

## Lecture 14

### Colors

Java permits the programmer to control colors and fonts. We start by demonstrating how to handle colors:

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

public class ShowColors extends JFrame
{
    public ShowColors()
    {
        super("using Colors");
        setSize(300, 250);
        setLocation(600,100);
        show();
    } // end public ShowColors()

    public void paint(Graphics g)
    {
        g.setColor( new Color(255,0,0));
        g.fillRect(25,25,100,50);           // corner at (25,25)
                                           // 100 pixels wide,50 high

        g.drawString("Current RGB: " +g.getColor(), 25,90);

        g.setColor(Color.blue);
        g.fillRect(25, 100, 50,100);      // corner at (25,100)
                                           // 50 pixels wide, 100 high
        g.drawString("Current RGB: " +g.getColor(), 25,215);
    } // end paint(Graphics g)

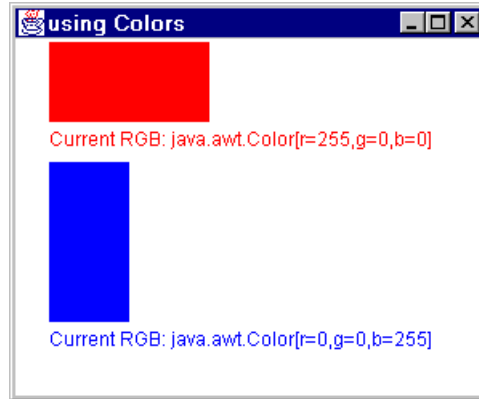
    public static void main(String args[])
    {
        ShowColors app = new ShowColors();

        app.setBackground(Color.white);

        app.addWindowListener(
            new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                    System.exit(0);
                }
            }
        );

    } // end main(String args[])
} // end class ShowColors
```

The output is:



Java has a routine called `JColorChooser` that provides information useful when you are trying to select a color:

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

public class ShowColors2 extends JFrame
{
    private JButton changeColor;
    private Color color = Color.lightGray;
    private Container c;

    public ShowColors2()
    {
        super(" Using JColorChooser");

        c = getContentPane();
        c.setLayout(new FlowLayout());

        changeColor = new JButton("Change Color");

        changeColor.addActionListener
        (
            new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    color = JColorChooser.showDialog(
                        ShowColors2.this, "Choose a color", color);

                    if (color == null) color = Color.lightGray;

                    c.setBackground(color);
                    c.repaint();
                }
            }
        );

        c.add(changeColor);
        setSize(400,130);
        setLocation(300,300);
        show();
    }
}

// end ShowColors2()
```

```

public static void main(String args[])
{
    ShowColors2 app = new ShowColors2();

    app.addWindowListener
    (
        new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        }
    );
} // end main(String args[])

} // end class ShowColors2

```

////////////////////////////////////

On **this** in:

```

color = JColorChooser.showDialog( ShowColors2.this, "Choose a color", color);

```

Reading the material on the right, **JColorChooser** is a class with the method **showDialog()**. The defining line of **showDialog()** is:

```

public static Color showDialog(Component component, String title,Color color)

```

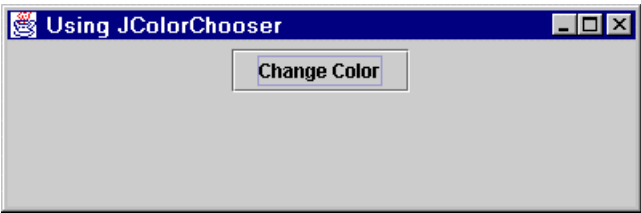
Thus **showDialog()** is a method that returns a color and has three parameters, **component**, **title**, and **color**.

The component in our specific case is **ShowColors2**, but the class being created is **ShowColors2**. The "this", enables the class to refer to itself.

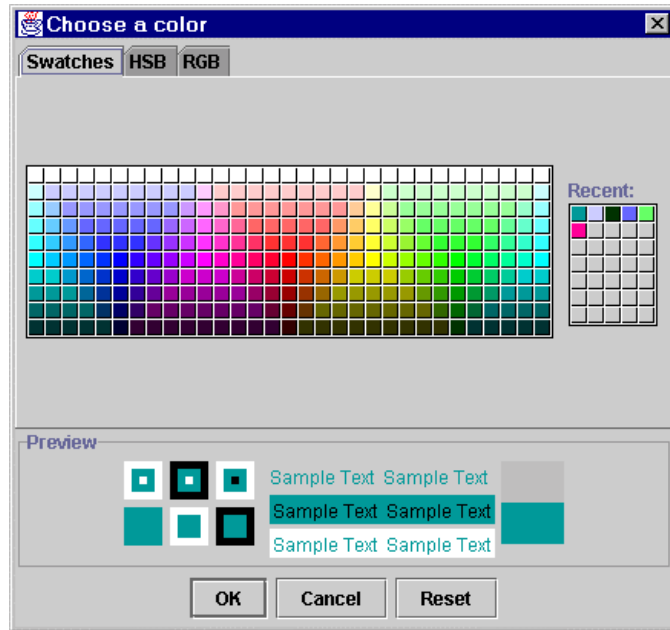
The entry **ShowColors2.this** implies that **this** is a variable of the class **ShowColors2**. It, **this**, is an implicit variable of all classes.

////////////////////////////////////

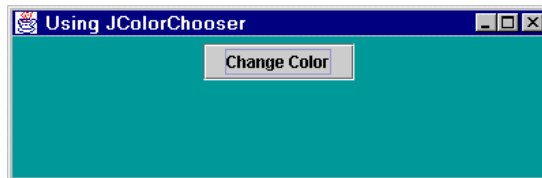
The output of **ShowColors2** opens with a window with a button:



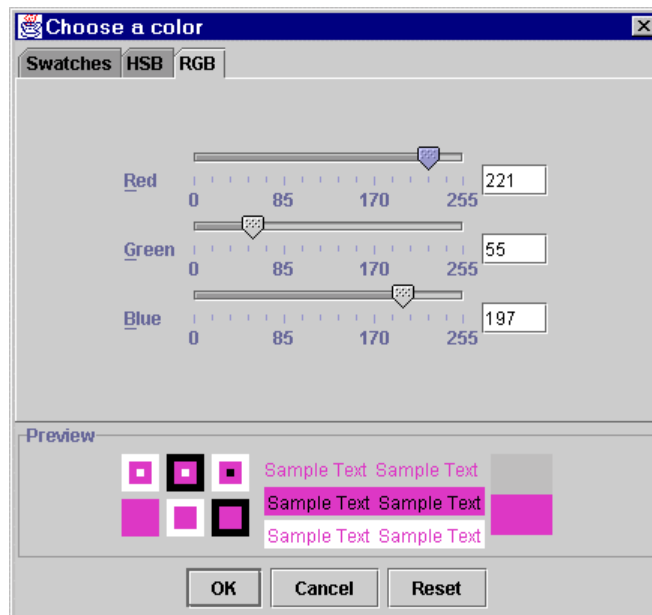
When you click on this button another window appears. As you click on the colors on the large grid a record of you choices is made on the small grid to the right:



When you click on the "OK" button above the original window appears, with its background painted in your last choice:



The larger screen's RGB option has sliders which give the correspondence between a color and its RGB numbers:



## Fonts

The programmer can also modify fonts and their styles:

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

public class Fonts extends JFrame
{
    public Fonts()
    {
        super("Fonts");
        setSize(300, 200);
        setLocation(300, 0);
        show();
    } // end public Fonts()

    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.setFont(new Font("Serif", Font.BOLD, 16));
        g.drawString("Serif, BOLD, 16 point", 20, 50);

        g.setColor(Color.blue);
        g.setFont(new Font("SansSerif", Font.ITALIC, 20));
        g.drawString("SansSerif, BOLD, 20 point", 20, 100);

        g.setColor(Color.black);
        g.setFont(new Font("SansSerif", Font.ITALIC+Font.BOLD, 10));
        g.drawString("SansSerif, BOLD, 10 point", 20, 150);

        g.setColor(new Color(125,125,0));
        g.setFont(new Font("Monospaced", Font.PLAIN, 24));
        g.drawString("Monospaced, Plain, 24 point", 20, 200);

    } // end paint(Graphics g)

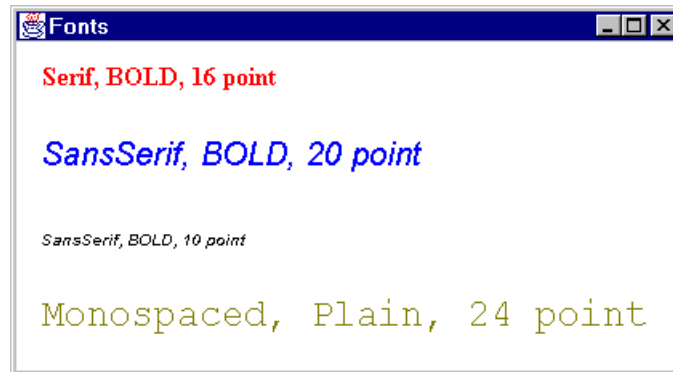
    public static void main(String args[])
    {
        Fonts app = new Fonts();

        app.setBackground(Color.white);

        app.addWindowListener
        (
            new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                    System.exit(0);
                }
            }
        );
    } // end main(String args[])

} // end class Fonts
```

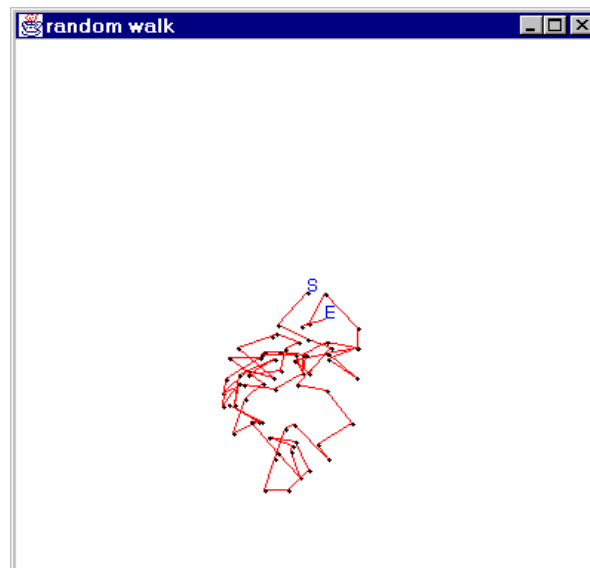
The output is:



### Lines, Circles

Java also has various routines for drawing lines, circular arcs, filling rectangles, filling ovals, etc. A complete listing can be found in chapter 11 of your text.

We will use the techniques for drawing straight lines and filling ovals to simulate a random walk. One simulation is shown below. The first window controls the number of walks (only one is shown) The second shows the walk. The starting point is marked by S, the ending point by E, and the black points mark the route of the random traveler.



There are two pieces of code a class `RandomWalk`, and the driver to test it.

```

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

public class RandomWalk extends JFrame
{
    int p = 70;
    int[] X = new int[p];
    int[] Y = new int[p];

    public RandomWalk(int a, int b)
    {
        super("random walk");
        setSize(400,400);
        setLocation(a,b);
        show();

        X[0] = 200; Y[0] = 200;

        for (int i = 0; i < p-1; i++)
        {
            X[i+1] = X[i] + (int)(50*(Math.random()-0.5));
            Y[i+1] = Y[i] + (int)(50*(Math.random()-0.5));
        }
    }

    // end RandomWalk(int a

    public void paint(Graphics g)
    {
        g.setColor(Color.blue);
        g.drawString("S", X[0],Y[0]);

        for (int i = 0; i < p-1; i++)
        {
            g.setColor(Color.red);
            g.drawLine(X[i], Y[i],X[i+1], Y[i+1]);
            g.setColor(Color.black);
            g.fillOval(X[i], Y[i], 3, 3);
        }
        g.setColor(Color.blue);
        g.drawString("E", X[p-1],Y[p-1]);

    }

    // end paint(Graphics g)

}

// end class RandomWalk

```

The driver for the walk is `TestRandomWalk`. It can be used to run two or three simulations at the same time. To do so, just remove the appropriate quote marks in the `main()` method.

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

public class TestRandomWalk extends JFrame
{
    TestRandomWalk()
    {
        super("Test random Walk");
        setSize(300,100);
        setLocation(100,10);
        show();
    }

    public void paint(Graphics g)
    {
        g.drawString("Driver" , 50,50);
    }

    public static void main(String args[])
    {
        TestRandomWalk trw = new TestRandomWalk();
        RandomWalk rw = new RandomWalk(100, 200);

        //RandomWalk rw2 = new RandomWalk(600, 20);
        //RandomWalk rw3 = new RandomWalk(600, 200);

        trw.setBackground(Color.white);
        rw.setBackground(Color.white);

        //rw2.setBackground(Color.white);
        //rw3.setBackground(Color.white);

        trw.addWindowListener
        (
            new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                    System.exit(0);
                }
            }
        );
    }
}

} // end class TestRandomWalk
```

There is only one window listener, so you exit by clicking on the driver window.

