Notes from

Well House Consultants

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 disctibute 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

2

Java in the Web Page

Java programs run in JREs – Java Runtime Environments. Those runtime environments may be traditional "keyboard to screen" applications, or they may run on servers, or within web clients (browsers). This module shows you how you can incorporate a java class (known as an Applet) within a web page, so that the methods of the class are run by the browser. Such applets are typically used to provide intelligent forms and graphics.

2.1 Structure overview

The Java programs which we have studied thus far on this course have all been run from the command line using a command such as:

```
java corsham
```

In other words, running the **main** method in the class *corsham* as an application, courtesy of the Java Virtual Machine supplied with the Java Development Kit.

It is likely, though, that you want to use Java to add interactive content to a web page. In which case you'll not be using the command line; you'll want instead to run a Java applet.

Let's look at the various component parts involved in creating a web page with the Java applet.

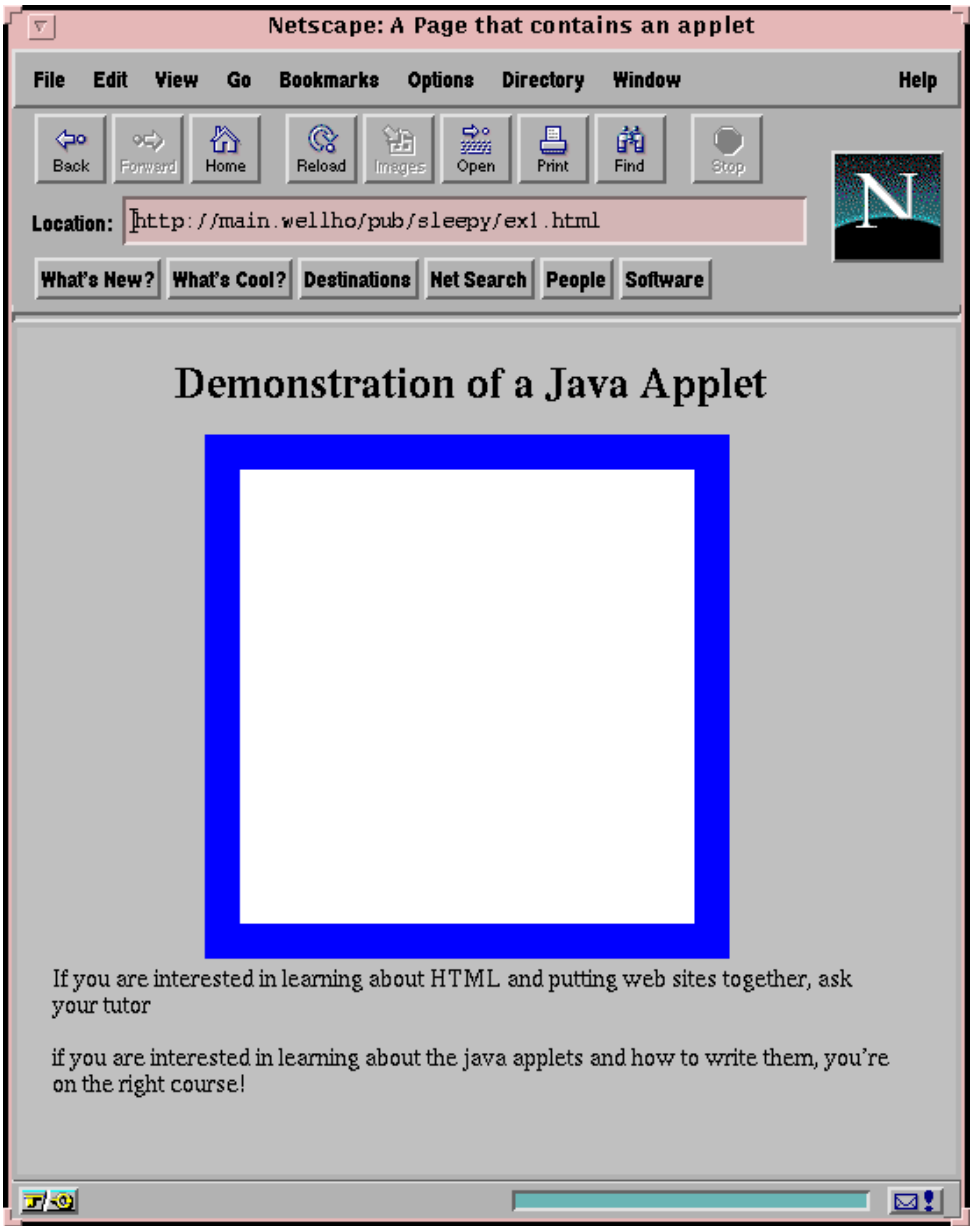


Figure 1 Web page containing a Java applet

To view Java in a web page, you need a Java-enabled web browser. These days, that will typically be a recent version of Netscape or Internet Explorer from Microsoft.

To create a web page with Java content and make it available on the Internet, or on your in-house Intranet, you will need:

- Access to a web server / the site from which the pages will be downloaded
- Knowledge of how to create a web page using HTML (either directly or indirectly) and how to load that page onto your web site
- Knowledge of the Java language, and how to write Java applets
- A Java compiler, although this need not be on the web server system.

On this course, we provide a web server on which you can test your work.

A prerequisite of this part of the course is a knowledge of the mechanics of the web and the ability to create your own page. If your knowledge is patchy or rusty, your tutor will assist as his time allows.

Web pages comprised of HyperText documents written using HTML tags and pages which include Java applets are no different (apart, of course, from the tags used!). Here is the HTML source for the page we showed at the start of this section:

```
<HTML>
<TITLE> A Page that contains an applet </TITLE>
<BODY>
<CENTER>
<H1>Demonstration of a Java Applet</H1>
<applet code=box.class width=300 height=300>
</applet>
</CENTER>
If you are interested in learning about HTML and
putting web sites together, ask your tutor<P>
if you are interested in learning about the java
applets and how to write them, you're on the
right course!
</BODY>
</HTML>
```

The applet tag is a request to run the applet contained in the class *box* in a rectangular box, 300 x 300 pixels, at the given point in the text flow.

When your Java-enabled browser sees the applet tag, it asks the server for the class file in much the same way that it would ask for a graphic on finding an **<IMG>** tag.

The class file is then downloaded and run by the browser.

When developing an applet, you provide not a single main method, but a series of up to six methods which extend the applet class.

The applet should be developed, as usual, in a *.java* file, and the Java compiler should be used to create the *.class* file. Once compiled successfully, the *.class* file should be copied to the same directory from which the HTML is downloaded. Upon (re)load, the new applet will be requested by the browser, downloaded and run.

Nearly all browsers assume that an applet has not changed from one invocation to the next. If you correct an applet and want to view the new version, you have to quit and restart your browser!

Note: The **<applet>** tag should be replaced by the **<object>** tag if you are writing applets to run on any intranet where all the browsers are guaranteed to be recent editions.

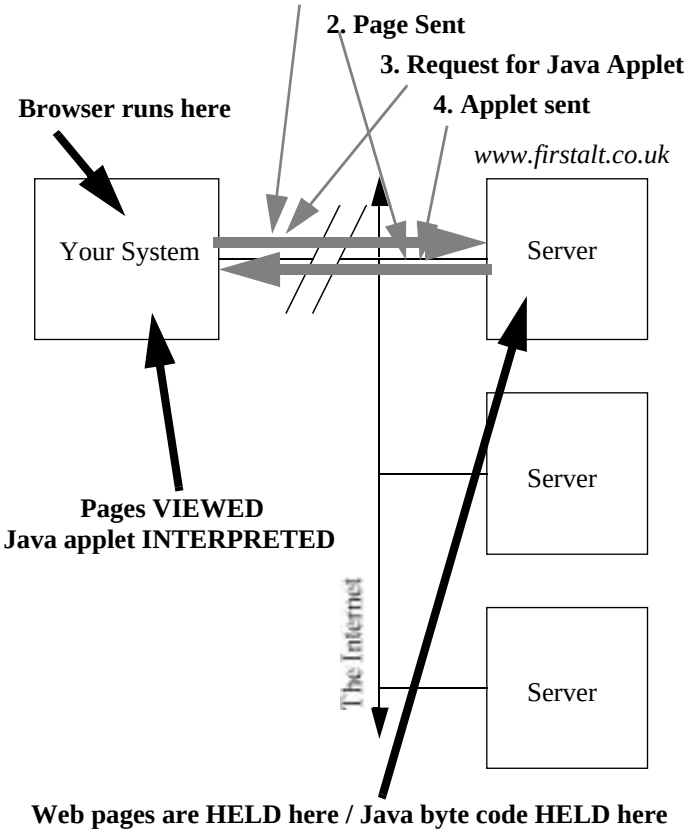


Figure 2 Analysing how an applet is read

Provided with the Java Development Kit is an alternative application called "AppletViewer" which lets you run an applet from a snippet of HTML.

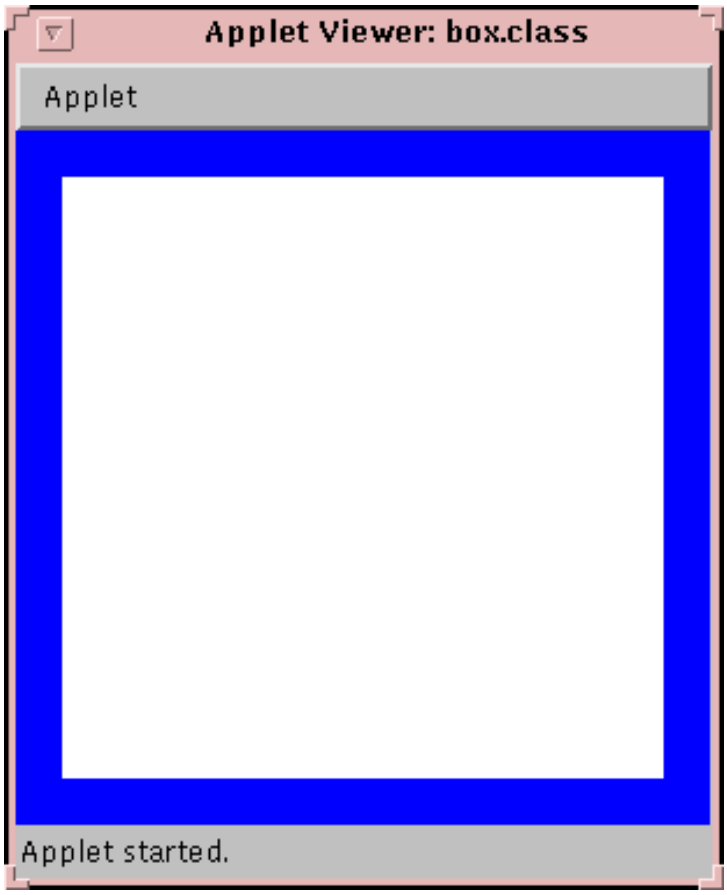


Figure 3 Running Applet Viewer on the box.html and box.class files

The command line was as follows:

```
seal% appletviewer box.html
seal%
```

And the *box.html* file:

```
<HTML>
<BODY>
This is a java applet:
<BR>
<applet code=box.class width=300 height=300>
</applet>
<BR>
</BODY>
</HTML>
```

The actual Java source code for this example was:

```
// Well House Consultants 2004.
/** First Applet */
import java.awt.*;
public class box extends java.applet.Applet {

   public void init() {
   }

   public void start() {
   }

   public void paint(Graphics mycanvas) {
   mycanvas.setColor(Color.blue);
   mycanvas.fillRect(0,0,size().width - 1,
                         size().height - 1);
   mycanvas.setColor(Color.white);
   mycanvas.fillRect(20,20,size().width - 41,
                         size().height - 41);
   }

   public void stop() {
   }

   public void destroy () {
   }
}
```

2.2 The methods you may and must provide

The main methods you may choose to override in the java.applet class (or its superclasses) are:

init	called when the applet is first loaded
start	called when the applet becomes visible
paint	called when the applet is to display itself
print	called when the applet is to print itself
stop	called when the applet becomes invisible
destroy	called when the applet is unloaded

Other methods are used to allow the applet to get information about its environment, and many more methods from the **awt** classes are used to handle the applet window.

2.3 Including Java in your page: HTML tags

Where is an applet placed within a document?

At the location dictated by the **<APPLET>** and **</APPLET>** pair of tags!

Within the applet tag itself, you will for certain have the following attribute set:

- CODE

which actually tells the browser the URL of the Class file containing the applet executable.

It is very likely that you will have:

- HEIGHT and WIDTH

which specify the height and width (in pixels) of the display area to be used by the applet.

Other attributes you might have in Java, and which will be familiar to HTML programmers, include: ALIGN, ALT, HSPACE and VSPACE.[1] Also in use is CODEBASE, which is rather like a BASE tag, but affects only the location of applet code.

Also be aware of these new attributes used in Java:

ARCHIVE -- which lets you specify that the Java Class downloaded is to be archived on your local client system -- the suffix of the given file name indicating whether you are going to use a *.zip* file or a *.jar* (Java archive) file, up and coming, and with extra security features like digital signatures!

ARCHIVE tags -- which are supported only on Netscape at the moment, can give enormous savings in download time if you use the same applet in session after session.

MAYSCRIPT -- necessary if the Java applet is to access Java*Script* features within the browser.

NAME -- lets you provide unique names for each instance of an applet in your page. Useful if you have two copies of the same applet running at the same time, and other applets need to know which to talk to!

TITLE -- used by Internet Explorer when it wants to provide a title for the applet!!

We have considered **<APPLET>** and **</APPLET>**.

Why are there two tags? What can possibly go between them?

First, any plain text which occurs between the tags will be ignored by Java-knowledgable browsers, but will be displayed by browsers which do not know of Java.

Sample of HTML code

```
<APPLET CODE=graph.class ALT="Cannot Access the applet
   graph.class which should display here">
Your Browser does not know about Java.<BR>
If Java were available, a graph of the share price would be
displayed here.
</APPLET>
```

And what is displayed? The result of running the applet *graph.class*, you hope, except ...

A message to say that the class cannot be accessed if that is the case ... or

A message to tell you that your browser does not know about Java, if that is the case!

Hang on a minute.

[1] these will be familiar to those of you who know about placing images

Graphing applet? Good idea, but where does it get its data?

Built-in? Well, so far, yes, that's all that we've looked at. Works well enough, but a little limited.

What if the server wants to send that data without having it built-in?

Let's take, for example, a scheme drawing a graph of share prices for the last five days, and work up towards seeing how that would work out. What might you want to do?

- Complete a form giving the name of the share(s) or interest to you
- Receive a graph and statistics
- Change to a histogram, change scales, switch from an actual to a smoothed curve
- Add in / take out some of the shares from the graph

The data can be sent to your applet through the **<PARAM>** tag, which takes a series of names and values.

2.4 The Abstract Windowing Toolkit

What's actually new in terms of the Java language in our applet?

Nothing really. We're just providing a number of methods which extend the *java.applet.Applet* class.

And yet the class file does look different, mainly because our first (and simple) example is not doing any great calculations and is comprised almost entirely of calls to methods that we haven't yet met, in classes that are still new to us!

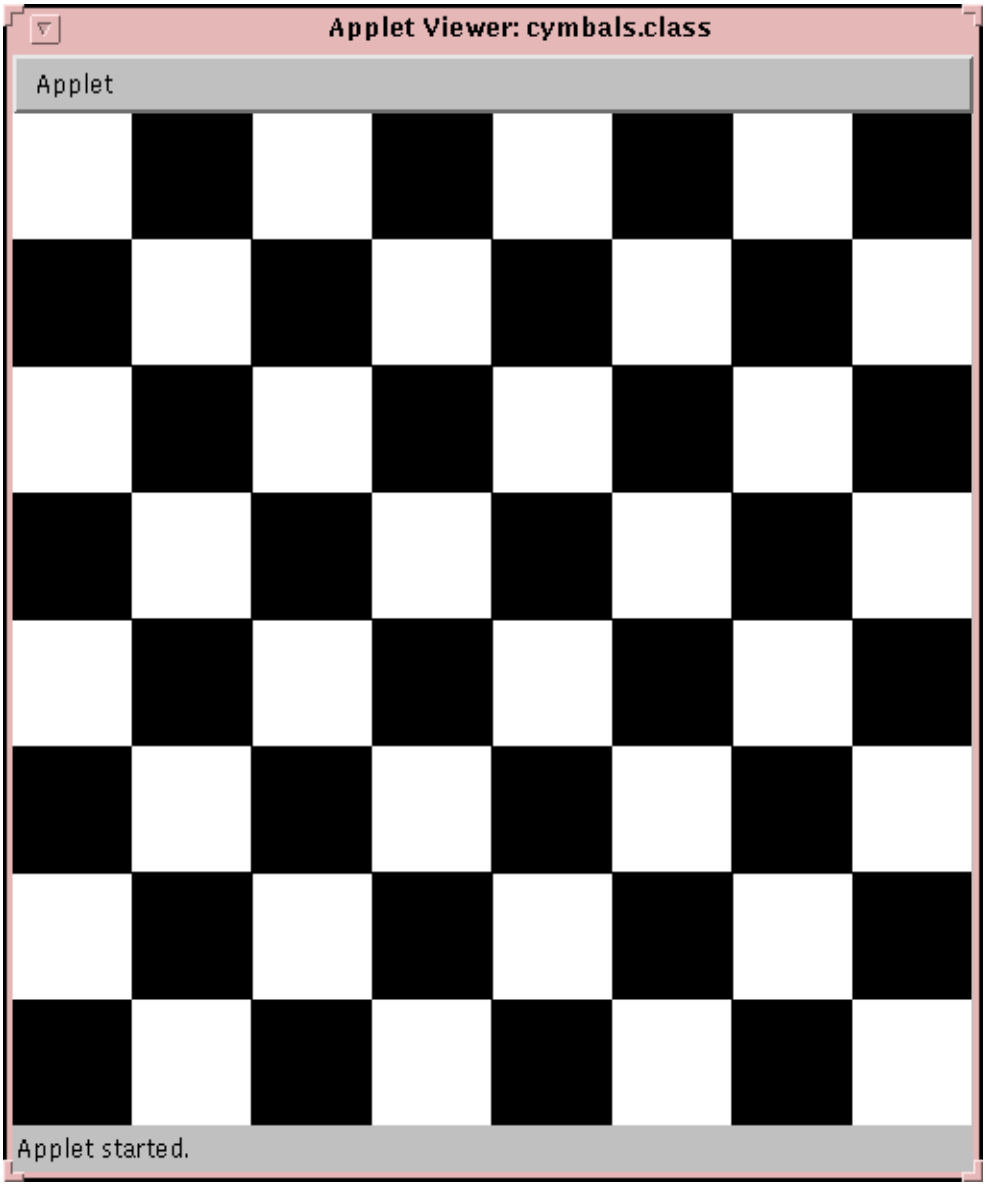
Instead of using **System.out.print** and other methods that we'll learn about before the end of this course, applets produce their graphics through the Abstract Windowing Toolkit, a big topic which we'll see in detail on the *Java Advanced* course.

Exercise

Write an applet to draw a chess board:

Our example answer is cymbals

```
seal% appletviewer cymbals.html
```



For Advanced Students

Select one or more from the following:

- Draw the chess board on a red table, i.e. add a red border
- Try using different WIDTHs for your applet. How can you keep the chess board square?
- Enhance your web page so that when you view it through a browser which is not Java-enabled, that fact will be reported to you. [Check with your tutor as to which browsers are available to try this.]

3

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.

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 _____________________
Updated by: _____________________ on _____________________
```

License Ends.