

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

Java

Introduction

What is Java? What's it used for? What are the component parts?

The fundamental elements of Java 4

The Java World. 6

Java is a programming language and more. It originated from Sun Microsystem's Oak project and Sun still develop, maintain and supply it. Although it's not open source, nor delivered under the GPL license, much is available as a no-charge download from Sun's various web sites.

At its first release, Java primarily was used as a language for writing applications to be embedded in browsers (known as applets), but it has grown into many other areas. These days, applets are very much a minority use of Java, although still an important one. Other uses include web server-side programming (using "servlets" or "Java Server Pages") and large-scale, enterprise-wide applications using resource servers, "Enterprise Beans" and more.

2.1 The fundamental elements of Java

Java is an object oriented language, and all code you write is organised into classes. If you structure the way a class is defined and called according to certain rules, then that class may be usable as a program, or as an applet, or as a servlet. The code you actually write ("source code") is English-like text, and you save it into a regular text file, just as you do with other programming languages.

Source Code

Let's see an example of the source code of a Java 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");
}
}
```

We've saved this example into a file called Hello.java. Note that the file name should be the same as the class name declared in the file followed by ".java". This rule can be broken with some environments and compilers, but it's a good rule to follow.

Java is case sensitive. Note that we have started our class name with a capital, followed by lower case letters – another suggested convention.

Class files

Some languages¹ are interpreted and run directly from the source code, but Java isn't one of those. It's a language that's designed to run quickly, and interpreting the whole thing "public", etc., every time it's called is inefficient, so we compile the Java into a binary format. Let's use the "javac" program, supplied by Sun as part of their free downloads, to do the conversion:

```
bash-2.04$ ls
Hello.java
bash-2.04$ javac Hello.java
bash-2.04$ ls
Hello.class Hello.java
bash-2.04$
```

¹ Tcl, Shell, Jcl are examples

If things are OK, javac doesn't produce any message. But if your source code isn't in the correct syntax or refers to something that doesn't exist, you'll probably get an error message at compile time, such as:

```
bash-2.04$ javac Oops.java
Oops.java:5: cannot resolve symbol
symbol   : class string
location: class Oops
public static void main(string[] args) {
                        ^
1 error
bash-2.04$
```

Edit the source file, correct your error, save the file and compile again. You may have to go round this cycle a number of times until you get rid of all your errors. By the way, the mistake here was that we used a lower case "s" not a capital "S" for the word "String".

Once you compile successfully, the class file is in a published binary format, but it's very rare for most programmers to have to get involved at that level. Suffice it to say at this stage that the class file is compact and is independent of host computer architecture. Unlike most compilers, javac does not produce a snippet of executable code tuned for the particular processor architecture on which it is run. You might like to see what the format looks like. Here's a binary dump:

```
0000000  ?  ?  ?  \0  \0  \0  .  \0 035  \n  \0 006  \0 017  \t
0000020  \0 020  \0 021  \b  \0 022  \n  \0 023  \0 024  \a  \0 025  \a
0000040  \0 026 001  \0 006  <  i  n  i  t  > 001  \0 003  (  )
0000060  V 001  \0 004  C  o  d  e 001  \0 017  L  i  n  e  N
0000100  u  m  b  e  r  T  a  b  l  e 001  \0 004  m  a  i
0000120  n 001  \0 026  (  [  L  j  a  v  a  /  l  a  n  g
0000140  /  S  t  r  i  n  g  ;  )  V 001  \0  \n  S  o  u
0000160  r  c  e  F  i  l  e 001  \0  \n  H  e  l  l  o  .
0000200  j  a  v  a  \f  \0  \a  \0  \b  \a  \0 027  \f  \0 030  \0
0000220 031 001  \0  $  A  p  r  o  g  r  a  m  t  o
0000240  e  x  e  r  c  i  s  e  t  h  e  J  a
0000260  v  a  t  o  o  l  s  \a  \0 032  \f  \0 033  \0 034
0000300 001  \0 005  H  e  l  l  o 001  \0 020  j  a  v  a  /
0000320  l  a  n  g  /  O  b  j  e  c  t 001  \0 020  j  a
0000340  v  a  /  l  a  n  g  /  S  y  s  t  e  m 001  \0
0000360 003  o  u  t 001  \0 025  L  j  a  v  a  /  i  o  /
0000400  P  r  i  n  t  S  t  r  e  a  m  ; 001  \0 023  j
0000420  a  v  a  /  i  o  /  P  r  i  n  t  S  t  r  e
0000440  a  m 001  \0  \a  p  r  i  n  t  l  n 001  \0 025  (
0000460  L  j  a  v  a  /  l  a  n  g  /  S  t  r  i  n
0000500  g  ;  )  V  \0  !  \0 005  \0 006  \0  \0  \0  \0  \0 002
0000520  \0 001  \0  \a  \0  \b  \0 001  \0  \t  \0  \0  \0 035  \0 001
0000540  \0 001  \0  \0  \0 005  *  \u00000001  \u00000001  \0  \0  \0 001  \0
0000560  \n  \0  \0  \0 006  \0 001  \0  \0  \0 003  \0  \t  \0  \v  \0
0000600  \f  \0 001  \0  \t  \0  \0  \0  %  \0 002  \0 001  \0  \0  \0
0000620  \t  ?  \0 002 022 003  \u00000001  \0 004  \u00000001  \0  \n
0000640  \0  \0  \0  \n  \0 002  \0  \0  \0 006  \0  \b  \0  \a  \0 001
0000660  \0  \r  \0  \0  \0 002  \0 016
0000670
```

The Java Runtime Environment

How do we use a binary class that won't run on your computer? We run it through another program known as a Java Virtual Machine (JVM) which interprets each of the elements of the class file and performs the actions stipulated.

There are going to be many things that you want to do in your Java that others want to do as well, things as simple as outputting text. So along with the JVM, Sun provide a large library of extra classes which give you the facilities you need without having to write them yourself. There are so many extra classes that they're arranged into "packages" to make them more manageable, and the combination of the Java Virtual Machine and these standard packages is known as the Java Runtime Environment or JRE.

Let's run the example program that we compiled earlier in the "java" JRE, which lets us run an appropriate class as a stand-alone program:

```
bash-2.04$ java Hello
A program to exercise the Java tools
bash-2.04$
```

Exercise

Log in to the system that you're using for the course, and use any editor of your choice (the tutor will be able to help you find an editor that suits you personally) to create a file containing a Java program similar to ours, but which displays the message "This is my first Java Program" rather than "A Program to exercise the Java tools". Compile it through javac, and run it through java.

2.2 The Java World

Well House Consultants' courses specialise in the teaching of programming languages and their application, so you'll be going on in subsequent sections to cover the details of what to put into the source files and (unless this is a compressed course) how to think through the design of your application and source code so that it's well structured, easy to use, easy to maintain, and easy to update in the future as your requirement(s) develop. At first, a lot of this may seem very theoretic, so here we'll give you a brief glance further into the Java world that you're headed for.

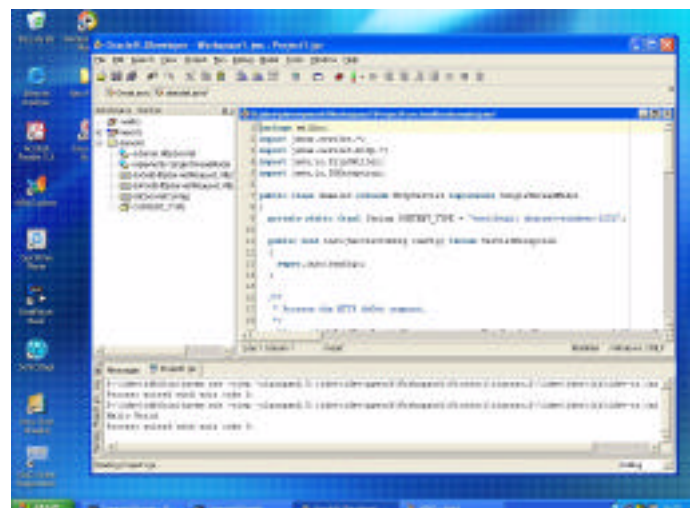
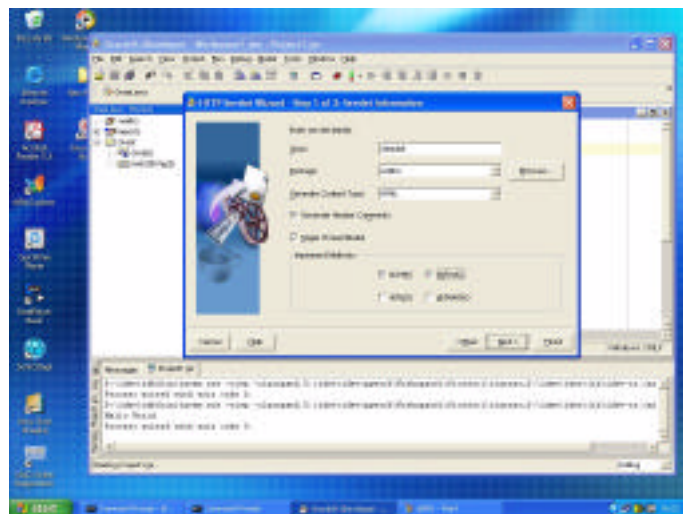
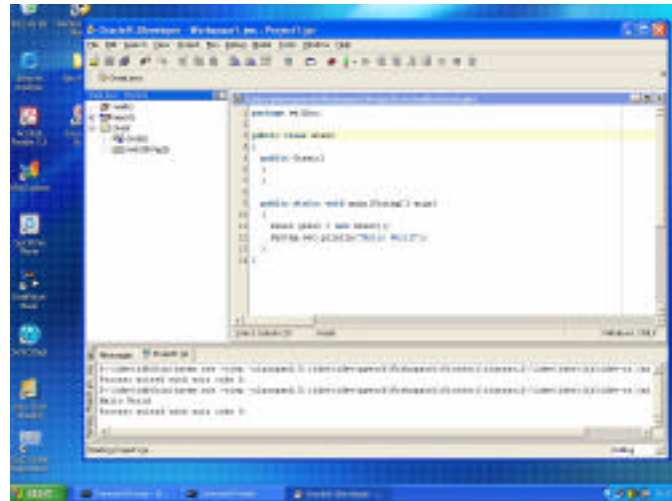
Java development environments and tools

As your Java source code grows in size, you'll find that it's quite a task to keep track of all the various classes and other components involved. Development environments such as JDeveloper, JBuilder and Forte provide you with facilities to make this management much easier, and with shortcuts that let you enter and edit code far more efficiently than you could with a standard editor. We find that it's best to teach you the fundamentals of the Java language before we expose you to these tools. Other-

wise, you're likely to find yourself spending a great deal of time trying to understand some of the suggested code and options that the tool offers but which we haven't yet covered.

To give you a flavour of the look and feel of a Java Development Environment, here are some screen shots from Oracle's Jdev. It's easier to learn than some of the environments we've seen, but you still need to understand the questions being asked and the code before you can make good use of it

Figure 1 Some screen shots from Oracle's Jdev



And here's the code that we generated for a simple Servlet:

```

package wellho;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;

public class demolet extends HttpServlet implements SingleThreadModel
{
    private static final String CONTENT_TYPE = "text/html; charset=windows-1252";

    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);
    }

    /**
     * Process the HTTP doGet request.
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException
    {
        String var0show = "";
        try
        {
            var0show = request.getParameter("showthis");
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>demolet</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a GET. This is the reply.</p>");
        out.println("</body></html>");
        out.close();
    }

    /**
     * Process the HTTP doPost request.
     */
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        String var0show = "";
        try
        {
            var0show = request.getParameter("showthis");
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>demolet</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a POST. This is the reply.</p>");
        out.println("</body></html>");
        out.close();
    }
}

```

Sun's distribution includes a number of tools in addition to `java` and `javac`, including:

jar	a tool for creating and manipulating java archive files (jars)
javadoc	generates API documentation from source files
javap	generates a human-readable description of the API of a class file
jdb	a text-based debugger

Java is ideally suited for larger projects. Such projects may involve considerable management of development and runtime files, such as sources, classes, libraries etc., where many of the files are derived from others and need to be updated when any of the files on which they depend are changed. Apache Ant is a Java-based build tool. In theory, it is somewhat like `make`, but without `make`'s wrinkles. It's open source and becoming very popular. See <http://ant.apache.org/> for further details.

Java Runtime Environments

Stand-alone programs probably aren't what you'll be writing in the longer term, even though they're excellent for learning the fundamentals of Java. Later on you'll be slotting your classes into other Java runtime environments, some of which are open source and others are commercial products.

Server side, there are a number of environment interfaces you may use. These include:

Servlets	Executable programs with a web interface
JSPs	Embedding server executable content within a web page
RMI and EJBs	Providing object servers

JREs that support these interfaces include Apache Tomcat, BEA WebLogic JRockit and IBM's WebSphere application server.

Client side, most users want to access information via their browser these days and plugins that support Java are available for most common browsers. Java is built in to certain browsers too, but do beware that it's often in old (or even ancient) versions. AppletViewer, a part of Sun's distribution, is a JRE with the same interface that a browser provides but without the caching or overhead of a browser, and is useful for development and testing.

And it turns out that the JRE that you use for stand-alone programs can do much more. You can have what looks like a stand-alone program at "the top" but have it act as a client or server (or both!). You can have it provide a graphic user interface (GUI) in much the same way that an applet would, and much more.

Java distributions

For one programmer, Java might be the tool he uses to program a toaster. Another might be using it to provide the core financial accounting services for a major bank. Is it possible for both of these requirements to be met by the same downloaded language? Yes...and no.

Java is described as a "simple" language and indeed at it's centre it is. The syntax is not overcomplex, the facilities of the language itself are relatively few, which make it into something of a lean and mean core facility. The real power comes in the standard packages that are available, and that's where the requirements of our domestic appliance programmer will vary from the requirements of our banker.

The Java 2 Standard Edition (J2SE) provides the essential compiler, tools, runtimes, and APIs for writing, deploying, and running applets and applications in the Java programming language. All developers will need to download a copy of J2SE, or its equivalent.¹

The Java 2 Enterprise Edition (J2EE) technology and its component-based model

¹ you'll find that many of the development environments we mentioned earlier include it

simplifies enterprise development and deployment. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and inter-operable business applications. The download has essential extra classes and environments to use in addition to the J2SE, which you will also need.

The Java 2 Micro Edition (J2ME) specifically addresses the consumer space, which covers the range of extremely tiny commodities, such as smart cards or a pager, up to the set-top box. This download provides a highly optimised runtime environment. Again, to develop code, you'll also need J2SE.

All of the above can be downloaded from <http://java.sun.com>

Also available is a Java Runtime environment (J2RE) which will be required by users rather than developers. It includes the JVM and the classes that make up the JRE, but not tools such as the compiler. Different licensing rules will apply.

Java standard packages

Much of the power of a Java application is vested not in the language itself, but in all the standard classes provided and optional classes available. There are so many classes that they've been organised into bundles (called "packages") for easier management. The first release of Java included eight packages. These days it depends on just what edition you're talking about, but you'll probably find you have somewhere more than 100 packages in your JRE.

Standard package names start "java." or "javax.". For example, there's *java.lang* to provide basic language facilities, *java.net* to provide network access, *javax.swing* to provide the Swing GUI and *javax.servlet* to provide Servlet support. You'll come across many more standard packages as you learn Java; however, as a rule of thumb, if the package name starts "java" or "javax", it's standard, but if it starts with something else like "com", it isn't.

Java versions

Java started off as Java release 1.0, and progressed through to Java 1.5. Sun were very conservative in moving the release numbers forward, so much so that the "1." just became a part of the name. In summer 2004, Java 1.5 was re-branded Java 5.

Java 1.2 was a major step, with a tripling of the number of packages provided. At that point Sun rebranded the new version the "Java 2 Platform".

Although Java is designed to be processor- and operating-system independent, you need to be careful as you develop code that you do so for a runtime environment that will be available to your user. See how the number of standard packages and classes has increased:

classes/packages

Java 1.0	212	8
Java 1.1	504	23
Java 2 1.2	1520	59
Java 2 1.3	1842	76
Java 2 1.4	2991	135
Java 2 5.0	more	more

As well as the extra packages, there have been some language changes, such as an **assert** statement added to the language at release 1.4.

Note that there is a very large installed base of browsers running Java 1.1, so you may still want to use it for writing simple applets. If that's the case, you'll need to obtain a copy of the JDK (Java Development Kit) for that release. The JDK became the SDK (Software Development Kit), now Java 2 SDK Standard edition, Version 1.5 or 5.0

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.