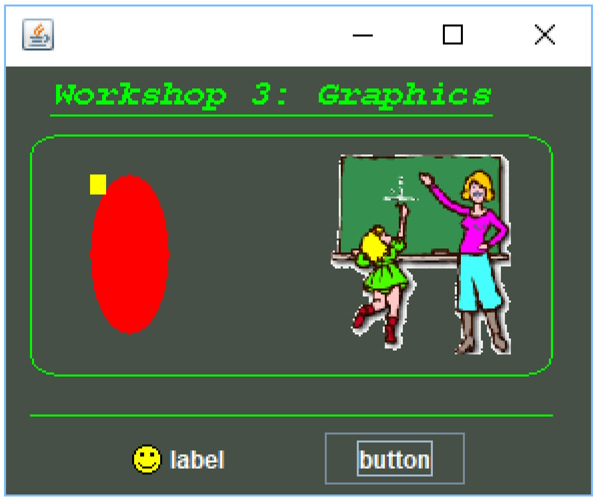
Internet Programming 300130

WORKSHOP-3

(5 Marks, due by Week 6)

EXERCISE 3.1 (5)

Write a Java program to produce the following GUI look.



You may start your class definition by such as

```
public class DrawGraphics extends JFrame { ... }
```

1. If you need to brush yourself on the relevant topics, you could do **3.2** below first.
2. The **JFrame** has the size of 300 by 250 pixels.
3. Check out the **java.awt.Graphics** class for the methods to do the drawing.
4. The title **Workshop 3: Graphics** should be drawn with the **MonoSpaced** font at size of 18 points. The title should be both *bold* and *italic*.
5. The method **getFontMetrics().stringWidth(string)** can return the width of a string. You can use this method to calculate the length of the line below the title.

6. The background colour should have the *RGB* values **70**, **80**, **70**. Check out the **Color** class and the **setBackground()** method for a **Component**.
7. Check out the usage of **ImageIcon** and **JLabel** classes. Place the image **smiley.gif** () along the label **label**.
8. Check out the usage of method **setToolTipText().** Set a tool tip "This is a label" for the label, and "This is a button" for the button.
9. The button and label may be added to a **JPanel** first, before being added eventually to the **JFrame** under the **BorderLayout** layout manager. Check out the method **setHgap()** of the **FlowLayout** class to see how we can keep the button and the label at a distance of **50** pixels.
10. When the GUI window is closed, the program should also exit.
11. The last part is for the painting of the image **educ1.gif** ( ). To do it, please check out the **java.awt.Image**, **java.awt.Graphics**, and **javax.swing.ImageIcon** classes. More specifically, you may try

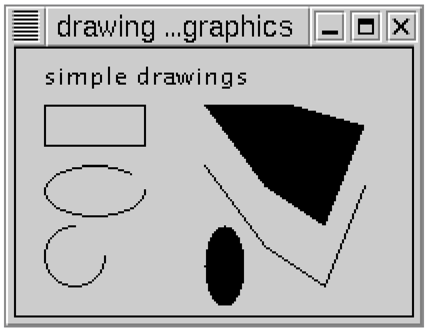
```
    Image image=new ImageIcon("educ1.gif").getImage();
    g.drawImage(image,170,75,90,100, this);
```

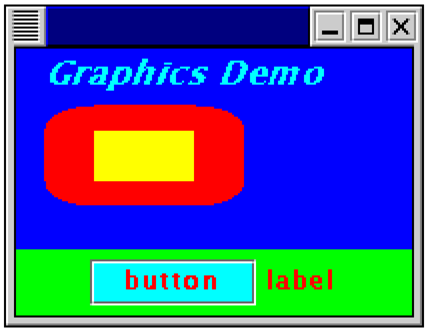
to draw the image.

Exercise 3.2 (Not to be marked)

A Quick Reminder

1. Some rudimentary drawings have already been given in the lectures for week 6, where we have explained how to draw





on some **JFrame**s. You may want to refresh yourselves on such basic drawing operations. The Java code **ShowGraphics.java** for generating the 2nd figure is quoted below again.

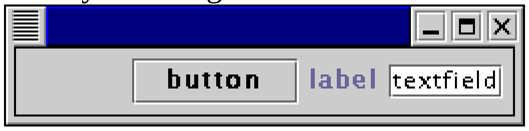
```
import javax.swing.*;
import java.awt.*;

public class ShowGraphics extends JFrame {

  public void paint(Graphics g) {
    // must use getContentPane(), not "this"
    getContentPane().setBackground(new
Color(0.0f,0.0f,1.0f));
    super.paint(g); // paint components

    // do some paintings
    g.setColor(Color.cyan);
    g.setFont(new Font("SansSerif",
              Font.BOLD+Font.ITALIC, 16));
    g.drawString("Graphics Demo",20,40);
    g.setColor(new Color(255,0,0));
    g.fillRoundRect(20, 50, 100, 50, 50,25);

    // it's "this" container!
    this.setBackground(Color.yellow);
    g.clearRect(45, 63, 50, 25);
  }

  public static void main(String[] args) {
    JFrame f=new ShowGraphics(); //used JFrame for variety
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(200,150);

    // back/foreground colors both active
    JButton b=new JButton("button");
    b.setBackground(Color.cyan);
    b.setForeground(Color.red);

    //label's background color taken over by its container
    JLabel l=new JLabel("label");
    l.setBackground(Color.cyan); // taken over later
    l.setForeground(Color.red); // active

    Container p=new JPanel(); // used as a container
    p.setBackground(Color.green);
    p.setForeground(Color.white); // taken over later
    p.setLayout(new FlowLayout());// default too
    p.add(b);
    p.add(l);

    f.getContentPane().add(BorderLayout.SOUTH,p);
    f.setVisible(true);

    f.repaint();  // needed for some platforms
  }
}
```

Beware the *component* you actually use when using such as *component***.setBackground()**.

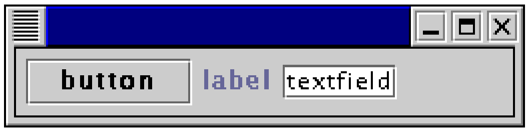
2. We have also shown how to make use of **LayoutMangers** such as **FlowLayout** and **BorderLayout**, including the following example

```
import javax.swing.*;
import java.awt.*;

public class FlowDemo extends JFrame {
  public FlowDemo() {
    Container c=getContentPane();
    FlowLayout lay=new
            FlowLayout(FlowLayout.RIGHT);
    c.setLayout(lay);
    c.add(new JButton("button"));
    c.add(new JLabel("label"));
    c.add(new JTextField("textfield"));
    setSize(250,50); // not c.setSize()
    setVisible(true);
    try { Thread.sleep(3000); } catch(Exception e){}
    lay.setAlignment(FlowLayout.LEFT);
    lay.layoutContainer(c);
  }

  public static void main(String[] args) {
    new FlowDemo().setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
  }
}
```

The layout changes from



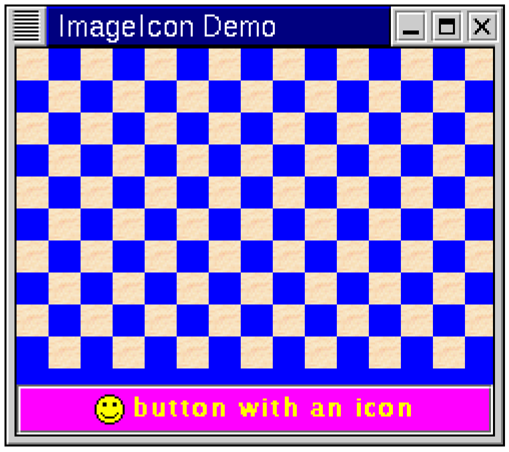
to



after 3 seconds.

Read, compile and run the program.

3. The use of **ImageIcon** to plot images has been addressed in the class already. For instance, the GUI below



can be produced by the following Java code.

```
import javax.swing.*;
import java.awt.*;

class IconDemo extends JPanel {
  ImageIcon icon=new ImageIcon("space.gif");
  int h=icon.getIconHeight(),
      w=icon.getIconWidth();

  // do the painting on a JPanel
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.blue);
    for(int i=0; i<10; i++)
      for(int j=0; j<15; j++)
         if(((i+j)/2)*2==i+j)
            icon.paintIcon(this,g,j*w,i*h);
  }

  // make a JPanel and add a button
  public void display() {
    ImageIcon icon=new ImageIcon("smiley.gif");
    JButton b=new JButton("button with an icon",icon);
    setLayout(new BorderLayout());
    add(BorderLayout.SOUTH,b);
    b.setBackground(Color.magenta);
    b.setForeground(Color.yellow);
  }

  public static void main(String[] args) {
    JFrame f=new JFrame("ImageIcon Demo");
    IconDemo ic=new IconDemo();
    ic.display();
    f.getContentPane().add(ic);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(16*15,16*10+50); //gif images have 16x16
                               //pixels
    f.setVisible(true);
  }
}
```

Read, compile and run the program. Resize the frame to observe the effects.