the screen shown in figure 16 (make sure you have *Project* highlighted as shown).

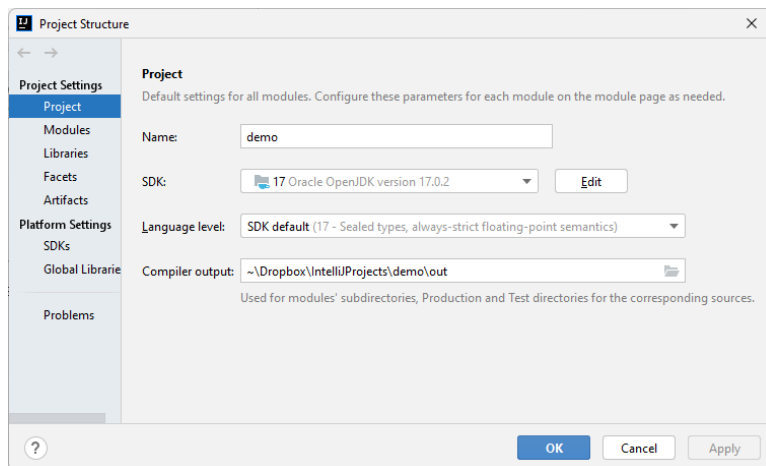


Figure 16

Clicking on the down arrow in the SDK section will enable a drop-down menu as shown in figure 17

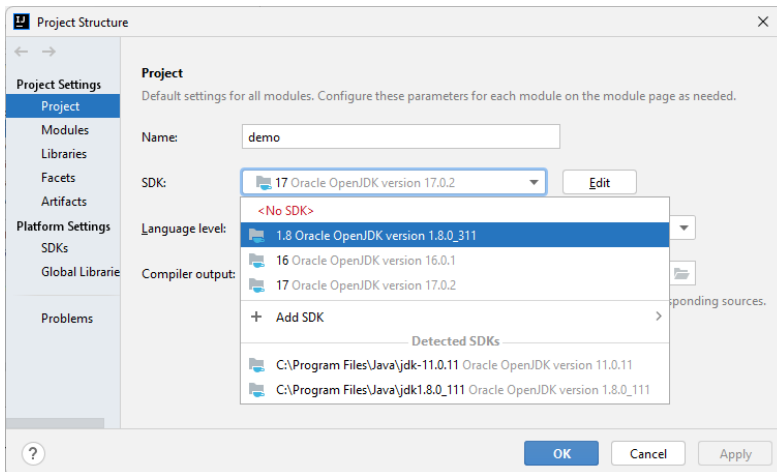


Figure 17

As long as you have downloaded JDK 1.8 as described in section 1, you will be able to select this option and proceed with your project. With the JDK 1.8 selected you can then proceed exactly as with any other project, as described above. In this case, simply choose a Java project as before, not a JavaFX project.

3.2 Creating JavaFX projects with Java 11 or above

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 18. If it is not, then make sure it is ticked.

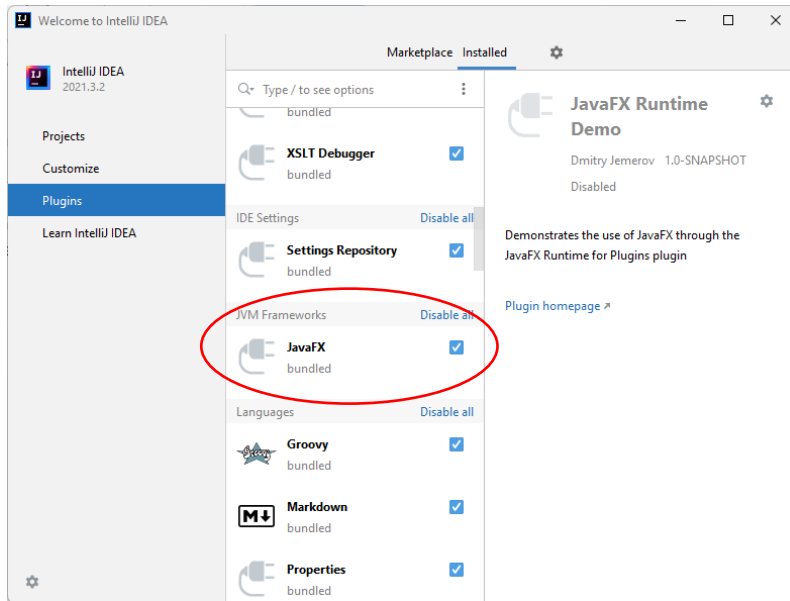


Figure 18

Now you can start the project. This time, choose a JavaFX project, rather than a Java project as shown in figure 19 and give it a name as usual – we have chosen *FirstJavaFXProject*.

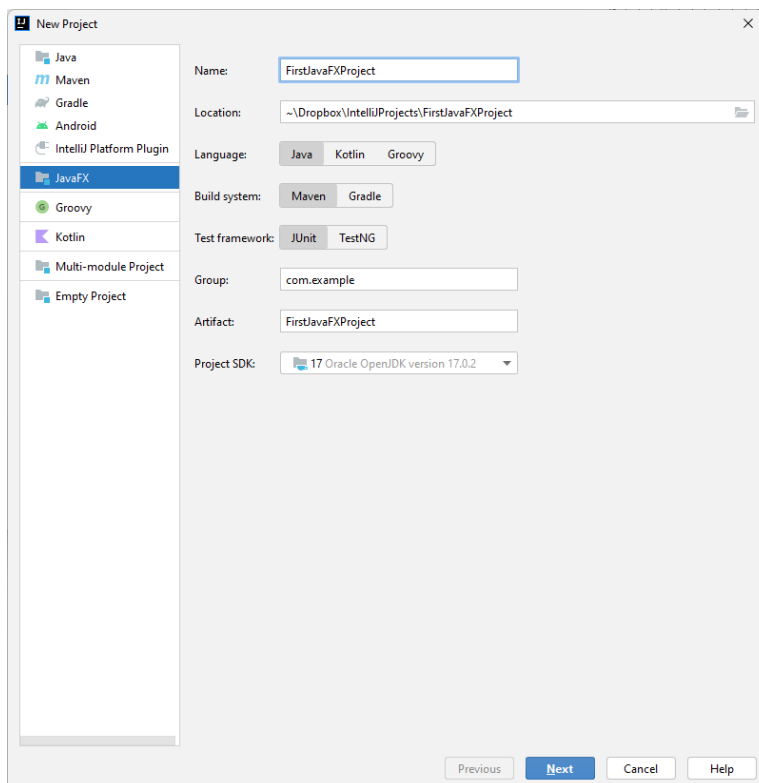


Figure 19

The screen shown in figure 20 will appear. You do not need any of the dependencies at this stage, so you can simply proceed by pressing *Next*.

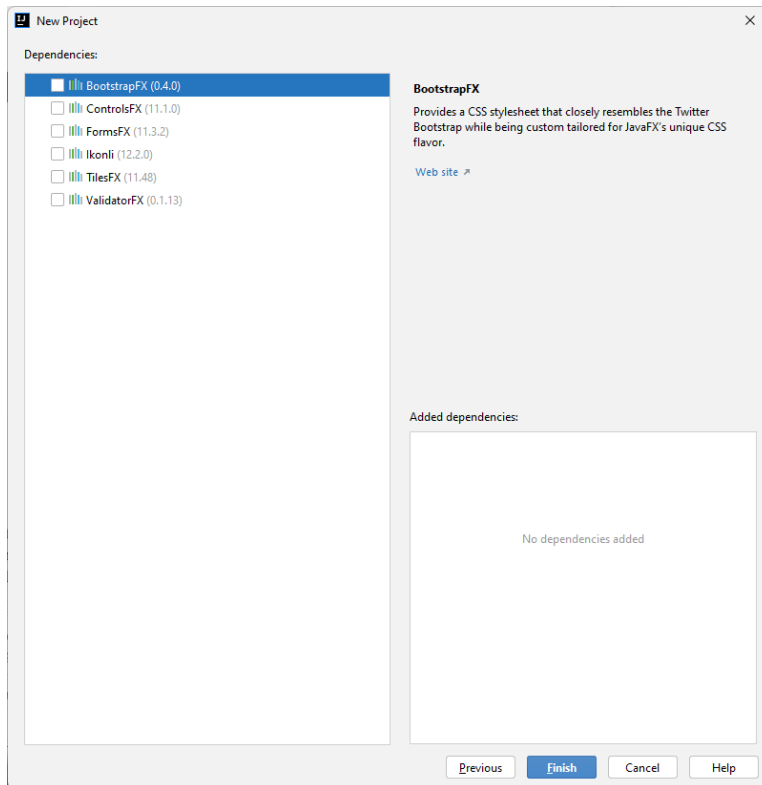
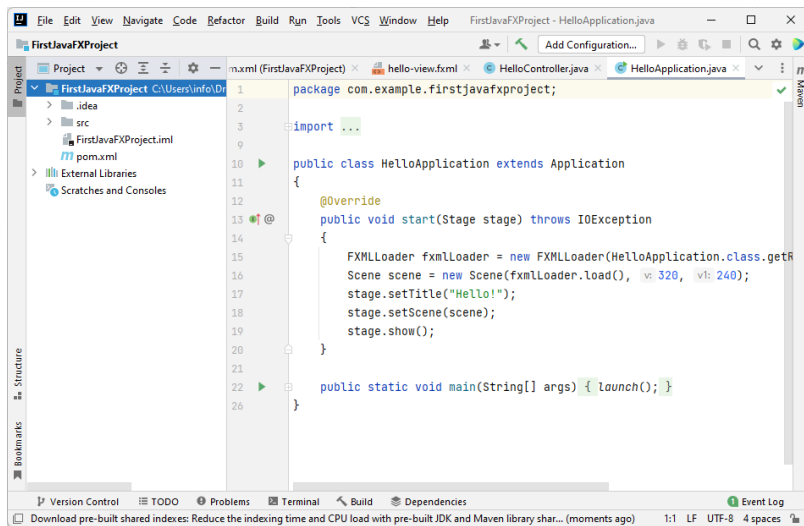


Figure 20

Your project will now be listed in the left-hand window as shown in figure 21.



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 21.

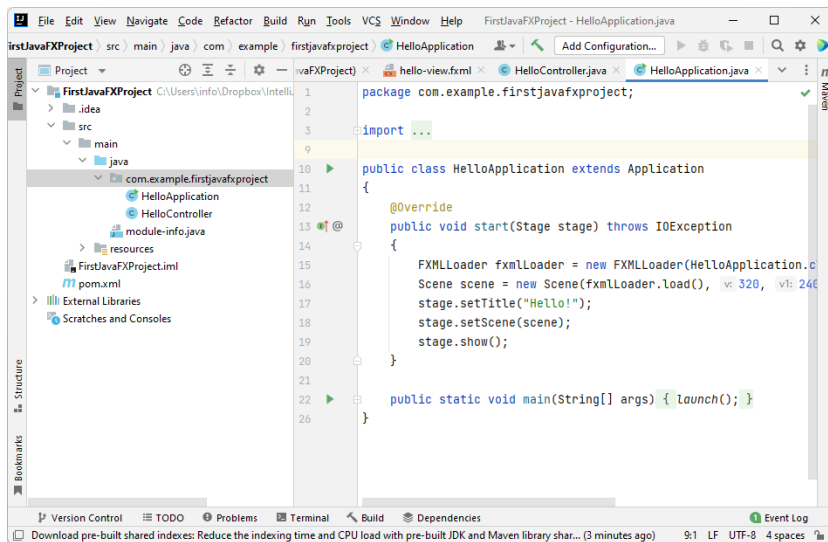


Figure 21

You can now create a new class. JavaFX projects will not allow your class to be in an unknown package (see chapter 12), 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 22.

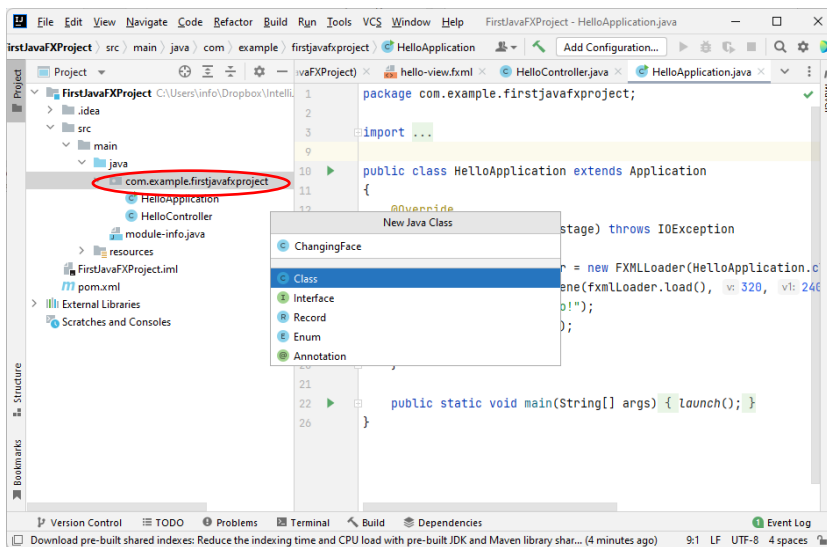


Figure 22

Figure 23 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;
```

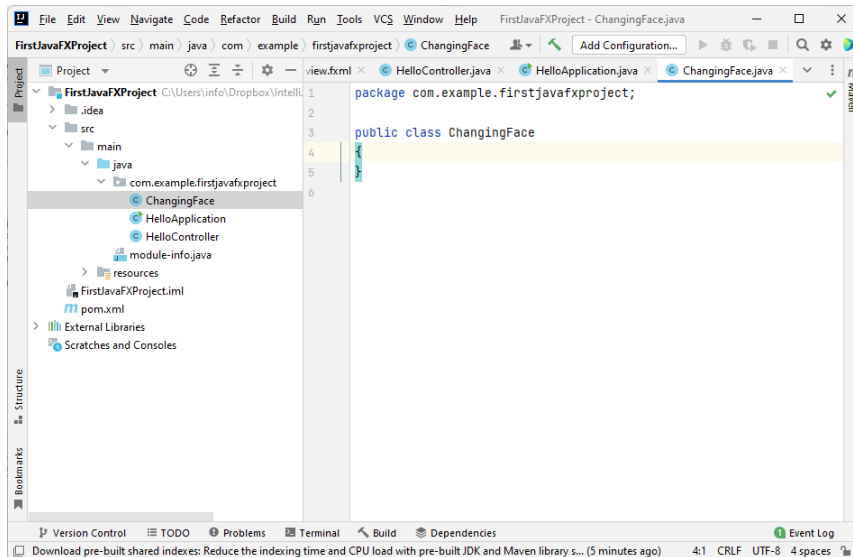


Figure 23

At this stage you can, if you wish rename your project. Right-click on the folder as shown in figure 24 and choose **Refactor | Rename**.

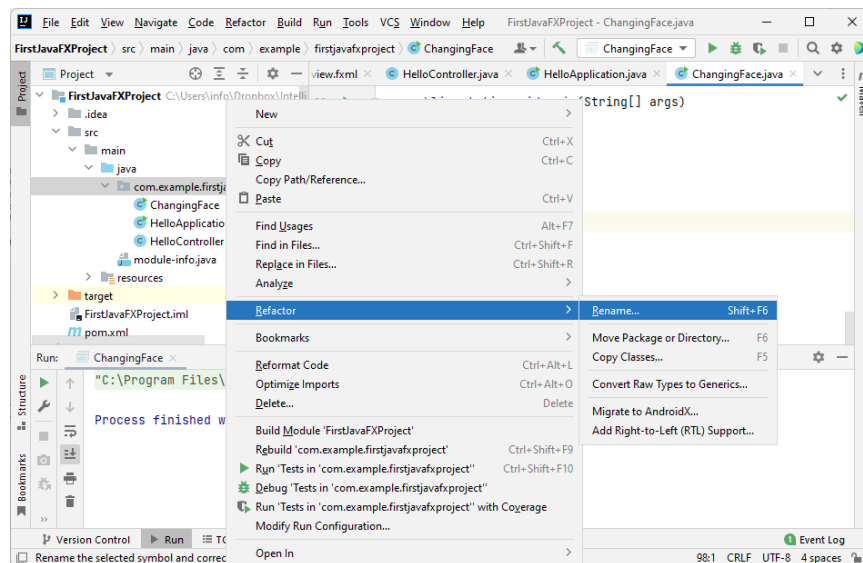


Figure 24

You will be presented with the dialogue shown in figure 25. It is recommended that you choose in *Whole Project*, which will simplify the directory structure.

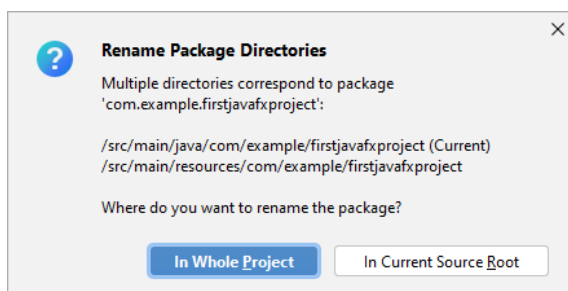


Figure 25

You will now be able to choose a new name for your package (figure 26).

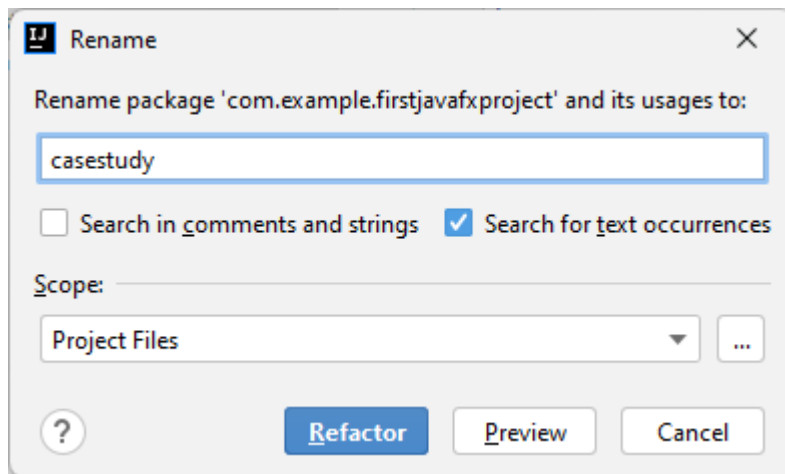


Figure 26

Choosing a name such as *casestudy* without specifying a directory structure (as we have done here) will cause the package to be relocated as shown in figure 27. As you can see the package instruction in the class has also been changed.

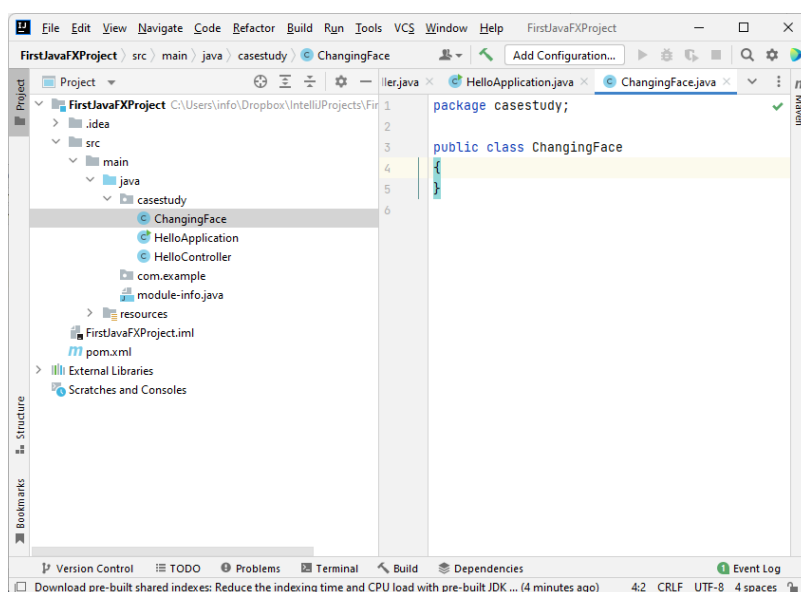


Figure 27

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.