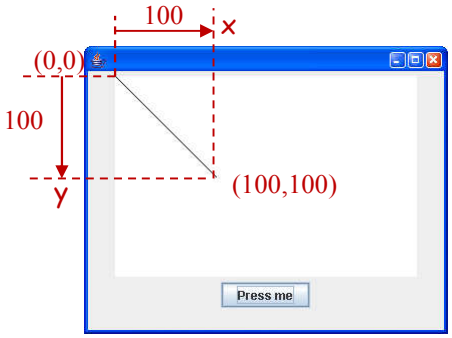
Drawing on the screen - simple graphics

- In this section:
  - A simple application using graphics ("DrawExample")
  - A more complex application using graphics ("SomeShapes")
  - Other graphics facilities

A simple graphics application (JFS, p23)

- First we design its appearance:



Demo
DrawExample

Coordinates are (x,y)

In terms of *pixels*

The *line* runs from pixel coordinate (0,0) to pixel coordinate (100,100)

- Now its *behaviour*:
  - Line drawn in drawing panel when the button is pressed
- This is an application with a permanent window
- New features: drawing, a button, reacting to a button press

The Java source code (JFS, p23,24) Slide 1/3

New features

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

public class DrawExample extends JFrame
    implements ActionListener {

    private JButton button;
    private JPanel panel;

    public static void main(String[] args) {
        DrawExample frame = new DrawExample();
        frame.setSize(400, 300);
        frame.createGUI();
        frame.setVisible(true);
    }
```


Slide 2/3

```
    private void createGUI() {
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container window = getContentPane();
        window.setLayout(new FlowLayout() );

        panel = new JPanel();
        panel.setPreferredSize(new
                               Dimension(300, 200));
        panel.setBackground(Color.white);
        window.add(panel);

        button = new JButton("Press me");
        window.add(button);
        button.addActionListener(this);

    }
```

Slide 3/3

```
    public void actionPerformed(ActionEvent event) {
        Graphics paper = panel.getGraphics();
        paper.drawLine(0, 0, 100, 100);
    }

}
```

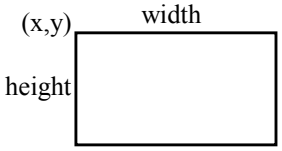
Notes:

- `createGui` now also
  - Makes a *panel* for drawing on, and adds it to the display
  - Makes a *button* for clicking, and adds it to the display
- We have added another new method: `actionPerformed`
  - This is a special, Java defined method
  - It is *called automatically* when we press the button
  - In this example `actionPerformed` contains drawing actions (in later examples it will do other things)

Analysing `paper.drawLine(0, 0, 100, 100);`

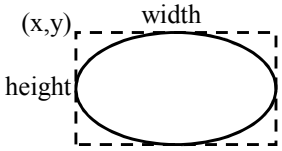
- `paper` specifies the screen area to draw in (the *panel*)
- `drawLine` is a graphics "method" from one of the AWT libraries – a predefined action
- `paper.drawLine(0, 0, 100, 100);` is a "call" of the `drawLine` method:
  - "Now ask `paper` to do a `drawLine` action on itself"
- In the call we specify *details* of the line to be drawn:
  - If the line is to run from coordinate (x1,y1) to coordinate (x2,y2) then in the call we must have `(x1,y1,x2,y2)`
  - These details are called the "parameters"
  - The parameters must be *in the correct order*,
  - and *of the correct type* (other methods may need, e.g., text, such as `"Hello"`)

Some other Graphics methods (JFS, p27)




```
paper.drawRect(x,y,width,height);
```
(also `fillRect`)

squares?




```
paper.drawOval(x,y,width,height);
```
(also `fillOval`)

circles?

- All of these draw or fill using the "current colour"
  - Initially black

Changing the drawing colour (JFS, p28)

- There are colour names like:

  `Color.black` `Color.blue` `Color.red`
  - Note the upper/lower case and the spelling!
  - The colours can also be in upper case: `Color.RED`
- The *background colour* of the drawing panel can be changed with a call of the library method:

  `panel.setBackground(` colour name `);`
- During drawing, the current *drawing/filling* colour can be changed by:

  `paper.setColor(` colour name `);`
  - In a *sequence* of drawing method calls, items drawn *after* the `setColor` have the new colour
  - `setColor` is rather like "change pen"
- Which brings us to *sequences...*

Sequences of *statements*

- In programming language terminology, instructions are often called *statements*
- Methods, such as `actionPerformed`, may contain a *sequence* of statements
  - **They are carried out in precisely the order given**
- When drawing, the order of drawing actions may be important if items *overlap*:
  - Items drawn later are "on top", or "in front" - "nearer to the viewer"
  - Only the items drawn *after* a `setColor` (and before the next one) are affected by the `setColor`

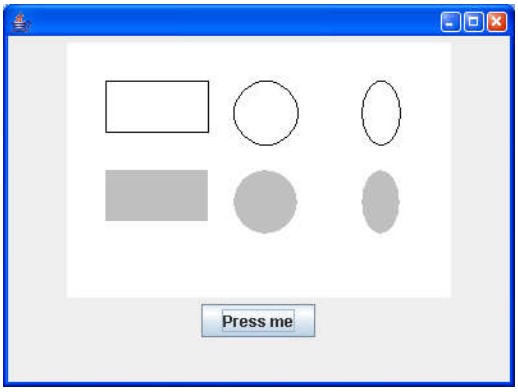
Sequence example: `SomeShapes`

- Just looking at the `actionPerformed` method:

```
public void actionPerformed(ActionEvent event) {
    Graphics paper = panel.getGraphics();
    paper.drawRect(30, 30, 80, 40);
    paper.drawOval(130, 30, 50, 50);
    paper.drawOval(230, 30, 30, 50);
    paper.setColor(Color.lightGray);
    paper.fillRect(30, 100, 80, 40);
    paper.fillOval(130, 100, 50, 50);
    paper.fillOval(230, 100, 30, 50);
}
```

- Note: Each *statement* is a *method call*, and each is followed by a `;`

- `SomeShapes` displays this image on the screen:



Demo
SomeShapes

Demo
SomeShapes
(slow)

End of Section