Exercise 1 Part B

The first part of this tickable exercise is issued as part of the introduction to the use of Linux on the PWF. The task set here is thus “Part B” of the complete Tick.

Log on to the PWF. Create a new directory and select it as your current one, eg

```
mkdir Tick1B
cd Tick1B
```

Issue the following commands that copy two files form the Computer Lab’s teaching filespace into your new directory.

```
cp $CLTEACH/acn1/TickBase.class .
cp $CLTEACH/acn1/TickBase1.class .
```

You should be able to check that the files are present. These two files provide a basis upon which the exercise builds.

Now inspect Figure 1 which is documentation associated with the two files that you have just copied. A more extensive version of the same material is available on-line as

`www.cl.cam.ac.uk/Teaching/current/ProgJava/notes/TickDoc/`

Now prepare a file that called `Tick1.java` containing the text

```
// Tick 1.  Your Name Goes Here

public class Tick1 extends TickBase
{

public static void main(String []args)
{
    (new Tick1()).setVisible(true);
}

public String myName()
{
    return "Your Name";
}

}
```

Obviously you will put your own name in the places that are suggested by what I have written here!

Class TickBase

public class **TickBase**

The "TickBase" class provides a foundation for Java Tick 1. It i
intended to be used by writing a new class that extends it and
provides it with a "main" method, as in

```
public class Tick1 extends TickBase
{

public static void main(String []args)
{
   (new Tick1()).setVisible(true);
}

public String myName()
{
   return "Arthur Norman";
}

}
```

and this will lead to an application that can be launched and will
display a certificate confirming success.

Field Summary

Constructor Summary

TickBase()	The sample version of "main" uses a constructor called TickBase() that makes an instance of the object that displays certificates.

Method Summary

java.awt.Color	**myColour**() By overriding the myColour method you can change the colour used in the certificate.
java.lang.String	**myName**() Override the myName method with a version that returns whatever your own name is to personalise the certificate that you receive.

Constructor Detail

TickBase

public **TickBase**()

The sample version of "main" uses a constructor called TickBase() that makes an instance of the object that displays certificates.

Method Detail

myName

public java.lang.String **myName**()

Override the myName method with a version that returns whatever your own name is to personalise the certificate that you receive. To receive a tick you are required to do this.

myColour

public java.awt.Color **myColour**()

By overriding the myColour method you can change the colour used in the certificate. The default is Color.BLUE

Figure 1: Documentation of Tick 1 Part B.

Compile your program and then run it:

```
javac Tick1.java
java Tick1
```

If all has gone well a window should appear, and it should have some text and a pattern on it. There is a menu that you can select. If you copy the files to your own machine you can try the `print` menu, but on the PWF there are technical reasons why that is not supported, and these lie outside just Java. On *some* systems it may be OK, but not all! So select the menu item labelled `postscript`. You should then see a dialog box asking you to choose a file name. I suggest that you select the name `tick1.ps` and I very strongly suggest that you use the extension `.ps` whatever name you actually choose. When you accept the file-name you have chosen the "select file" dialog box disappears and you can not see that anything much has happened, but the file you indicated should have been created for you. It should contain an image of the screen window in the Postscript document format. Close the little Java window, and you can send this to a printer using the command

```
lpr tick1.ps
```

The resulting sheet of paper is what goes to your ticker.

As an optional extra you can arrange to change the colour of (some of) the text generated by adding lines roughly like the following to your Java source file:

```
public java.awt.Color myColour()
{
    //                          RED  GREEN  BLUE
    return new java.awt.Color(0.7f,  0.1f,  1.0f);
}
```

where the three floating point numbers given (note that you have to write a letter 'f' at their end) should each be in the range 0.0 to 1.0 and they give the proportions or red, green and blue in the colour.

You can also check what happens if you present your name in different ways. For instance I tried "A C Norman" as well as "Arthur Norman". If you wanted to keep your program in a file called say `MyTick.java` rather than `Tick1.java` you would have to change its name within the file too. Verify that you can do that.

Discussion

This exercise is intended to send several signals and messages about Java:

- One can build new programs building on existing components that do quite a lot for you. Here you copied in the TickBase class files, but your own program then builds on them and can customise the behaviour of the provided code in various ways. Through doing this a very short fragment of code let you create a window and print its contents;

- To use the software component TickBase you do not need to see its internal structure: all you need is documentation about how to use it. As part of stressing this I am not going to provide you with the source code of TickBase, but by the end of this course you will probably be able to re-create it;

- Part of the way of using components like this involves the Java keyword `extends`, and part of the way that the code runs involves the keyword `new`. These are both key parts of the Object Oriented structure of Java, and you should look forward to finding out more about just what they really mean later on.

- The page of documentation included as part of these notes tells you that TickBase is interested in a `myName()`. This documentation is in the style of the bulk of the Java on-line documentation, and was created by using a simple tool called `javadoc` that interpreted some special comments in the TickBase source code. However the full output from javadoc is on the web page listed a little earlier and perhaps gives a bit more of an idea of just how much complexity is involved under the surface. The lesson that I learn is that if you use javadoc for anything other than a full-scale project you will need to edit its output heavily to remove material that your audience does not really need to see.

(End of tickable exercise)