

Notes from Well House Consultants

These notes are written by Well House Consultants and distributed under their Open Training Notes License. If a copy of this license is not supplied at the end of these notes, please visit

*<http://www.wellho.net/net/whcotnl.html>
for details.*

1.1 Well House Consultants

Well House Consultants provides niche training, primarily but not exclusively in Open Source programming languages. We offer public courses at our training centre and private courses at your offices. We also make some of our training notes available under our "Open Training Notes" license, such as we're doing in this document here.

1.2 Open Training Notes License

With an "Open Training Notes License", for which we make no charge, you're allowed to print, use and distribute these notes provided that you retain the complete and unaltered license agreement with them, including our copyright statement. This means that you can learn from the notes, and have others learn from them too.

You are NOT allowed to charge (directly or indirectly) for the copying or distribution of these notes, nor are you allowed to charge for presentations making any use of them.

1.3 Courses presented by the author

If you would like us to attend a course (Java, Perl, Python, PHP, Tcl/Tk, MySQL or Linux) presented by the author of these notes, please see our public course schedule at

<http://www.wellho.net/course/index.html>

If you have a group of 4 or more trainees who require the same course at the same time, it will cost you less to have us run a private course for you. Please visit our onsite training page at

<http://www.wellho.net/course/otc.html>

which will give you details and costing information

1.4 Contact Details

Well House Consultants may be found online at

<http://www.wellho.net>

graham@wellho.net

technical contact

lisa@wellho.net

administration contact

Our full postal address is

404 The Spa

Melksham

Wiltshire

UK SN12 6QL

Phone +44 (0) 1225 708225

Fax +44 (0) 1225 707126



Hello Java World

In this module we write the simplest of programs - a program which displays the words "Hello World". This shows you how to use the editor, compiler and runtime environment that you'll be using in subsequent models and gives you a minimal framework on which to expand.

<i>A first program explained.</i>	<i>4</i>
<i>A further program.</i>	<i>7</i>

2.1 A first program explained

Methods and classes

You write all your executable java code in "methods", and all methods must be grouped into "classes". Optionally, you can group classes into "packages" but we won't do that until later on.

A method is a piece of program code with a name. It can receive any number of input objects or primitives (known as parameters) when it is run, and it can return 0 or 1 object or primitive when it completes running.

A class is a group of methods, all of which are concerned with objects of the same type.

If you're already an experienced programmer from some other language, you may think that a method sounds very much like a function, proc, sub, subroutine, macro or command, and you would be right, although there is a little more to it. Java is an Object Oriented language, where you run a method *on an object*.

Let's see a first Java class and method in a form that we can run it as if it was a stand-alone program:

```
// Tiniest of programs to check compile / interpret tools

public class Hello {

public static void main(String[] args) {
    System.out.println("A program to exercise the Java tools");
}
}
```

Blocks and statement structure

Within a java source file, we group together program elements into blocks within curly braces ({ }) where blocks are frequently nested within blocks.

Our simplest program has two nested blocks. The outer block defines the class (it's called "Hello") and the inner block defines a method called "main". We could add other methods if we wanted, and they would go inside the outer block and outside the inner block. There is no practical limit to the number of methods we can define in a class, nor to the length of a block.

Within methods, our executable program code will consist of a series of statements, each of which ends with a ; character. Unless it specifies otherwise, each statement is performed in turn. The ; is mandatory, and you can put as much (or as little) white space as you like into a statement.

Declaring classes and methods

If you're declaring a class or a method, you have to tell Java the name that you want to associate with the class or method, and how accessible it is to be from elsewhere in your application.

Our class is declared in this example as

```
public class Hello
```

which means:

- It's available to any other class (**public**)
- It's a class (**class**)
- It's called Hello (**Hello**)

Our method is declared as

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

which means:

- It's available to run from any other class (**public**)
- It's not dependent on any particular object (**static**)
- It doesn't pass anything back to the code that calls it (**void**)
- It's called **main** (**main**)
- It takes one parameter, an array of Strings that it will know as "args"

This combination of keywords and choices just happens to be what the JVM for stand-alone programs is looking for when it's run - i.e. a static method called "main" that takes a String array of parameters when it's called. If you vary any part of that specification, then you might still be able to compile correctly but your code won't be able to run as a stand-alone.

Within a statement

Each Java executable statement comprises a series of operators and operands:

```
System.out.println("A program to exercise the Java tools");
```

println and "." are operators. The string written in double quotes is an operand, as is **System.out**.

When a statement is run, each of the operators operates on the operands before, after, or on both sides of it.¹ The result may form an operand which will be acted on further by subsequent operators.

Our first sample statement tells Java to print out a copy of the String that's in the brackets to the **System.out** channel. **println** is just one of the thousands of methods available as standard in Java 1.4. We suggest you get yourself a good reference book that lists them all in some sort of logical order as there's no way you'll remember them.

Reserved words

In our example, words like "public" and "class", "static" and "void" are understood by the Java Virtual Machine. Words like "main" and "println" are not understood by the JVM, but are nevertheless unchangeable as they are part of the standard classes and methods provided.

On the other hand, the words "Hello" and "args" are our choice, and we can change them if we wish. You must not use words that the JVM itself understands (they are "reserved words") for things you name yourself. You should also avoid using words that relate to standard classes and methods for things you name.

¹ depending on what the operator is

Commenting your source

The final element of our first program is the very first line. It appears to break the rules that we've given so far. It's not in a block, it doesn't seem to have operators, and it doesn't end in a semicolon.

It's a comment.

If the java compiler comes across two slashes when it's "tokenizing" the source code, it ignores subsequent text up to the end of the line. This allows you to put reminders of how your code works and what it does into your source. You can also write a comment starting with `/*` in which case everything up to the following `*/` will be treated as a comment.

It is *vital* that you comment any programs you write, providing information about what the program does, how it does it, notes of any tricks you've used, etc. Although it may take you a few seconds longer when you're actually writing the program, a few comments well placed can save you and your colleagues hours of headache later when you come to maintain or enhance the code.

The code in operation

Having explained the code in depth, let's see the whole program again, and then let's compile and run it:

```
bash-2.04$ cat Hello.java
// Tiniest of programs to check compile / interpret tools

public class Hello {

public static void main(String[] args) {
    System.out.println("A program to exercise the Java tools");
}
}
bash-2.04$ javac Hello.java
bash-2.04$ java Hello
A program to exercise the Java tools
bash-2.04$
```

2.2 A further program

You've seen a lot of structure to make up the simplest "hello world" program in Java. Fortunately we can now extend that to provide further functionality at very little cost.

Following is a Java program that uses just the first few features of the Java language that we've just covered to print out a series of lines of text:

```
/* This Java program will print out a series of lines of text. The class
Two has a main method (where the execution will start) and several other
methods, some of which are called */

public class Two {

    public static void beginner() {
        System.out.println("This is my starter");
        System.out.println("Still in my starter");
    }

    public static void main(String[] args) {    // Starts execution here
        System.out.println("A program to exercise the Java tools");
        beginner();
        another();
        System.out.println("Message from main");
        another();
    }

    public static void noway() {    // No calls to this one
        System.out.println("This will not happen");
    }

    public static void another() {
        System.out.println("I am here");    }
}
```

Figure 1 Running public class Two

```
bash-2.04$ java Two
A program to exercise the Java tools
This is my starter
Still in my starter
I am here
Message from main
I am here
bash-2.04$
```



Exercise

Write a java class with two methods in it, one called "employer" to print out the name of your employer, and the second called "employee" to print out your name. Add an extra method to the class so that you can run the class as a program under the stand-alone JVM, and add calls to that extra method so that your final result looks like:

```
bash-2.04$ java Who
Well House Consultants
Graham Ellis
bash-2.04$
```

License

*These notes are distributed under the **Well House Consultants Open Training Notes License**. Basically, if you distribute it and use it for free, we'll let you have it for free. If you charge for its distribution of use, we'll charge.*

3.1 Open Training Notes License

Training notes distributed under the **Well House Consultants Open Training Notes License** (WHCOTNL) may be reproduced for any purpose PROVIDE THAT:

- This License statement is retained, unaltered (save for additions to the change log) and complete.
- No charge is made for the distribution, nor for the use or application thereof. This means that you can use them to run training sessions or as support material for those sessions, but you cannot then make a charge for those training sessions.
- Alterations to the content of the document are clearly marked as being such, and a log of amendments is added below this notice.
- These notes are provided "as is" with no warranty of fitness for purpose. Whilst every attempt has been made to ensure their accuracy, no liability can be accepted for any errors of the consequences thereof.

Copyright is retained by Well House Consultants Ltd, of 404, The Spa, Melksham, Wiltshire, UK, SN12 6QL - phone number +44 (1) 1225 708225. Email contact - Graham Ellis (graham@wellho.net).

Please send any amendments and corrections to these notes to the Copyright holder - under the spirit of the Open Distribution license, we will incorporate suitable changes into future releases for the use of the community.

If you are charged for this material, or for presentation of a course (Other than by Well House Consultants) using this material, please let us know. It is a violation of the license under which this notes are distributed for such a charge to be made, except by the Copyright Holder.

If you would like Well House Consultants to use this material to present a training course for your organisation, or if you wish to attend a public course is one is available, please contact us or see our web site - <http://www.wellho.net> - for further details.

Change log
Original Version, Well House Consultants, 2004

Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____

License Ends.