**Exercise 1: Hello World, *graphically!***

The starting point for learning a new language is often a very simple program that simply says “Hello” to you. These “Hello World” programs usually reveal basic operations, introduce syntax, and give you a chance to get started.

Such a program might look like this:

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

However, even this simple program raises the question: *where*? That is, where does this message appear? to whom or what is it delivered? As it happens, the program is sending the message to the *System output device, or console*. If all we cared about were text input and output, we would use this framework, but we want to do this as part of a Java web *applet*, and we want to do it *graphically*, which means we have to think about it as constructing a graphic display containing some text, and we have to answer questions like “what color?” “how big?” and “what font?”.

We do this in the applet by sending messages (commands) to a “graphics” object, telling it about the graphic we are creating. That’s why the source on the next page includes “import java.awt.*;”

**Exercise 1: Hello World, *extends Applet!***

The programs created for this course are considered “fragments” of a larger application (the web browser), with which they can interact to some extent. Thus, they are not “applications” (which have a “main” method, as shown above), but “applets”, which “extend” (a Java concept) a pre-existing framework known as “Applet”. Unlike “main” programs, applets go through certain preset life-stages. That is, they receive certain “messages” (method calls) based on events in the containing browser’s universe—events like “init”, “start”, “stop” and “destroy”. And because applets are primarily graphical creatures, they also receive a “graphics environment” (aka a “Graphics” object) and messages instructing them to “paint” and “update” the graphics environment (due to browser window scrolling, resizing, etc.) That’s why the source on the next page includes “import java.applet.*;”

**Exercise 1: Hello World, *going beyond the basics!***

So, our HelloWorld (no spaces!) applet receives a “draw here” message (via the paint method and associated Graphics object) when the browser is ready to display the applet’s results. For static output, all we have to do is write “paint”, as shown in the source on the next page. However, for maximum learning, investigate alternative values for font, color, and position, and consider what else might be fun to do. Then look and see if you can find out how to do it. If you can, share the “how” with us. If you can’t, ask.
Exercise 1: “Hello World”

The Java source code

```java
package ex1;
import java.awt.*;
import java.applet.*;

public class HelloWorld extends Applet
{
    public void paint( Graphics g )
    {
        Font f1 = new Font("Serif",Font.PLAIN, 48);
        g.setColor( Color.BLACK );
        g.setFont( f1 );
        g.drawString( "Hello World!", 10, 200);
    }
}
```

The HTML for the web page

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Sample Applet Runner</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<body bgcolor="tan">
<h1>Exercise: Hello World</h1>
<h2>HelloWorld.java</h2>
<h3>Author: Jan Doe</h3>
<p>My first applet!</p>
<APPLET codebase ="./" code="ex1.HelloWorld.class" name="HelloWorld"
width="400" height="400"
alt="Your browser understands the &lt;APPLET&gt; tag but isn't running the applet, for some reason.">
Your browser is completely ignoring the &lt;APPLET&gt; tag!</APPLET>
<p><a href="HelloWorld.java">Applet Source</a></p>
</body>
</html>
```