Java - Applets

C&G criteria: 1.2.2, 1.2.3, 1.2.4, 1.3.4, 1.2.4, 1.3.4, 1.3.5, 2.2.5, 2.4.5, 5.1.2, 5.2.1, 5.3.2.

Java is not confined to a DOS environment. It can run with buttons and boxes in a Windows like manner. Forte is written in Java and gives a good example of what Java can and cannot do with a GUI. When running graphically Java can run within an applet or as a stand-alone application. Here Java will be run in an applet that requires another program to host the applet. This will usually be the Java Virtual Machine in a web browser or the applet viewer that comes with the JDK. Java 2 has more sophisticated graphics than 1.1 and earlier through the use of swing. Some older browsers (IE 3 and possible IE 4) will not be able to handle swing although an up to date JDK will by use of the applet viewer. VJ will not create swing code but Forte will.

Java code here will not use swing but will support the 1.1 event model. Legacy code from the 1.0 event model will not be used. This code sets up a button to be pushed:

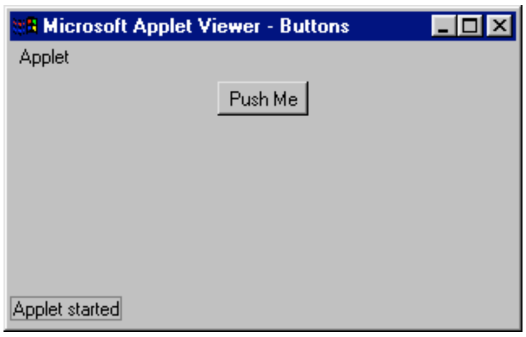
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Buttons extends Applet implements ActionListener

{
        private Button push;

        public void init()
        {
                push = new Button("Push Me");
                add(push);
                push.addActionListener(this);
        }

        public void actionPerformed(ActionEvent event)
        {
                if(event.getSource()==push)
                        push.setBackground(Color.blue);
        }
}
```



The Button turns blue when pushed. This is a lot of code for a result that could be done in about 3 lines with VB. Forte allows Buttons to be dragged and dropped and has a go at creating the code. care should be taken here as it is not a good way to see what the code does and the event code generated is less than ideally concise. The buttons that can be drawn with VJ are Microsoft AWT controls and will not work with pure Java.

There is a lot of new code here, some key points are:

1. The import statement is used to reference parts of Java and their methods that are not included by default. Import is similar to #include<> in C.
2. public static void main (String args[]) has gone. With applets public void init() acts as the main start up method.
3. Button like String is a Class so merits a capital letter.
4. init() is used to create the controls and actionPerformed is fired when the control is clicked. With many controls some sort of selection is required to tell what is going on.

Modifications

1. Add some more controls, they do not have to do anything. All controls must first be declared then added. Some will also deserve some sort of event listener.

TextField allows data to be typed in, the number of characters in the TextField must be specified.

```
myTextField = new TextField(10);  //holds 10 characters
```

TextArea is the big brother it holds more than 1 row of data.

```
myTextArea = new TextArea(10,5) //10 by 5
```

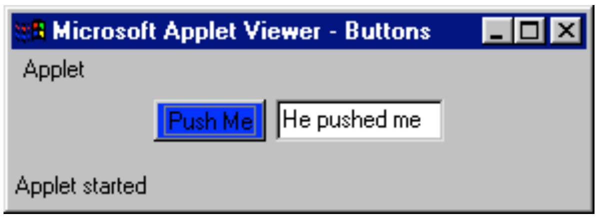
Label allows instructions to be shown. Data cannot be typed into a Label.

```
myLabel = new Label("A Caption");
```

2. Have pushing a Button put some text in a TextField. If the TextField was called output the key line would be:

```
output.setText("He pushed me");
```

This is fired by the condition - event.getSource()= =push



3. Write code to input data from a TextArea and output it to another.

If a String is declared called message the code to put the TextArea text into the String would be:

```
message =input.getText();
```

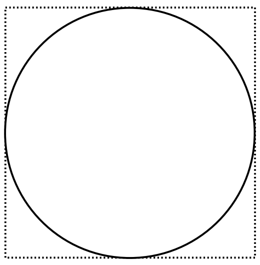
Drawing on the canvas

Applets make a lot of use of drawing to place shapes lines and words onto the screen. This is handled by the paint(Graphics g) method. Add this method the current program.

```
public void paint(Graphics g)
{
        g.drawString("A String",10,70);
}
```

Graphics is a Java library within java.awt, g.drawString is the method to draw a String at the x, y co-ordinates 10,70. Add code to draw squares, filled squares, circles and filled circles within paint, draw String next to the shapes to say what they are. These are all methods of g, for example:

```
        g.drawOval(40,80,140,180); // the 4 co-ordinates represent a box within which the
shape is drawn.
```



To draw the circle I must specify the 4 corners of the square surrounding it

So far all the shapes and text have been black. Colour can be changed by the Graphics method setColor. When typing the dot operator after Color the drop down shows all the colours that are available within Java. Note there are not very many of them.

```
        g.setColor(Color.red); //this code will draw a red oval.
        g.drawOval(40,80,140,180);
```

Adapt the shape code to draw each shape and its corresponding descriptive String in a different colour.

Drawing with an event

It is more useful if drawings are created by a user event such as clicking a Button or by some decision in code. This is achieved by calling the paint method from an actionPerformed or other place in code through repaint(). This code enables a Button to change the drawing colour to blue.

```
private Color drawColour;
//declare a Button
//add the Button in init()
public void actionPerformed(ActionEvent event)
{
        if(event.getSource()==push)
        {
                drawColour = Color.blue;  //assign a different colour to the Color variable
                repaint();  //call painy
        }
}
public void paint(Graphics g)
{
        g.setColor(drawColour);
        g.drawOval(40,80,140,180);
}
```

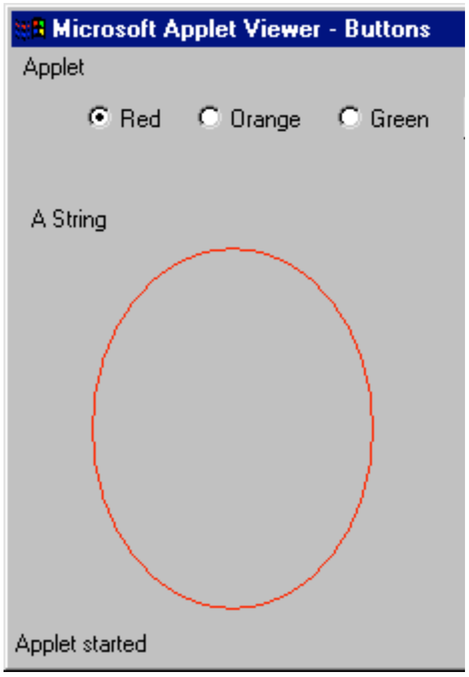
Modifications

1. Set up 2 TextFields and 2 Buttons each Button draws the text from its corresponding TextField onto the screen.
2. Allow the user to choose from 3 colours to display the Strings. This can be done with Buttons a better approach is described below.

Check box groups

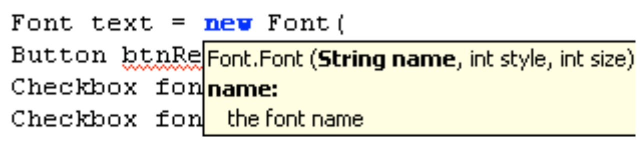
These are the same as option buttons in Microsoft land. They allow 1 choice from amongst the group. They act quite similarly to Buttons except rather than using ActionListener they respond to ItemListener so the Class has to be extended to include this.

```
private Color drawColour;
private CheckboxGroup colours;   //declare the group
private Checkbox red, orange, green;  //these are the Checkboxes within the group
public void init()
{
        colours = new CheckboxGroup();  //new the CheckboxGroup
        red = new Checkbox("Red",colours, true); //red is default
        add(red);
        red.addItemListener(this); // note addItemListener not addActionListener
        orange = new Checkbox("Orange",colours, false); // colours is the group
        add(orange);
        orange.addItemListener(this);
        green = new Checkbox("Green",colours, false);
        add(green);
        green.addItemListener(this);
        //add other controls
}
public void itemStateChanged(ItemEvent e)
{
        if(e.getSource()==red)
                drawColour=Color.red;
        if(e.getSource()==orange)
                drawColour=Color.orange;
        if(e.getSource()==green)
                drawColour=Color.green;
        }
}
```



Fonts

All the previous code has used a default font style and size, g.setFont( )allows this to be changed. Not surprisingly Font is a Class and an object from that Class is created with new. Create a new Font object and stop when the '(' is typed. Note what is needed to specify a Font



```
Font text = new Font(
Button btnRe
Checkbox fon
Checkbox fon
```

Font.Font (**String name**, int style, int size)
name:
the font name

name is restricted to the few names of Fonts that Java supports. These must be typed correctly.
style is normal (0), bold (1), or italic (2). Bold + italic is 3.
size is the font size, any smallish integer will do.

Declare this Font with the variables at the top of the Class.

```
        Font text = new Font(""SanSerif",Font.BOLD,18);
```

Then set the Font in paint(Graphics g).

```
        g.setFont(text);
```

Some example font names are:

- SanSerif
- Serif
- Monospaced
- TimesRoman

Modifications

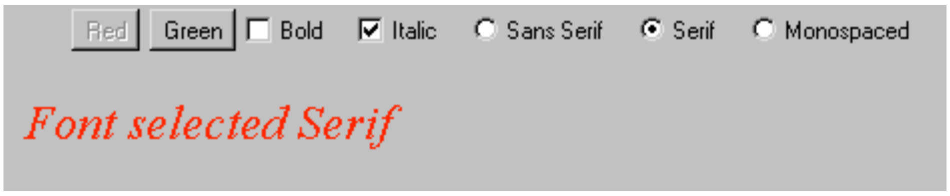
1. Set up a CheckboxGroup to allow the user to select the font that text is displayed in.

2. The font style could be normal, italic and/or bold. A CheckboxGroup would not be appropriate here but individual CheckBoxes that are not part of a group can be selected in multiples. These can be used to set the value of style. Note that style is an integer so can be declared as such and altered by the itemStateChanged code for each CheckBox. A possible solution is to have an integer for bold and italic and add them up when setting the font. The values of bold and italic are set by the itemStateChanged methods for the relevant CheckBoxes. To tell if they have been checked or not getState is required.

```
if(e.getSource()==bold) //reference the control
{
        boolean state = bold.getState(); //is it checked or not
        if(state)  //or if(state == true)
                iBold = 1;
        else
                iBold =0;
}
```

Font text can then be declared in paint rather than at the top of the class, it will get its font style from the value of bold. Italic will work in the same fashion.

```
Font text = new Font("SanSerif",iBold,18);
```

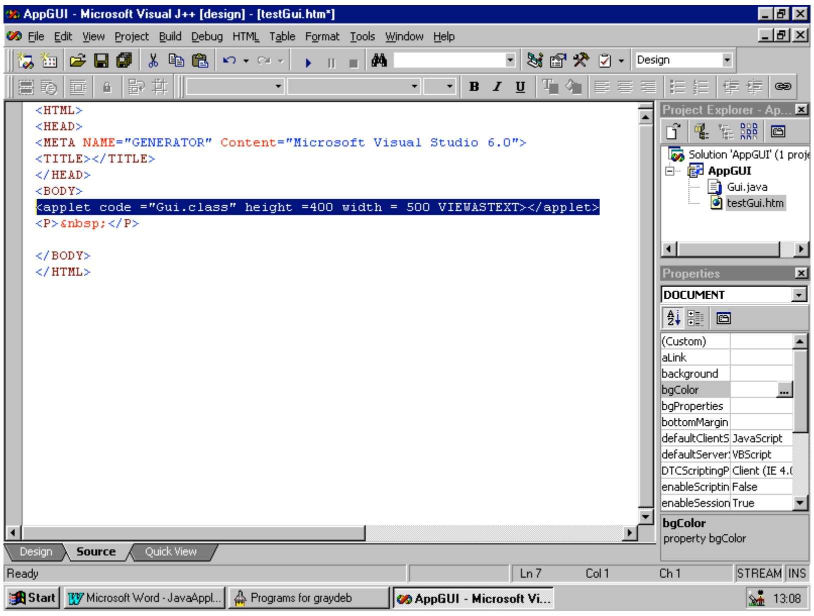


3. Add a TextField to allow the user to type in the font size.

4. All the code so far has been run in applet viewer.   Applets will often be run on the web using a browser. The HTML code to run an applet is just another tag.

```
<applet code ="Gui.class" height =400 width = 500 ></applet>
```

The Class name must be that of the Class that is being run. Height and width correspond to how much of the HTML page space is allocated to the applet. VJ will allow a blank web page to be added to an existing solution. The page has to be viewed in source to edit the code.



```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>
<BODY>
<applet code ="Gui.class" height =400 width = 500 VIEWASTEXT></applet>
<P>&nbsp;</P>

</BODY>
</HTML>
```

(Custom)	
aLink	
background	
bgColor	
bgProperties	
bottomMargin	
defaultClientS	JavaScript
defaultServer	VBScript
DTCScriptingP	Client (IE 4.(
enableScriptin	False
enableSession	True

You may have to change the solution properties to ensure that the web page loads and not the applet when running the solution.