Installing and Using Apache NetBeans®

1. Installing Apache NetBeans

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

https://netbeans.apache.org/download/index.html

Once you have installed both the JDK and NetBeans 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 NetBeans

2.1 Configuring NetBeans

Apache NetBeans comes preconfigured with various settings that can be customised if required. For example, to follow the curly bracket style of the text (with each bracket on a new line) follow these instructions:

- Navigate to **Tools > Options**
- Navigate to **Editor > Formatting** and set the **Braces** options as shown in Figure 1

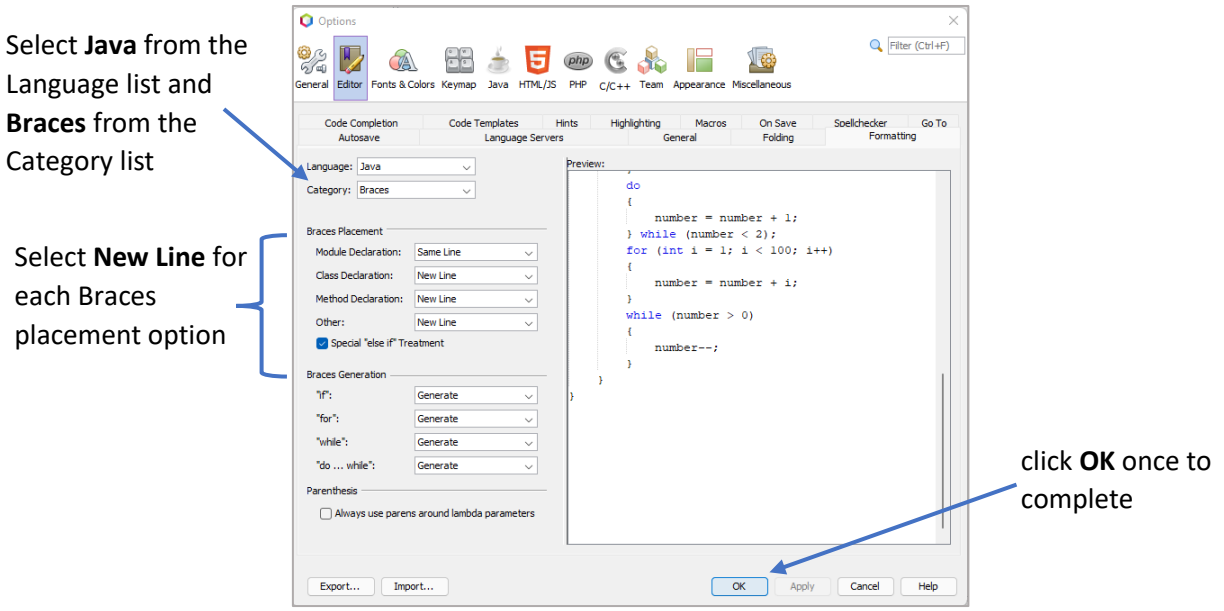

Figure 1

2.2 Starting a new project

Each project in NetBeans 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, choose **New Project** as shown in Figure 2.

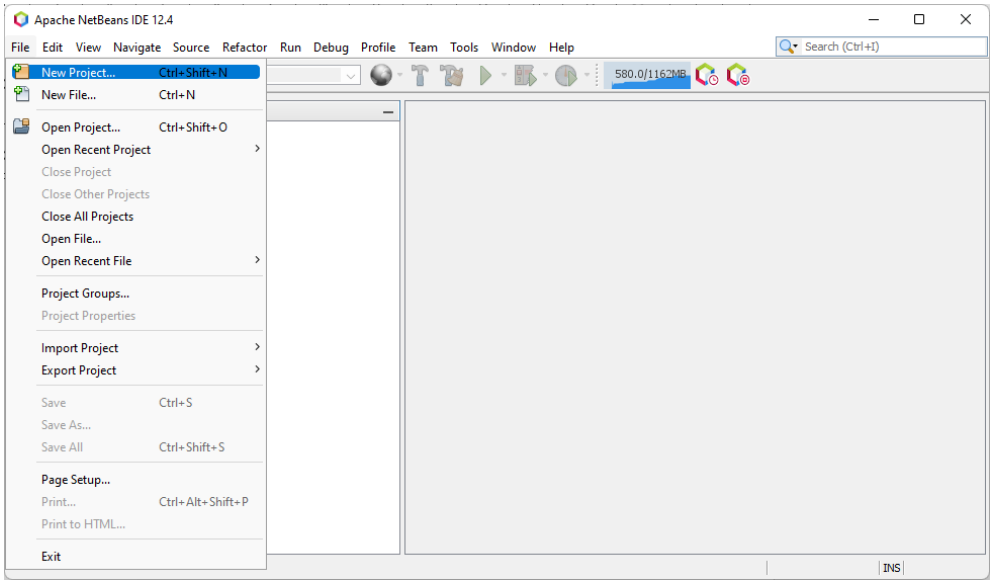

Figure 2

You will now see the screen shown in Figure 3. From the *Categories* window, select *Java with Ant*, and then choose *Java Application* from the *Projects* window. Then press *Next*. Leave the *Filter* field blank.

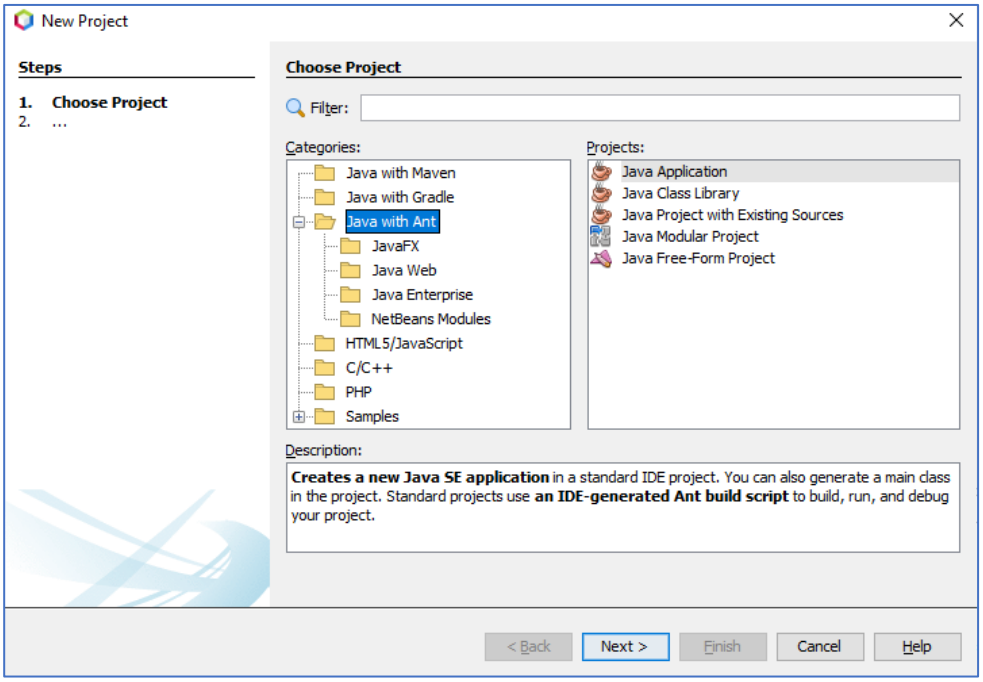

Figure 3

You will now see the screen shown in Figure 4.

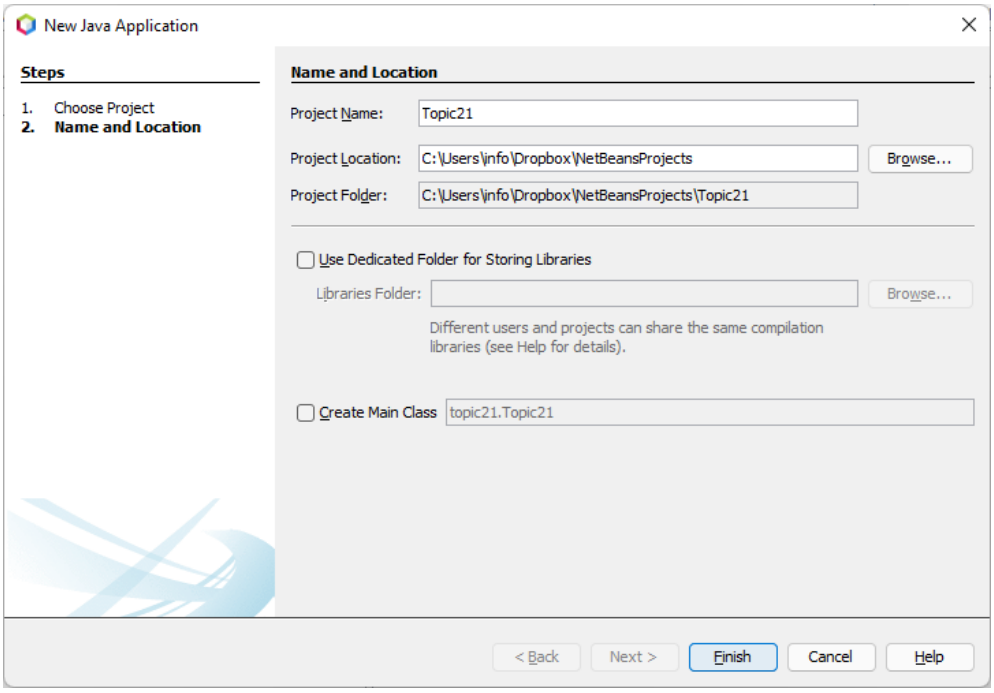

Figure 4

Enter the name of the project (in the above example *Topic21*) in the *Project Name* field.

In your early programs you are not going to create packages for your applications, so you want all your source files to be kept in what NetBeans calls a **default package**. The easiest way to achieve this is to uncheck the *Create Main Class* box, and create your main class later. Now you can press *Finish*. Your screen will now appear as in Figure 5, with *Topic1* listed as your only open project.

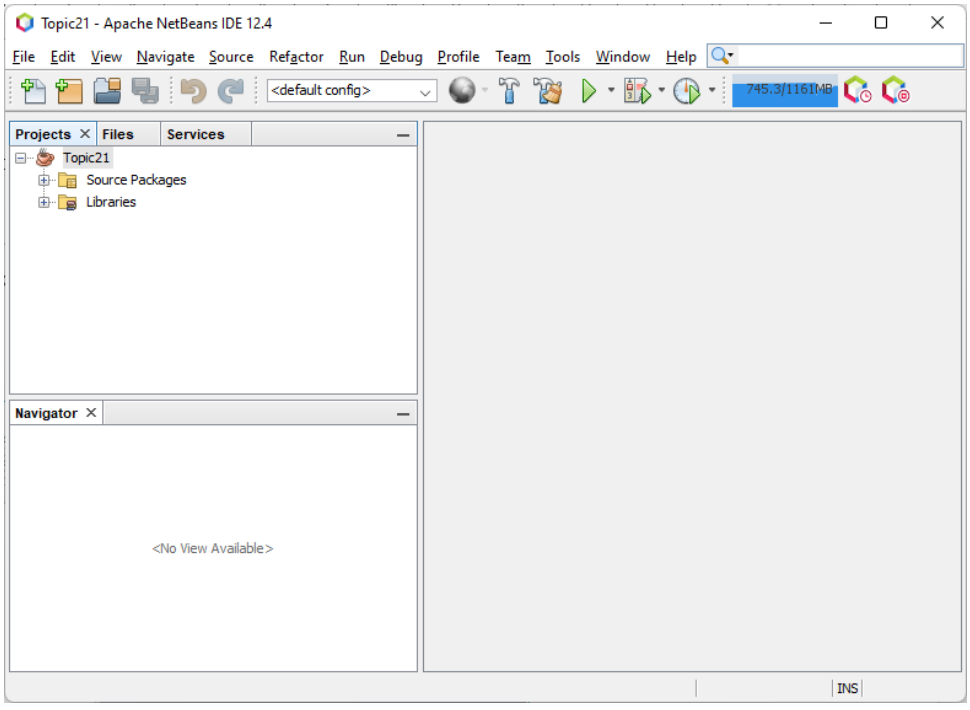

Figure 5

2.3 Adding new files to the project

To add a new file to your project, you can either choose *New File* from the *File* menu, press the + sign in the top left hand corner, or type Ctrl+N. You will see the screen shown in Figure 6.

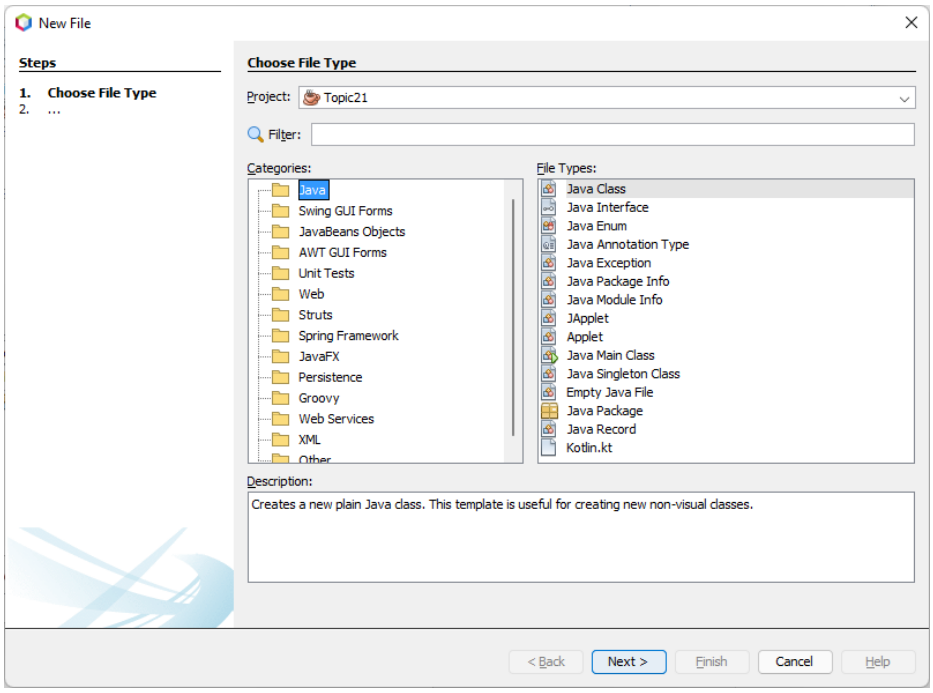

Figure 6

Choose *Java* from the categories section, and from the *File Types* window you can choose *Java Class* (provides just the class header, and opening and closing braces), *Java Main Class* (provides an outline main method) or *Empty Java File*, depending on which you want.

Pressing *Next* will cause a new screen to appear, as in Figure 7.

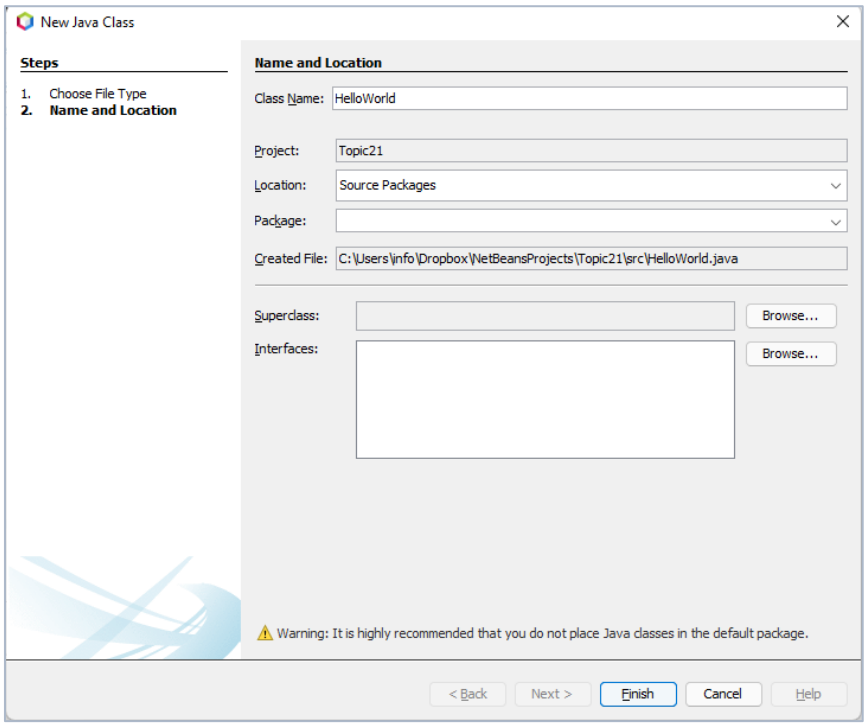

Figure 7

Type in the name that you wish to give to your class (*HelloWorld* in this example), and press *Finish*. The class will be listed in the default package, and (if you had chosen *Java Main Class*) the code will appear as shown in Figure 8. If you had chosen an empty class, this would be blank.

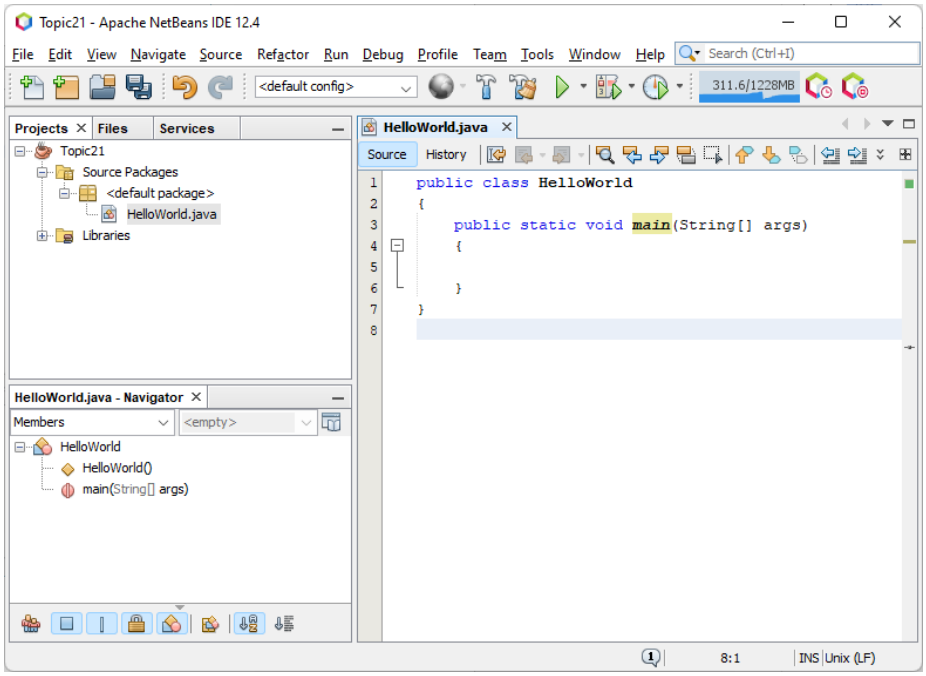

Figure 8

2.4 Compiling and running applications

Classes are automatically compiled as soon as they are written. Error messages are indicated in red and are revealed when you hover over the red icon with the mouse (as shown in Figure 9).

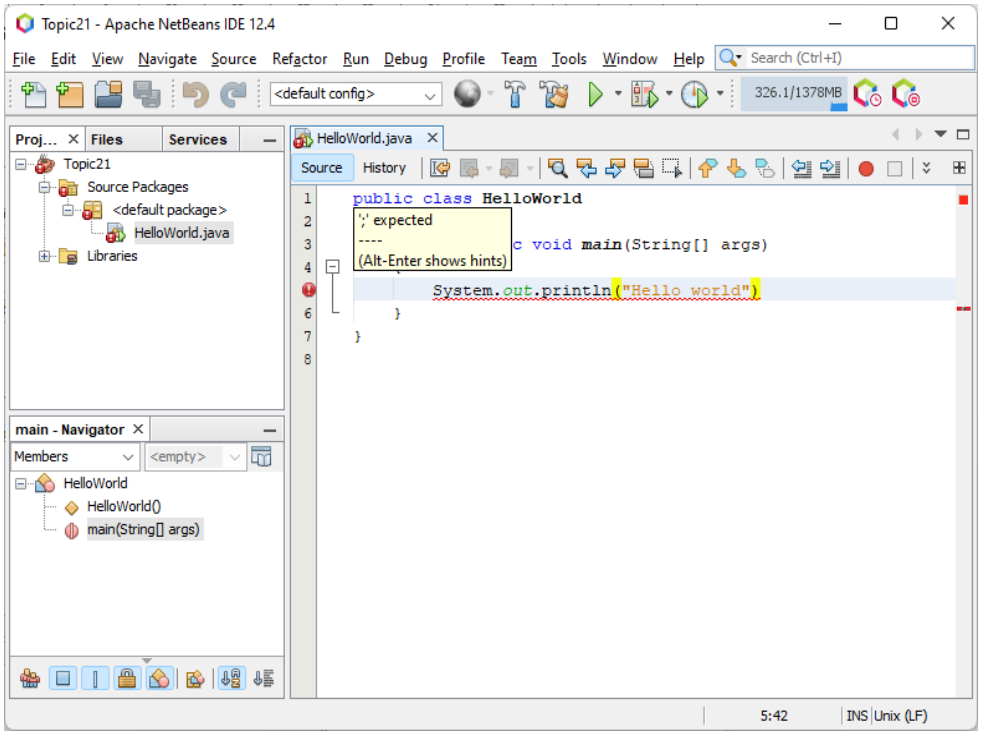

Figure 9

Programs can be run by pressing shift+F6 on the active window containing the main class; alternatively you can right-click on the class in the list and choose *Run File* from the drop down menu (or again press shift+F6).

Running a non-graphics program will cause the output to be displayed in an output window as shown in Figure 10. This window can be made to float, if you prefer (this is most easily done simply by dragging it).

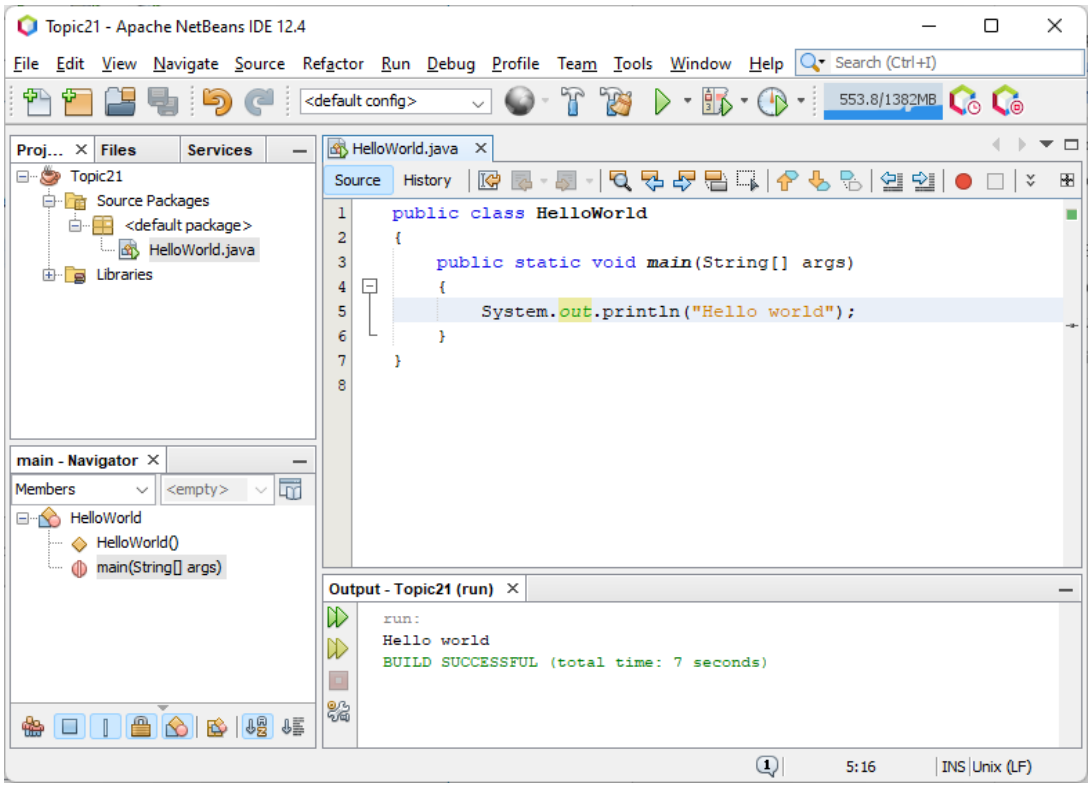

Figure 10

If you have existing source code files (.java) you can simply copy them into the *<default package>* folder while still in the NetBeans environment.

3. Using Apache NetBeans with JavaFX

1 Preparing NetBeans (this needs to be done only once)

Make sure that the latest JDK is installed.

Download the latest version of the JavaFX SDK. This comes in the form of a zipped file, and can be downloaded from the link below.

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

Extract the files to a location of your choice. Here we will assume it is 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 this folder there will be a folder called *lib* which contains the required .jar files. There will also be a folder called *bin* and a folder called *legal*.

In NetBeans choose **tools > libraries.** You will see the screen shown in Figure 11:

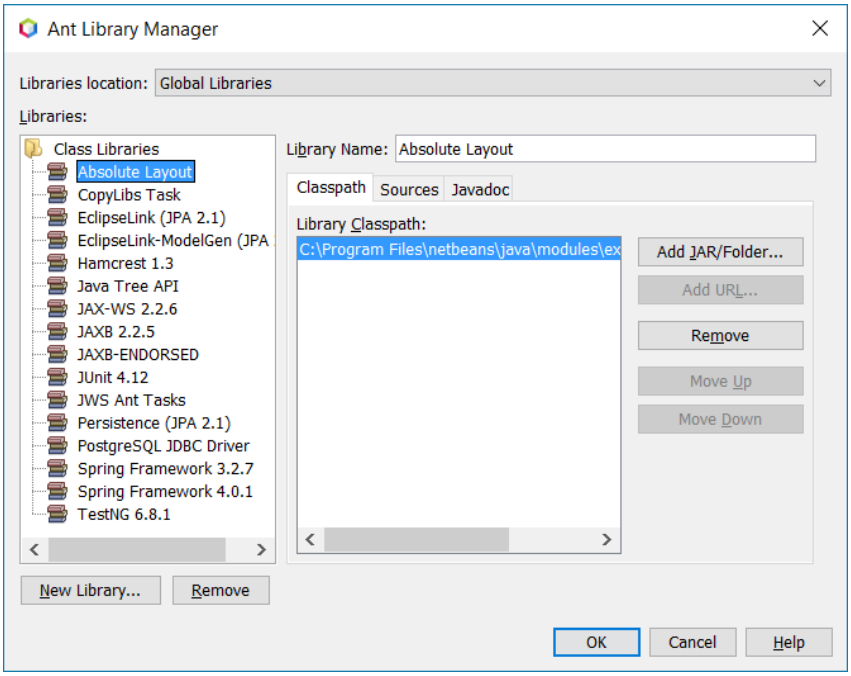

Figure 11

Choose **New Library** (Figure 12). Name the new library JavaFX (or a name of your choice), then choose Add JAR/Folder. 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. Finally click *OK*.

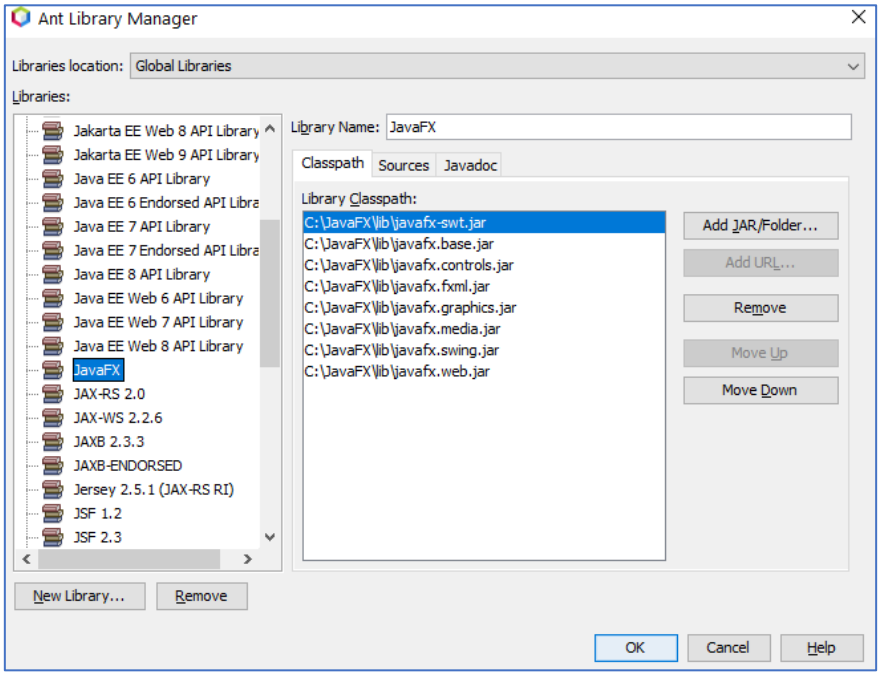

Figure 12

2 Starting a new JavaFX project (this needs to be done for each new JavaFX project)

Create a new Java application (do not choose JavaFX application).

Right-click on the project and choose *Properties*. Then select *Libraries*.

Click on the + sign next to *Modulepath* and choose *Add Library*. Add the JavaFX library that you created in step 1.

Click on the + sign next to *Classpath* and choose *Add JAR/Folder*. Add the .jar files in the same way as you did earlier (see Figure 13).

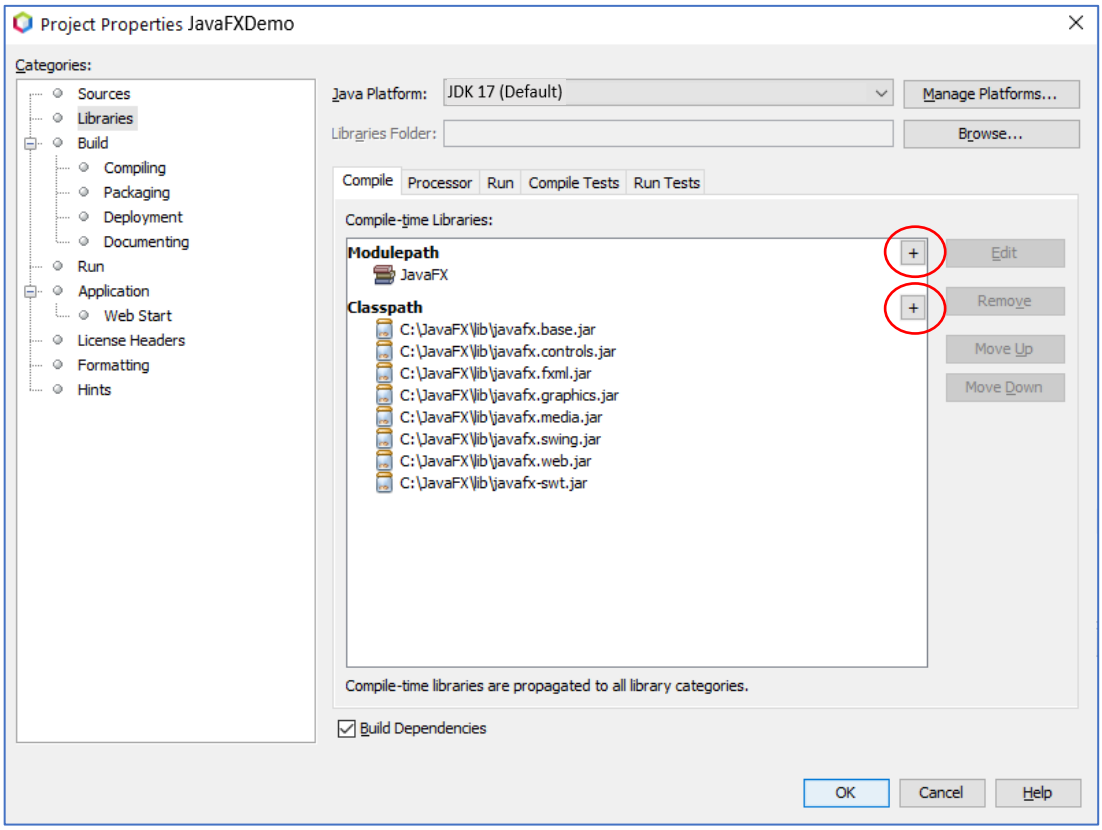

Figure 13

You can now go ahead and create a new JavaFX application. Add your new file 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 NetBeans does not add all the modules to the module path. To fix this proceed as follows:

In the same *Properties* dialogue as above, choose *Run* from the menu on the left. Add the following line to the VM options, replacing the directory (in red) with your own location for the JavaFX files if needed (see Figure 14) - 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
```

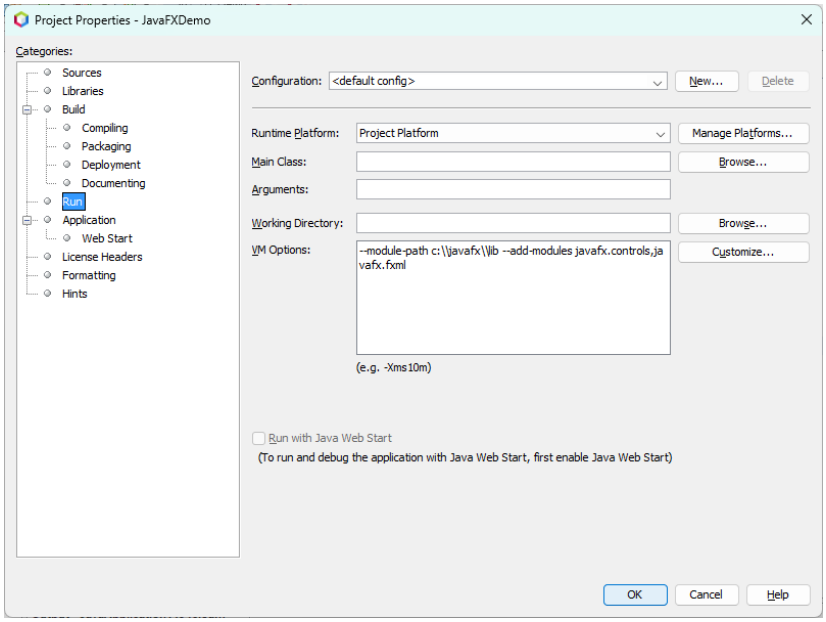
Project Properties - JavaFXDemo
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Configuration: <default config>
New...
Delete
Runtime Platform: Project Platform
Manage Platforms...
Main Class:
Browse...
Arguments:
Working Directory:
Browse...
VM Options:
--module-path c:\\javafx\\lib --add-modules javafx.controls,javafx.fxml
Customize...
(e.g. -Xms10m)
Run with Java Web Start
(To run and debug the application with Java Web Start, first enable Java Web Start)
OK
Cancel
Help

Figure 14