



Imperative and Object Oriented Programming

NetBeans Tutorial

Charlie Abela
Department of Artificial Intelligence
charlie.abela@um.edu.mt

Introduction to NetBeans

The NetBeans IDE is open source and is written in the Java programming language. It provides the services common to creating desktop applications -- such as window and menu management, settings storage -- and is also the first IDE to fully support JDK 5.0 features. The NetBeans platform and IDE are free for commercial and non-commercial use, and they are supported by Sun Microsystems. It can be downloaded from <http://www.netbeans.org/>

1. Using NetBeans

To be able to successfully build programs it is recommended to follow the intended procedure, described here.

First it's important to create a new project. Various project types are available; however in this case the intended type will be *Java Application*.

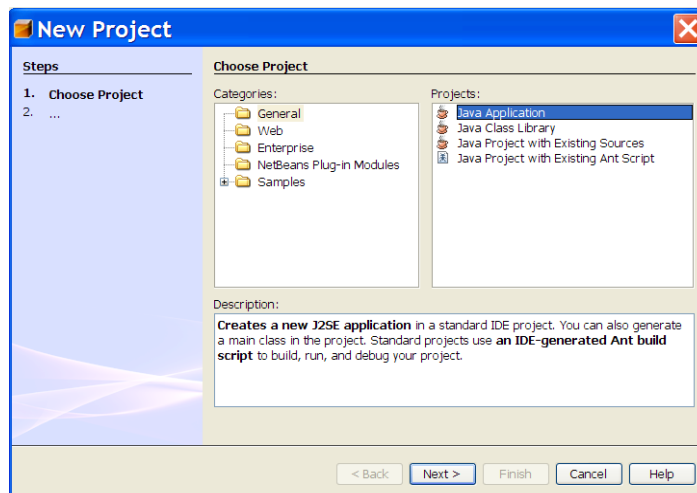


Figure 1: Choosing a Project type

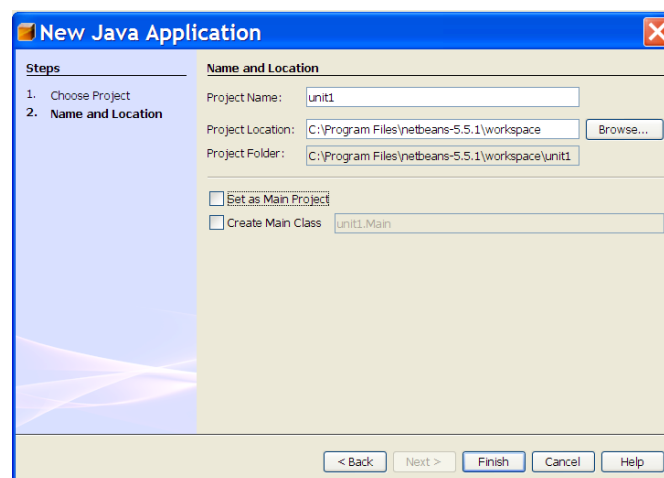


Figure 2: Setting up the project

It is recommended that you change the name of the project to something of this sort: *unit_1*. Usually it is also a good idea to set up the location in which to store the projects and future work in a well known directory. Here the chosen location is: *netbeans_home/workspace*. To keep control over the way that classes (e.g. the main launching class) are defined it is recommended to un-tick the two check boxes displayed in Figure 2.

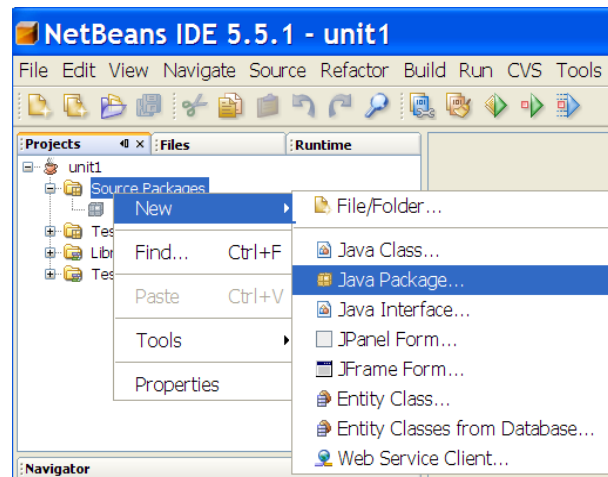


Figure 3: Creating a new package

After *Finish* is clicked the new project structure will be in place, however we now need to extend this by adding a new package. In this manner classes associated with the same topics can be stored in one package. The end result will be multiple packages in one project, each having a number of classes (programs). The process is simple, just rt-click on the *Source Package* label in NetBeans' *Projects* tab and click *New* → *Java Package*.

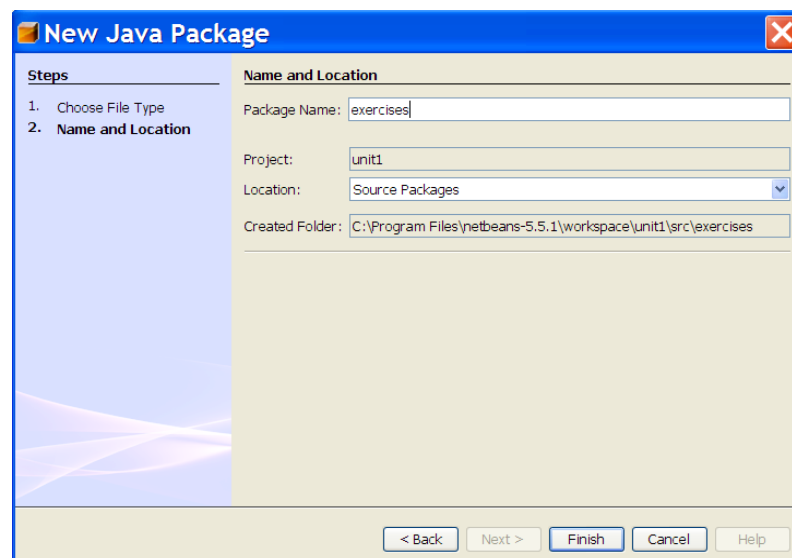


Figure 4: Setting up the package details

In this tutorial we will label the package exercises. Clicking *Finish* will terminate this process.

Next we will create a simple class example and store it within our newly created package.

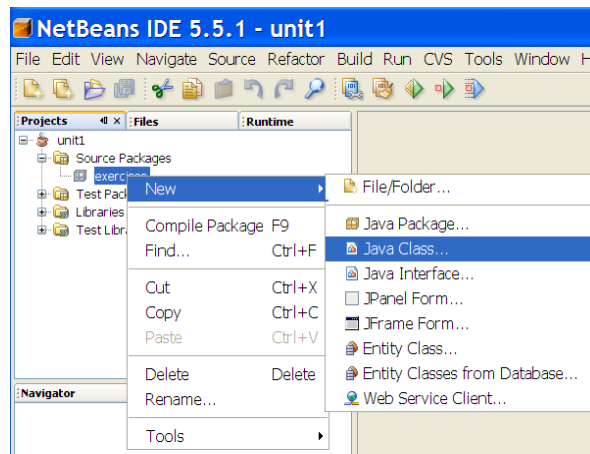


Figure 5: Create a new class

This process is very straight forward and involves a rt-click on the appropriate *exercises* package, after which we choose *Java Class*.

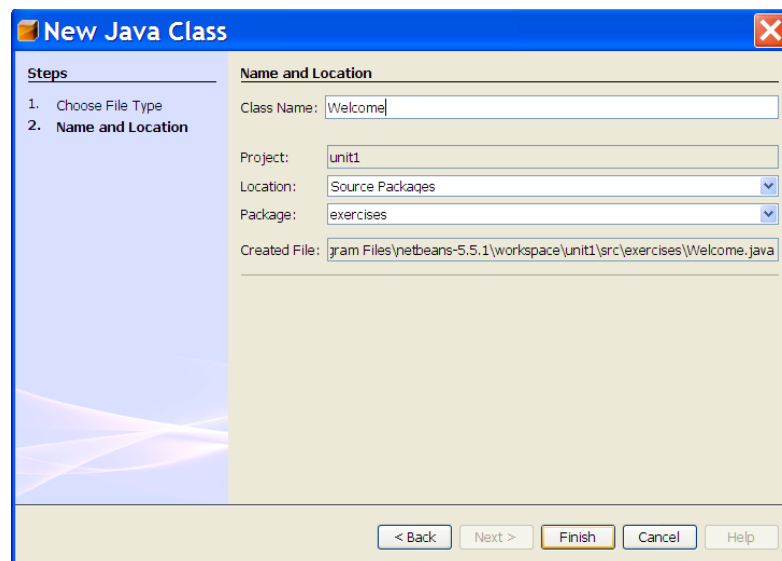


Figure 6: Adding details for the Welcome class

Set the class name to *Welcome*. The other details can be checked to verify that all is as intended.

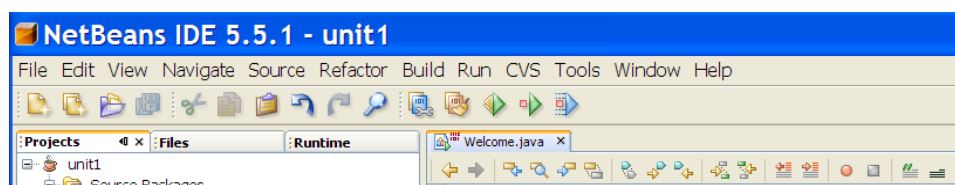


Figure 7: NetBeans toolbars

The new class structure will be generated and at this point we need to edit the code and add the necessary lines. Note that the first line of code is referring to the package name in which this class will be stored.

The code is missing a main method (since we un-checked the checkbox previously), which we add. Within this method we also need to add the line:

```
System.out.println("Welcome to Java");
```

Notice that while adding this line of code NetBeans' editor will be suggesting stuff to make our lives easier. The next step is to either save, compile and execute our program, or else execute it directly (which will automatically save and compile).

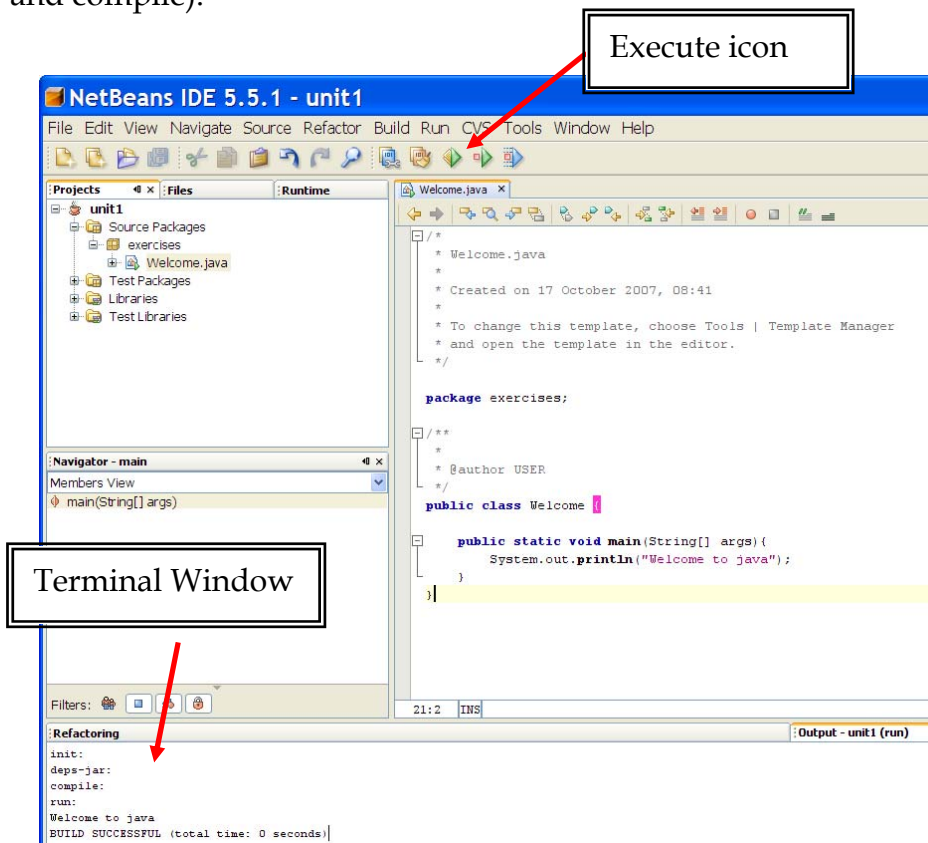


Figure 8: Running the Welcome class

Any output from the executed program will be displayed in the terminal window. In the above example the program executed correctly and the intended message "Welcome to Java" displayed. If some compile time and runtime errors were encountered, the appropriate exception message would have also been displayed in this window.