Getting Started

One of the difficult things about getting started with Java is installing everything you need. Even before you write a single line of code, the headaches begin! Hopefully, the following sections will make life easier for you.

We're going to write all our code using a free piece of software called NetBeans. This is one of the most popular IDEs (Interface Development Environment) in the world for writing Java programmes. You'll see what it looks like shortly. But before NetBeans will work, it needs you to install the necessary Java components and files. First up is something called the Java Virtual Machine.

The Java Virtual Machine

Java is platform independent. This means that it will run on just about any operating system. So whether your computer runs Windows, Linux, Mac OS, it's all the same to Java! The reason it can run on any operating system is because of the Java Virtual Machine. The Virtual Machine is a programme that processes all your code correctly. So you need to install this programme (Virtual Machine) before you can run any Java code.

Java is owned by a company called Sun Microsystems, so you need to head over to Sun's website to get the Java Virtual Machine, also known as the Java Runtime Environment (JRE). Try this page first:

http://java.com/en/download/index.jsp

You can check to see if you already have the JRE on your computer by clicking the link "Do I have Java?". You'll find this link under the big Download button at the top of the page. (Unless Sun have changed things around, again!) When you click the link, your computer will be scanned for the JRE. You will then be told whether you have it or not. If not, you'll be given the opportunity to download and install it.

Or you could just head over to this page:

http://java.com/en/download/manual.jsp

The "manual" in the above links means "manual download". The page gives you download links and instructions for a wide variety of operating systems.

After downloading and installing, you may need to restart you computer. When you do, you will have the Java Virtual Machine.

The Java Software Development Kit

At this stage, you still can't write any programmes. The only thing you've done is to install software so that Java programmes can be run on your computer. To write code and test it out, you need something called a Software Development kit.

Java's Software Development Kit can currently be downloaded from here:

http://developers.sun.com/downloads/

The one we're going to be using is called Java SE. (The SE stands for Standard Edition.). Click on that link, then on "Java SE (JDK) 6 Download". You'll then find yourself on a page with a bewildering list of options to download. Because we're going to be using NetBeans, locate this:

JDK 6 Update *number* **with NetBeans 6.x**

Click the Download link to be taken to yet another page. Click the top download to be taken to a page that asks you to select your operating system. Click Continue to finally get the download you need. A word of warning, though – this download will be big, at over a 130 megabytes at the time of writing! Once you've downloaded the JDK and NetBeans, install it on your computer.

We're going to be using NetBeans to write our code. Before launching the software, however, here's how things work in the world of Java.

Java – How things work

You write the actual code for your programmes in a text editor. (In NetBeans, there's a special area for you to write code.) The code is called source code, and is saved with the file extension **.java**. A programme called **Javac** is then used to turn the source code into Java Byte Code. This is known as **compiling**. After Javac has finished compiling the Java Byte Code, it creates a new file with the extension **.class**. (At least, it does if no errors are detected.) Once the **class** file has been created, it can be run on the Java Virtual Machine. So:

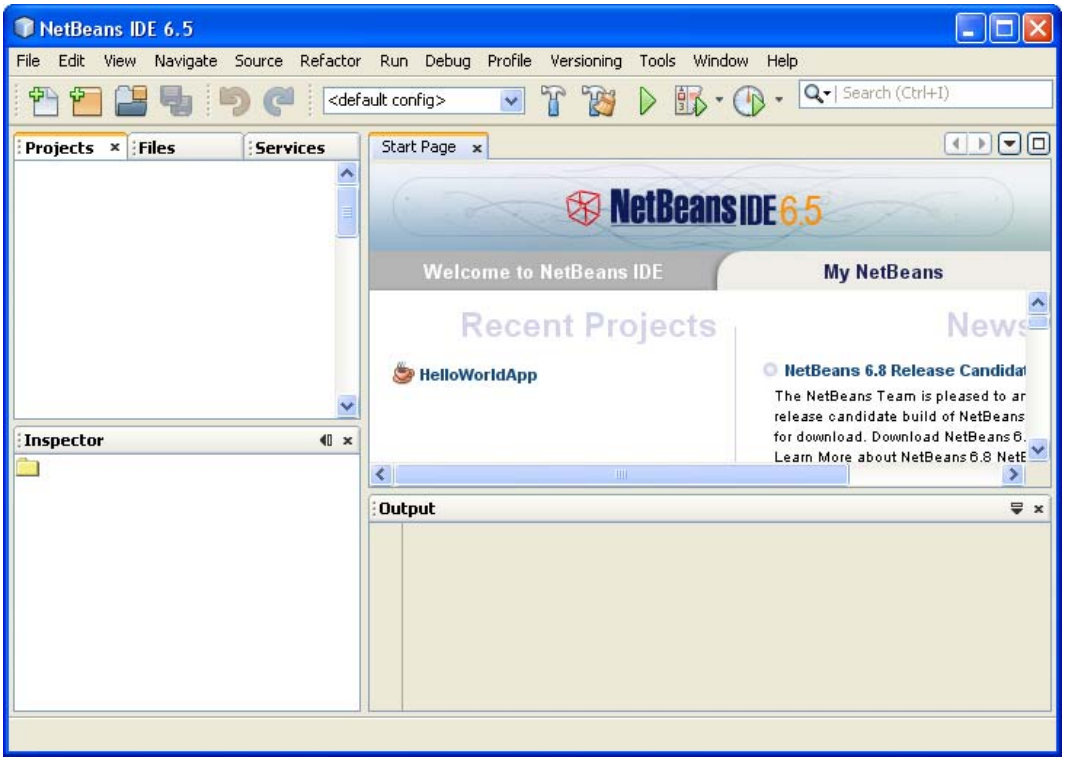
- Create source code with the extension .java
- Use Javac to create (compile) a file ending in .class
- Run the compiled class

NetBeans handles all the creating and compiling for you. Behind the scenes, though, it takes your sources code and creates the java file. It will launch Javac and compile the class file. NetBeans can then run your programme inside its own software. This saves you the hassle of opening up a terminal window and typing long strings of commands,

Now that you have a general idea of how Java works, launch your NetBeans software.

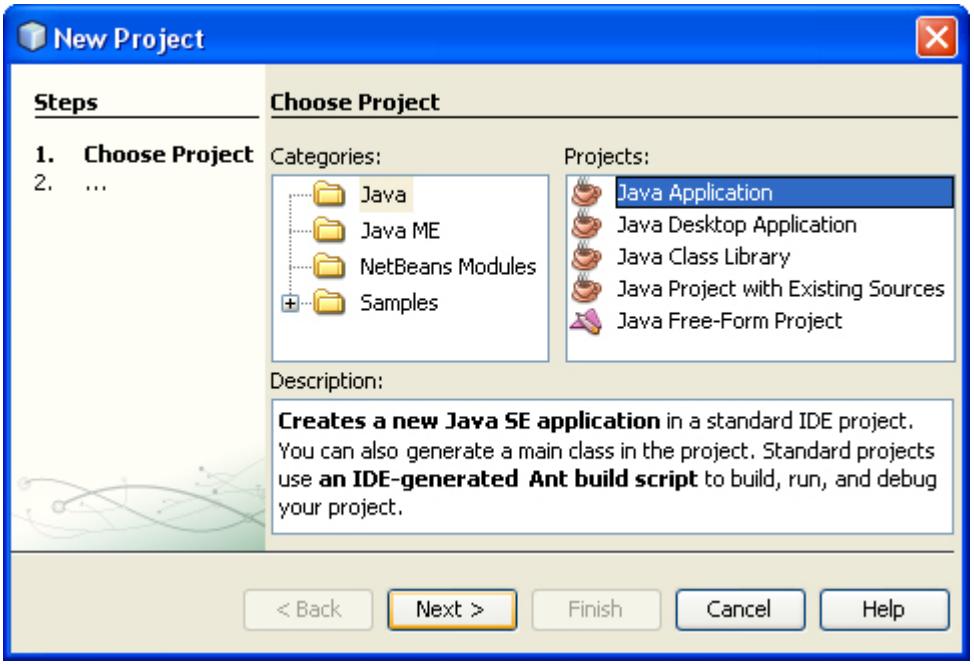
The NetBeans Software

When you first run NetBeans, you'll see a screen something like this one:

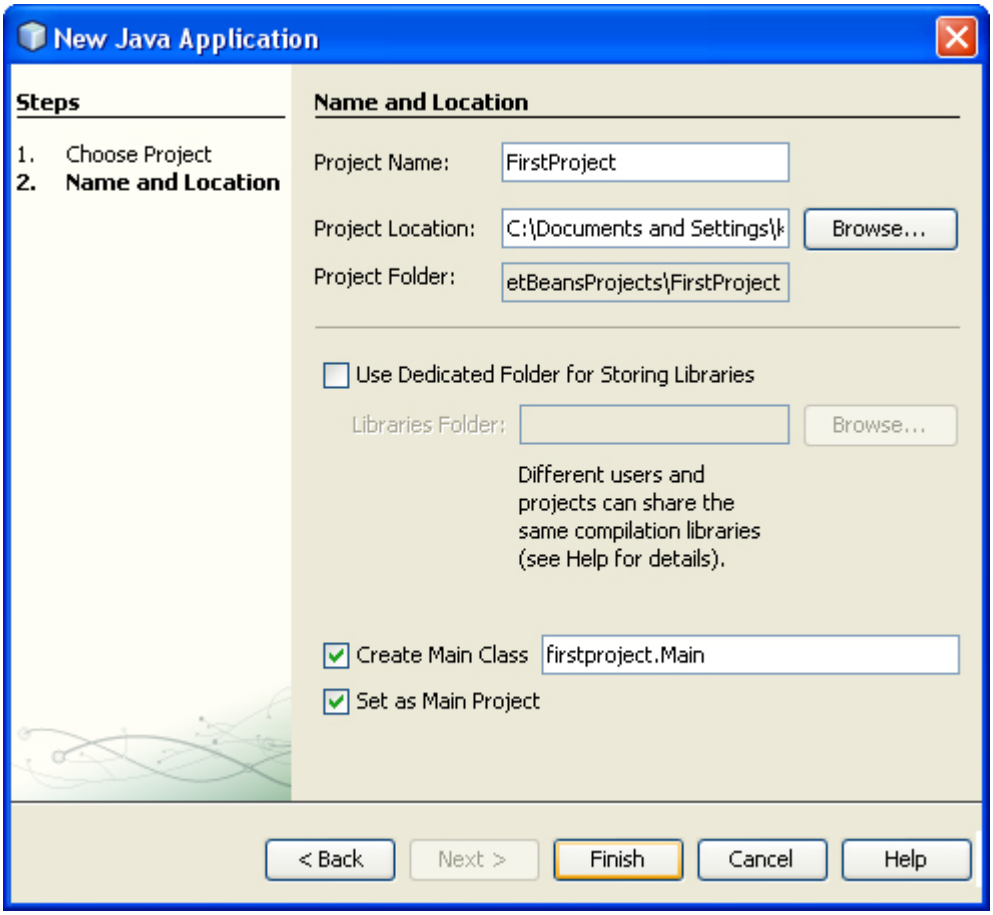


You may have to drum your fingers and wait a while, as it's not the fastest thing in the world.

To start a new project, click on **File > New Project**. You'll see the following dialogue box appear:



We're going to be creating a Java Application, so select **Java** under **Categories**, and then **Java Application** under **Projects**. Click the **Next** button at the bottom to go to step two:



In the Project Name area at the top, type a Name for your Project. Notice how the text at the bottom changes to match your project name (in the text box to the right of **Create Main Class**):

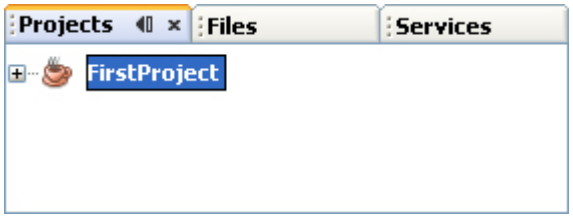
firstproject.Main

If we leave it like that, the Class will have the name **Main**. Change it to FirstProject:

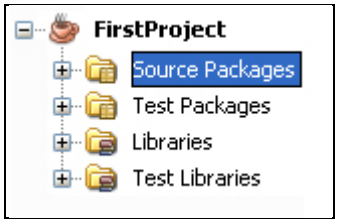
Now, the Class created will be called FirstProject, with a capital “F”, capital “P”. The package is also called firstproject, but with a lowercase “f” and lowercase “j”.

The default location to save your projects appears in the Project Location text box. You can change this, if you prefer. NetBeans will also create a folder with your project name, in the same location. Click the **Finish** button and NetBeans will go to work creating all the necessary files for you.

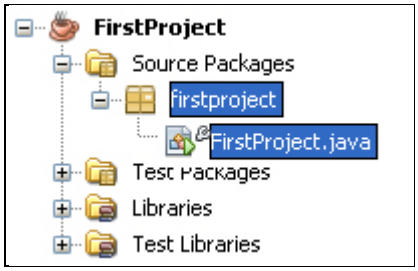
When NetBeans returns you to the IDE, have a look at the Projects area in the top left of the screen (if you can't see this, click **Window > Projects** from the menu bar at the top of the software):



Click the plus symbol to expand your project, and you'll see the following:

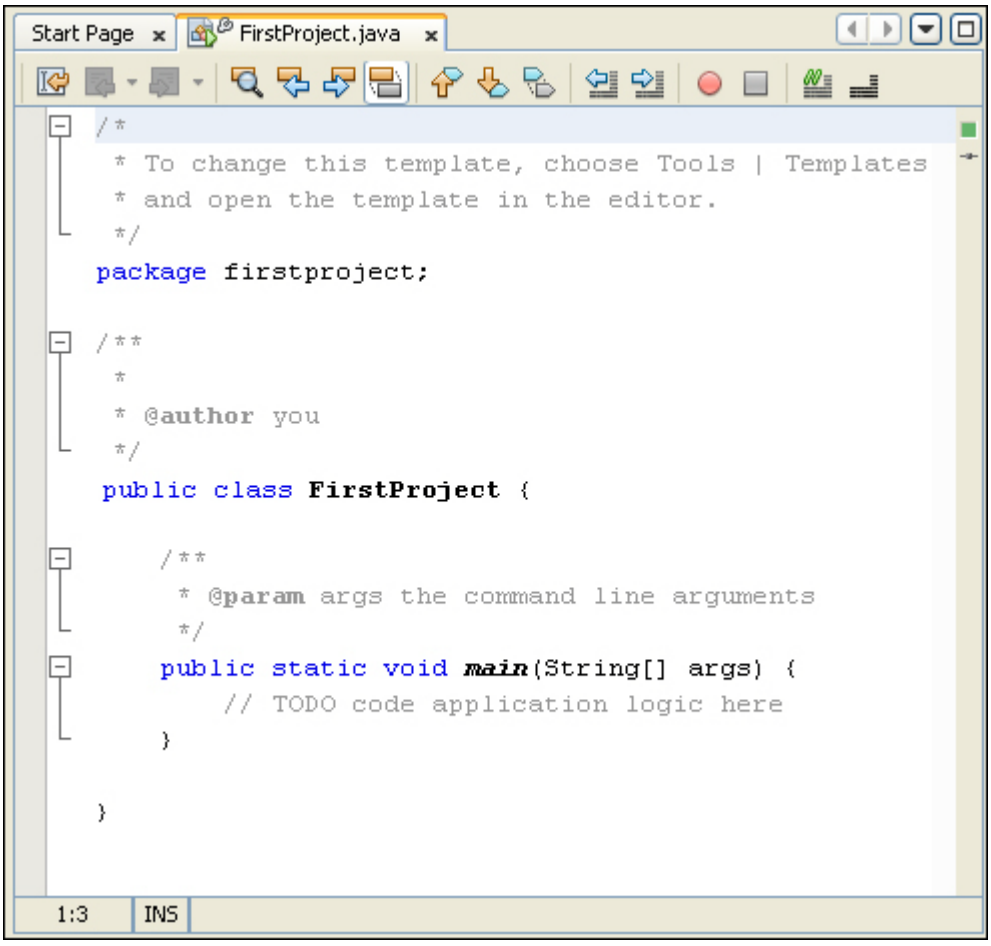


Now expand Source Packages to see your project name again. Expand this and you'll see the Java file that is your source code.



This same source code should be displayed to the right, in the large text area. It will be called **FirstProject.java**. If you can't see a code window, simply double click **FirstProject.java** in your Projects window above. The code will appear, ready for you to start work.

The coding window that appears should look like this (we've changed the author's name):



One thing to note here is that the class is called FirstProject:

```
public class FirstProject {
```

This is the same name as the java source file in the project window: **FirstProject.java**. When you run your programmes, the compiler demands that the source file and the class name match. So if your **.java** file is called firstProject

but the class is called FirstProject then you'll get an error on compile. And all because the first one is lowercase "f" and the second one uppercase.

Note that although we've also called the package **firsproject**, this is not necessary. We could have called the package **someprogramme**. So the name of the package doesn't have to be the same as the java source file, or the class in the source file: it's just the name of the java source file and the name of the class that must match.

Comments

In the image above, you'll notice that some text is greyed out, with lots of slashes and asterisks. These are comments. When the programme runs, comments are ignored. So you can type whatever you want inside of your comments. But it's usual to have comments that explain what it is you're trying to do. You can have a single line comment by typing two slashes, followed by your comment:

```
//This is a single line comment
```

If you want to have more than one line, you can either do this:

```
//This is a comment spreading
//over two lines or more
```

Or you can do this:

```
/*
        This is a comment spreading
        over two lines or more
*/
```

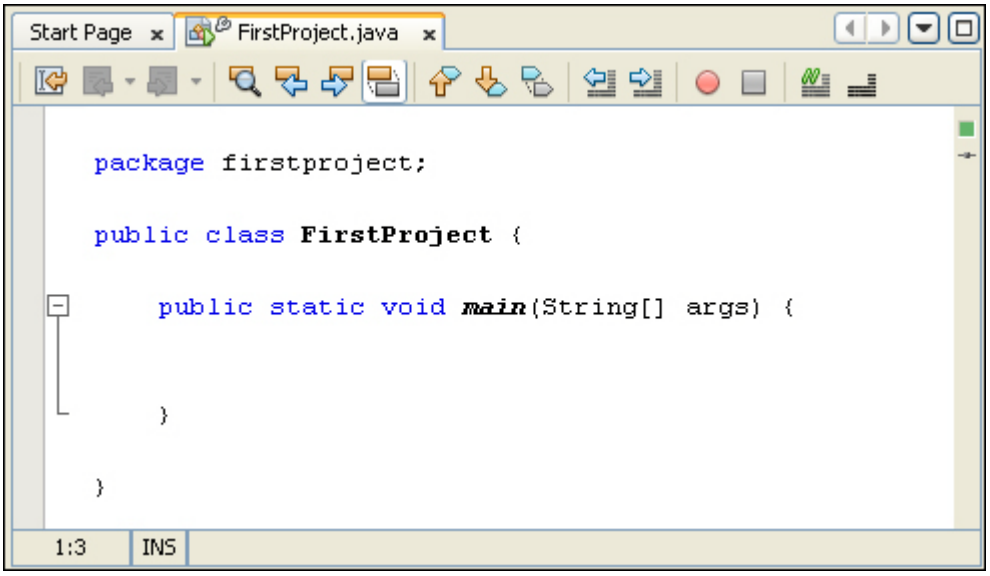
In the comment above, note how it starts with /*. To end the comment, we have */ instead.

There's also something called a Javadoc comment. You can see two of these in the coding image on the previous page. A Javadoc comment starts with a single forward slash and two asterisks (/**) and ends with an asterisk and one slash (*/). Each line of the comment starts with one asterisk:

```
/**
        *This is a Javadoc comment
*/
```

Javadoc comments are used to document code. The documented code can then be turned into an HTML page that will be helpful to others. You can see what these look like by clicking **Run** from the menu at the top of NetBeans. From the Run menu, select **Generate Javadoc**. There's not much to see, however, as you haven't written any code yet!

At this stage of your programming career, you can delete the comments that NetBeans generates for you. Here's our code again with the comments deleted:



Structure of the programme

Now it looks a lot cleaner! You can see we have the package name first. Notice how the line ends with a semicolon. If you miss the semicolon out, the programme won't compile:

```
package firstproject;
```

The class name comes next:

```
public class FirstProject {

}
```

You can think of a class as a code segment. But you have to tell Java where code segments start and end. You do this with curly brackets. The start of a code segment is done with a left curly bracket **{** and is ended with a right curly bracket **}**. Anything inside of the left and right curly brackets belong to that code segment.

What's inside of the left and right curly brackets for the class is another code segment. This one:

```
public static void main( String[ ] args ) {


}
```

The word “main” is the important part here. Whenever a Java programme starts, it looks for a method called **main**. (A method is just a chunk of code. You’ll learn more about these later.) It then executes any code within the curly brackets for **main**. You’ll get error messages if you don’t have a main method in your Java programmes. But as its name suggest, it is the **main** entry point for your programmes.

The blue parts before the word “main” can be ignored for now.

(If you’re curious, however, **public** means that the method can be seen outside of this class; **static** means that you don’t have to create a new object; and **void** means it doesn’t return a value – it just gets on with it. The parts between the round brackets of **main** are something called command line arguments. Don’t worry if you don’t understand any of that, though.)

The important point to remember is that we have a class called FirstProject. This class contains a method called **main**. The two have their own sets of curly brackets. But the **main** chunk of code belongs to the class **FirstProject**.

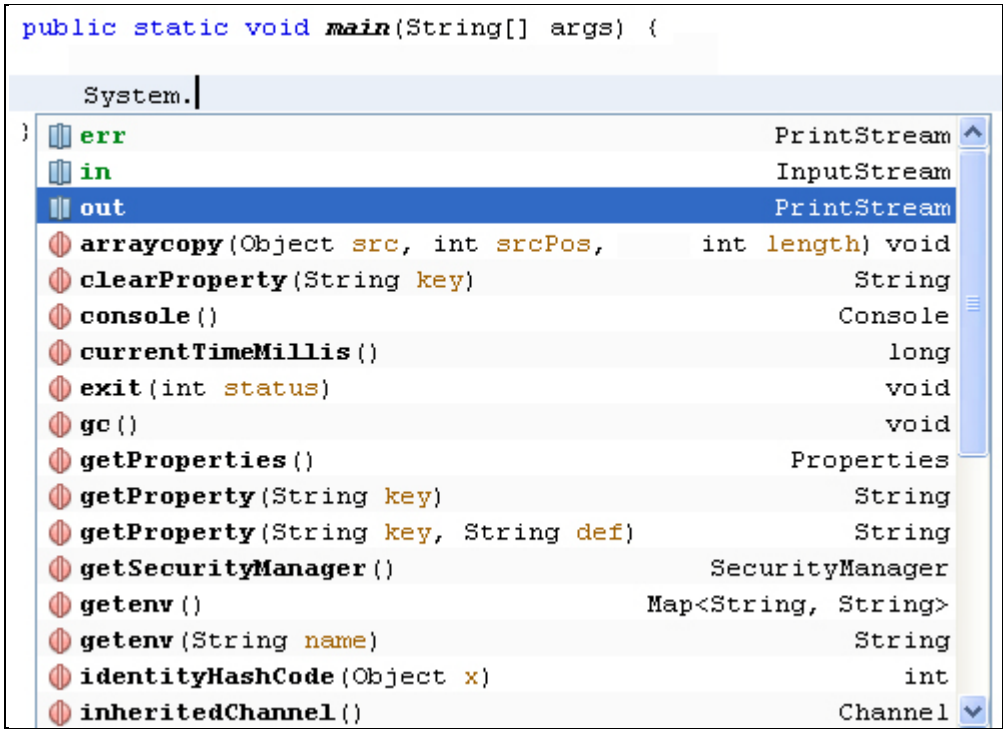
Running the code

You can run this code and turn it into a programme. It doesn’t do anything, but it will still compile. So let’s add one line of code just so that we can see how it works.

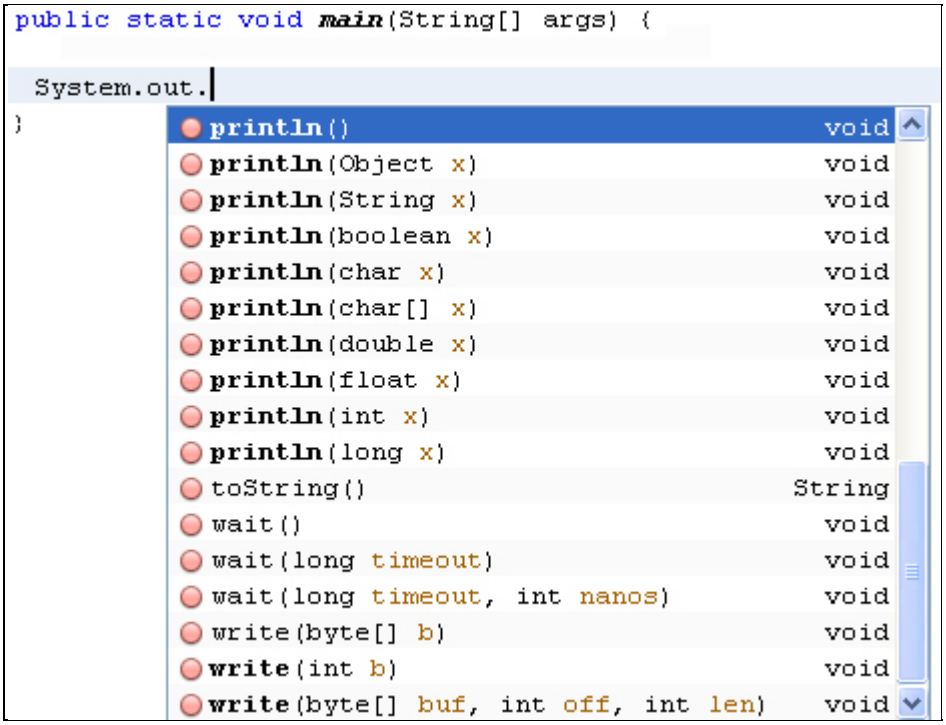
We’ll output some text to a console window. Add the following line to your **main** method:

```
public static void main(String[ ] args) {

        System.out.println("My First Project");

}
```

When you type the full stop after “System”, NetBeans will try to help you by displaying a list of available options:



Double click **out** to add it to your code, then type another full stop. Again, the list of options appears:



Select **println**( ). What this does is to print a line of text to the output screen. But you need to place your text between the round brackets of **println**. Your text needs to go between a pair of double quotes:

```
public static void main(String[] args) {

    System.out.println("");
}
```

Once you have your double quotes in place, type your text:

```
public static void main(String[] args) {

    System.out.println("My First Project");
}
```

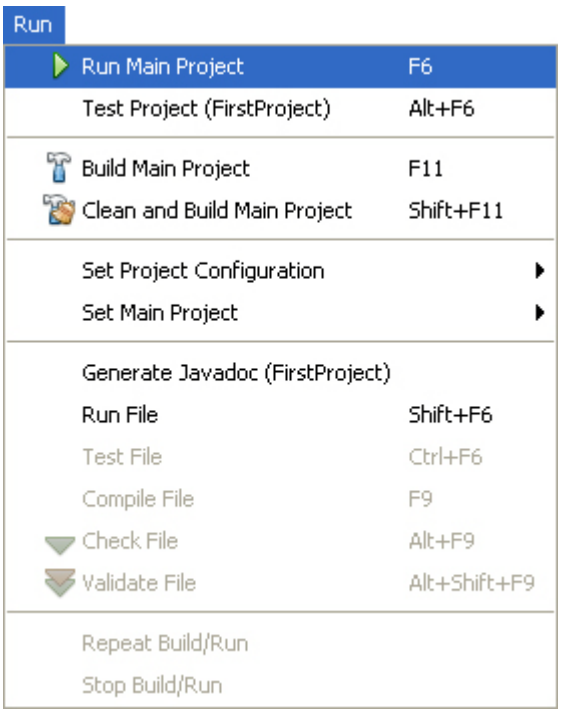
Notice that the line ends in a semicolon. Each complete line of code in Java needs a semicolon at the end. Miss it out and the programme won't compile.

OK, we can now go ahead and test this programme out. First, though, save your work. You can click **File > Save**, or **File > Save All**. Or click the **Save** icon on the NetBeans toolbar.

Running programmes

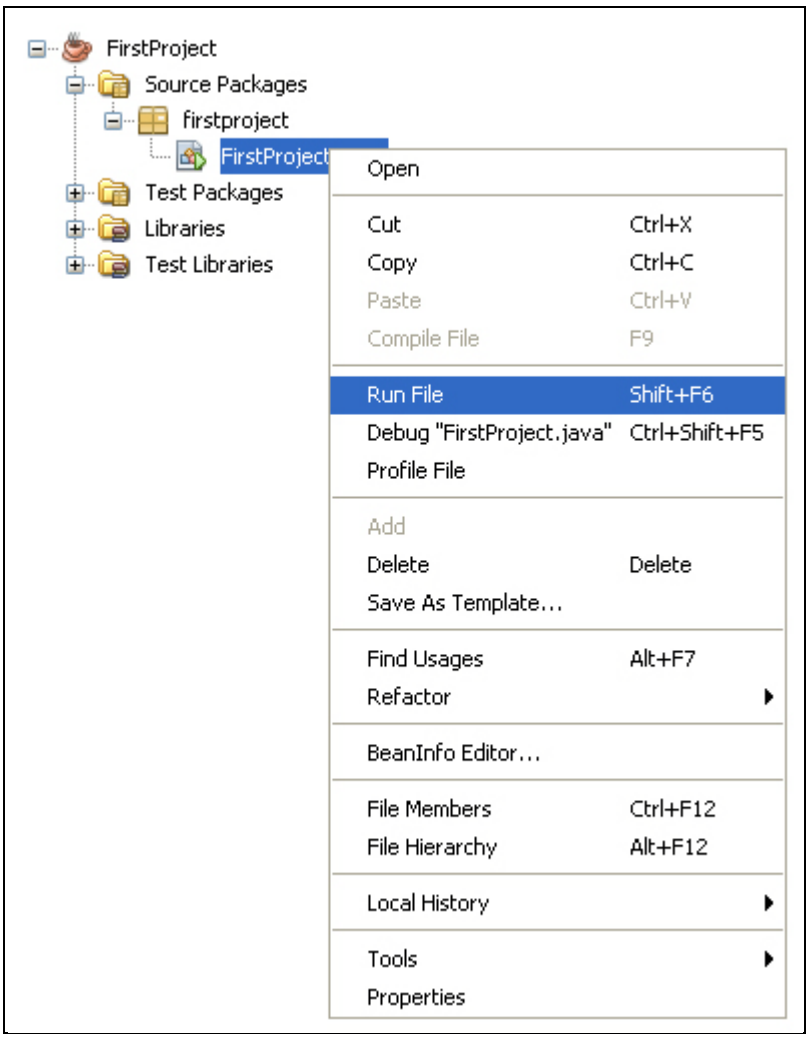
When you run a programme in NetBeans, it will run in the Output window at the bottom of your screen, just underneath your code. This is so that you don't have to start a terminal or console window – the Output window IS the console.

There are various ways to run your programme in NetBeans. The easiest way is to press F6 on your Keyboard. You can also run programmes using the menus as the top of NetBeans. Locate the **Run** menu, then select **Run Main Programme**:

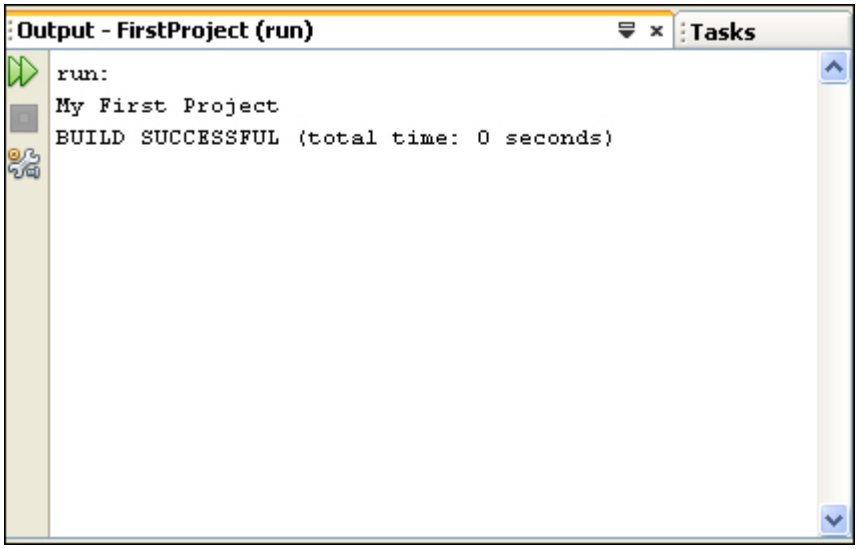


You can also click the green arrow on the NetBeans toolbar:

Another way to run your programmes is from the Projects window. This will ensure that the right source code is being run. Simply right click your java source file in the projects window and you'll see a menu appear. Select **Run File**.



Using one of the above methods, run your programme. You should see something happening in the Output window:

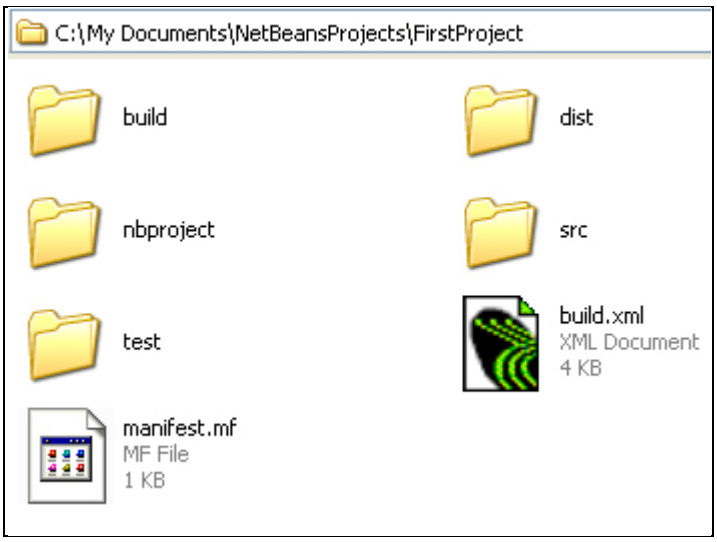


The second line in the Output window above is our code: My First Project. You can quickly run it again by clicking the two green arrows in the top left of the Output window.

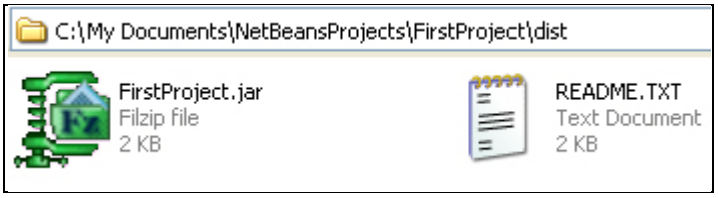
Sharing your programmes with others

You can send your programmes to other people so that they can run them. To do that, you need to create a JAR file (Java Archive). NetBeans can do all this for you. From the Run menu at the top, select **Clean and Build Main Project**.

When you do, NetBeans saves your work and then creates all the necessary files. It will create a folder called **dist** and place all the files in there. Have a look in the place where your NetBeans projects are and you'll see the dist folder:



Double click the **dist** folder to see what's inside of it:



You should see a JAR file and README text file. The text file contains instructions on how to run the programme from a terminal/console window.

Now that you know how to run your java source files, let's do some programming.