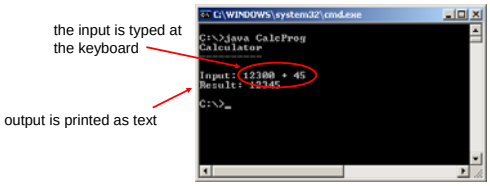
Lecture 21

Creating a window

Chapter 16
Graphical User Interfaces

Text vs. GUI

So far, the Java programs we have written have been text based



The *programmer* determines when input is required from the user

Text vs. GUI

Many applications present the user with a nice graphical interface

The *user* determines when to provide input to the program

Creating a window

How can we write a Java program that creates a window?

We are going to organise our code into three classes:

- **`MyApp`**
- **`MyJFrame`**
- **`MyJPanel`**

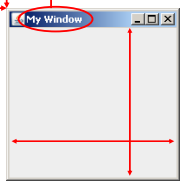
Why the three classes?

Creating a window

`MyApp`
the *starting point* for the program
creates an object of type MyJFrame:

```
new MyJFrame("My Window", 100, 100, 200, 200);
```

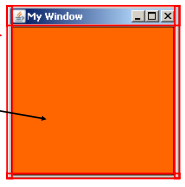
across, down width, height



Creating a window

`MyJFrame`
the main window with a border and a title bar

`MyJPanel`
the actual content of the window



The MyJPanel class

The MyJPanel class is defined in a similar way to other classes that we have defined:

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

public class MyJPanel extends JPanel {

}
```

there are several import statements

and this basically means that our class, MyJPanel, is a special type of JPanel object (which is defined by a standard Java class)

We will often refer to this as simply the "JPanel class"

The JPanel class

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

public class MyJPanel extends JPanel {

      public MyJPanel() {

      }

      public void paintComponent(Graphics g){
            super.paintComponent(g);


      }
}
```

this is the constructor method for this class – it is called just once, when the JPanel is first created

*this method defines what will be **drawn** in the window – it is called **automatically***

The paintComponent() method

The paintComponent() method is called *automatically*. It is passed a parameter, which is of type Graphics.

```
public void paintComponent(Graphics g){
      super.paintComponent(g);


}
```

We can make shapes appear in the window by calling instance methods on this Graphics object.

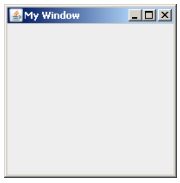
The paintComponent() method

For example:

```
public void paintComponent(Graphics g){
      super.paintComponent(g);




}
```



The paintComponent() method

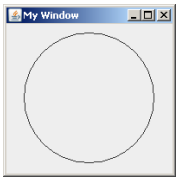
For example:

```
public void paintComponent(Graphics g){
      super.paintComponent(g);

      g.drawOval(20, 10, 150, 150);




}
```



The paintComponent() method

For example:

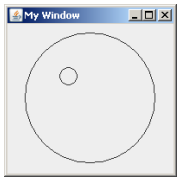
```
public void paintComponent(Graphics g){
      super.paintComponent(g);

      g.drawOval(20, 10, 150, 150);

      g.drawOval(60, 50, 20, 20);




}
```



The paintComponent() method

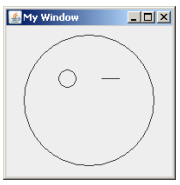
For example:

```
public void paintComponent(Graphics g){
       super.paintComponent(g);

       g.drawOval(20, 10, 150, 150);

       g.drawOval(60, 50, 20, 20);
       g.drawLine(110, 60, 130, 60);


}
```



The paintComponent() method

For example:

```
public void paintComponent(Graphics g){
       super.paintComponent(g);

       g.drawOval(20, 10, 150, 150);

       g.drawOval(60, 50, 20, 20);
       g.drawLine(110, 60, 130, 60);

       g.drawRect(55, 110, 80, 10);

}
```

The paintComponent() method

For example:

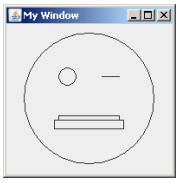
```
public void paintComponent(Graphics g){
       super.paintComponent(g);

       g.drawOval(20, 10, 150, 150);

       g.drawOval(60, 50, 20, 20);
       g.drawLine(110, 60, 130, 60);

       g.drawRect(55, 110, 80, 10);
       g.drawRect(60, 105, 70, 5);
}
```

What do the numbers mean in these method calls?



The coordinate system

Every pixel can be referred to by its horizontal and vertical position.

The top, leftmost pixel in the JPanel is at position (0,0)

The coordinate system

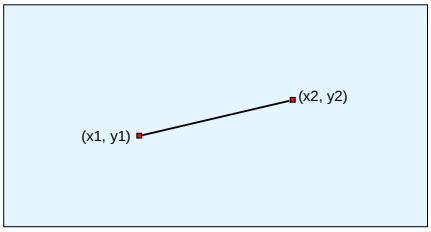
The coordinates are specified as:

(number of pixels across from the left , number of pixels down from the top)

Graphics instance methods

To draw a line between two points:

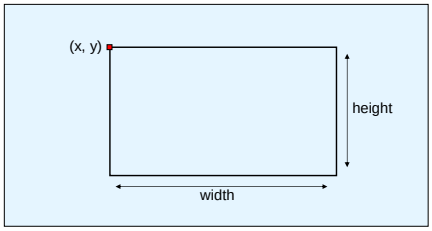
```
g.drawLine(int x1, int y1, int x2, int y2)
```



Graphics instance methods

To draw the **outline** of a rectangle:

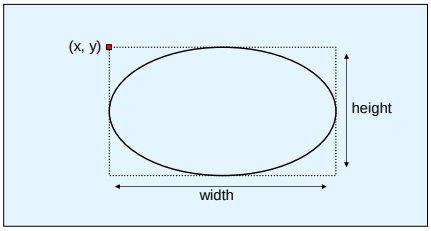
```
g.drawRect(int x, int y, int width, int height)
```



Graphics instance methods

To draw the **outline** of an oval:

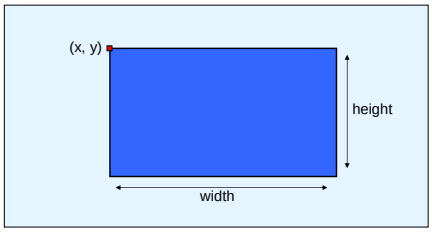
```
g.drawOval(int x, int y, int width, int height)
```



Graphics instance methods

To draw a **filled** rectangle:

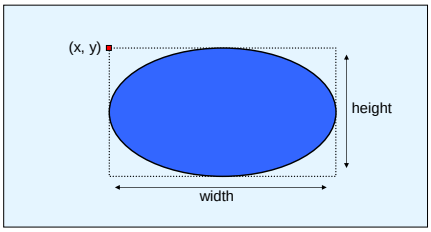
```
g.fillRect(int x, int y, int width, int height)
```



Graphics instance methods

To draw a **filled** oval:

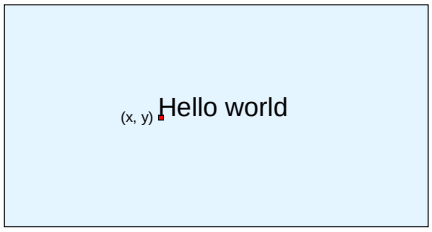
```
g.drawOval(int x, int y, int width, int height)
```



Graphics instance methods

To display text in the drawing window:

```
g.drawString(String text, int x, int y)
```



Graphics instance methods

To change the drawing colour

```
g.setColor(Color c)
```

There are 13 predefined colours:

Colours

And you can mix your own by creating an object of type Color:

```
new Color(int red, int green, int blue)
```

where the parameters should be between 0 and 255.
For example:

```
public void paintComponent(Graphics g){
       super.paintComponent(g);

       g.setColor(new Color(33, 178, 90));
       g.fillOval(20, 10, 150, 150);
}
```

Order

Shapes appear in the *order* that you draw them

```
g.setColor(Color.green);
g.fillOval(40, 60, 80, 80);

g.setColor(Color.blue);
g.fillOval(80, 60, 80, 80);
```

```
g.setColor(Color.blue);
g.fillOval(80, 60, 80, 80);

g.setColor(Color.green);
g.fillOval(40, 60, 80, 80);
```

The paintComponent() method

When does the paintComponent() method get called?

- when the window is first created and displayed
- when the window is resized by the user

This is the first of several methods we will encounter which is called *automatically*