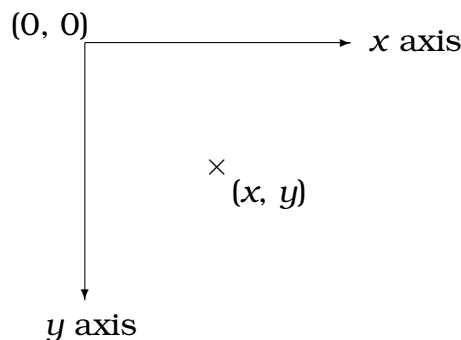


CS1Bh Lecture Note 12

Graphics

The Java language provides an extensive library of classes which allow the programmer to define and display graphical objects such as lines, rectangles, and ovals. In addition, the programmer can control the use of colour and the selection of fonts (lettering styles) which are used to print text. The classes related to graphical display which we will use are collected in Java's Abstract Windowing Toolkit (AWT), accessed via the package `java.awt`, and the popular Swing package, `javax.swing`.

Our graphical objects are arranged on a display called a `JFrame` (provided by the Swing package, and extending the `Frame` class provided by AWT) which is equipped with a two-dimensional co-ordinate system allowing placement of graphical objects at particular locations. The origin for the co-ordinate system is in the top-left corner of the frame. The x -coordinate increases as we go east and the y -coordinate increases as we go south.



12.1 A graphics context

In order to display graphical objects within a frame, we make use of a graphics *context*. A context is accessed via an object of the `Graphics` class. A graphics context is supplied for us when we implement a `paint()` method for our frame. The `paint()` method is defined at the level of `Component`, the superclass of many of the classes in the `java.awt` package. In particular the `JFrame` class is a subclass of `Component` so it inherits the `paint()` method from `Component`.

Given a `Graphics` object `g`, the `paint()` method invokes methods on this object to achieve its effects. The methods which would be used to perform simple graphical

operations include the following.

`void drawLine (int x1, int y1, int x2, int y2)` This method draws a line between $(x1, y1)$ and $(x2, y2)$.

`void drawRect (int x, int y, int width, int height)` This method will draw a rectangle of the specified height and width, situating the top-left corner of the rectangle at (x, y) . The rectangle is open (it is made up of four lines) so that the background colour of the frame shows through inside.

`void fillRect (int x, int y, int width, int height)` This method will draw a rectangle of the specified height and width, situating the top-left corner of the rectangle at (x, y) . The rectangle is filled (one block of solid colour).

`void clearRect (int x, int y, int width, int height)` This method draws a rectangle of the specified height and width, situating the top-left corner of the rectangle at (x, y) . The rectangle is filled using the background colour.

`void drawString(String s, int x, int y)` Prints the given string at (x, y) .

`void setColor (Color c)` Sets the colour for the next graphical operation (the next line, rectangle or text drawn). Note that American spelling is used—“color” instead of “colour”.

12.2 A simple example: random lines

As a simple example of the use of some of the above methods and the definition of a `paint()` method for a `JFrame`, we define a simple program which plots random lines of random colours within a 500 by 500 frame. The frame is constructed at line 8 by invoking the constructor method of the superclass of this object (the word **super** is used for this purpose). The string “Simple graphics in Java” is used to define the title for this window, as seen in the title bar which appears at the top. The size of the frame is set with the `setSize()` method and the frame is shown on the screen with the `show()` method. (A `hide()` method allows windows to be hidden from view when they are no longer needed.)

A method is defined to produce random integers in a given range. This later helps us to select random colours and draw lines between random positions. Java has pre-defined colours called `Color.red`, `Color.blue` and so forth, but it is possible to mix other ones by specifying the amount of red, green and blue which should be used. The number 0 represents none at all and the maximum is 255. Thus for example $(255, 0, 0)$ is red, $(255, 255, 0)$ is yellow, $(128, 128, 128)$ is gray, $(0, 0, 0)$ is black, and $(255, 255, 255)$ is white.

In the `paint()` method an infinite loop repeatedly selects a random colour by mixing random amounts of red, green and blue and then creates a new `Color` object with these values and then sets the colour of the graphics context `g` with the `setColor()` method (line 20). Then a line is drawn between a pair of randomly selected points (using the method `drawLine()`).

```
import java.awt.*; // 1
import java.math.*; // 2
import javax.swing.*; // 3
// 4

class SimpleGraphics extends JFrame { // 5
// 6

    public SimpleGraphics() { // 7
        super ("Simple graphics in Java"); // 8
        setSize (500, 500); // 9
        show(); // 10
    } // 11
// 12

    // A method to return random numbers between 0 and N // 13
    int rnd (int N) { // 14
        return (int) (N * Math.random()); // 15
    } // 16
// 17

    public void paint (Graphics g) { // 18
        while (true) { // 19
            g.setColor (new Color(rnd(255), rnd(255), rnd(255))); // 20
            g.drawLine (rnd(500), rnd(500), rnd(500), rnd(500)); // 21
        } // 22
    } // 23
// 24

    public static void main (String[] args) { // 25
        SimpleGraphics s = new SimpleGraphics(); // 26
        s.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 27
    } // 28
// 29
} // 30
```

12.3 A second example: fonts

A *font* determines the appearance of text when it is printed or displayed on the screen. The lecture note that you are holding in your hands now is printed in the Bookman font. The program fragments which appear are printed in Courier font.

Partly, fonts are chosen in the hope that they will produce an attractive (or at least readable) display when combined. But when we come to write programs which display text in a graphics area then we also need to consider matters such as the size of the text. Will the text be too large to fit into the area? If we want to have several lines of text then how much space should be leave between one line and the line which appears beneath it?

The Java programming language gives the programmer control over these and other matters by allowing them to select the font which will be used to format any text which they write to a graphics area on-screen. In the example program below we select

various versions of the Times Roman font to display text on the screen. We set two further attributes of the font, its *style* and its *point size*.

The font style determines the thickness of the slope of letters. Text in **bold style** is thicker than text in plain style. Text in *italic style* slopes more than plain text. If we must, we can combine bold and italic to get **bold italic**, which looks gross. In Java we code our request for bold and italic by using the *bitwise or* operator “|” as on line 27 below. The point size determines the height of the letters (72 points make an inch).

```
import java.awt.*; // 1
import javax.swing.*; // 2

public class FontExample extends JFrame { // 3
    // 4
    public FontExample() { // 5
        // 6
        super ("An example of fonts in Java"); // 7
        setBackground(Color.white); // 8
        setSize(500, 500); // 9
        show(); // 10
    } // 11
    // 12
    public void paint (Graphics g) { // 13
        Font plain = new Font("TimesRoman", Font.PLAIN, 18); // 14
        g.setFont(plain); // 15
        g.drawString("This is Times, plain.", 100, 100); // 16
        // 17
        Font bold = new Font("TimesRoman", Font.BOLD, 18); // 18
        g.setFont(bold); // 19
        g.drawString("This is Times, bold.", 100, 200); // 20
        // 21
        Font italic = new Font("TimesRoman", Font.ITALIC, 18); // 22
        g.setFont(italic); // 23
        g.drawString("This is Times, italic.", 100, 300); // 24
        // 25
        Font boldItalic = new Font("TimesRoman", // 26
            Font.BOLD|Font.ITALIC, 18); // 27
        g.setFont(boldItalic); // 28
        g.drawString("This is Times, bold italic.", 100, 400); // 29
    } // 30
    // 31
    public static void main (String[] args) { // 32
        FontExample f = new FontExample(); // 33
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 34
    } // 35
} // 36
```

12.4 More Programs

Here are two more example graphics programs which I will explain during the lecture....

```
import java.awt.*; // 1
import java.math.*; // 2
import javax.swing.*; // 3
// 4
class Smiley extends JFrame { // 5
// 6
    public Smiley() { // 7
        super ("Love 'n Peace"); // 8
        setSize (600, 600); // 9
        show(); // 10
    } // 11
// 12
    public void paint (Graphics g) { // 13
        g.setColor(Color.yellow); // 14
        g.fillOval(100,100,400,400); // 15
        g.setColor(Color.black); // 16
        g.fillArc(233, 350, 132, 50, 0, -180); // 17
        g.fillOval(200, 233, 40, 40); // 18
        g.fillOval(360, 233, 40, 40); // 19
        g.setColor(Color.yellow); // 20
        g.fillArc(233, 340, 132, 50, 0, -180); // 21
    } // 22
// 23
// 24
    public static void main (String[] args) { // 25
        Smiley s = new Smiley(); // 26
        s.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 27
    } // 28
// 29
} // 30
```

```

import java.awt.*; // 1
import javax.swing.*; // 2
// 3
class Smiley2 extends JFrame { // 4
// 5
    public Smiley2(int w, int h) { // 6
        super ("Love 'n Peace"); // 7
        setSize (w, h); // 8
        show(); // 9
    } // 10
// 11
    public void drawSmiley (Graphics g, int x, int y, int r) { // 12
        g.setColor(Color.yellow); // 13
        g.fillOval(x-r, y-r,2*r,2*r); // 14
        g.setColor(Color.black); // 15
        g.fillArc(x-r/3, y+r/4, 2*r/3, r/4, 0, -180); // 16
        g.fillOval(x-r/2, y-r/3, r/5, r/5); // 17
        g.fillOval(x+3*r/10, y-r/3, r/5, r/5); // 18
        g.setColor(Color.yellow); // 19
        g.fillArc(x-r/3, y+r/5, 2*r/3, r/4, 0, -180); // 20
    } // 21
// 22
    public void paint (Graphics g) { // 23
        int i; // 24
// 25
        final int radius = 25; // 26
// 27
        for (i=1; i<5; i++) { // 28
            drawSmiley(g, (i*i+2)*radius, (i*i+2)*radius, i*radius); // 29
        } // 30
    } // 31
// 32
    public static void main (String[] args) { // 33
        Smiley2 s = new Smiley2(600, 600); // 34
        s.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 35
    } // 36
// 37
} // 38

```

Murray Cole, 17th February 2003.