G51PRG:
Introduction to Programming
Second semester
Applets and graphics

Natasha Alechina
School of Computer Science & IT
nza@cs.nott.ac.uk

Previous two lectures

- AWT and Swing
- Creating components and putting them in a window (layout managers)
- Event handling
  - which components generate which events (buttons - ActionEvents, windows - WindowEvents…)
  - how to write an event listener (implement ActionListener or WindowListener…)
  - how to register it with the component which generates events (**`button.addActionListener(listener)`**)

Event Listeners

- Button pressed: **`ActionListener`**; **`actionPerformed(ActionEvent e)`** method
- Enter pressed in a text field: **`ActionListener`**
- Window closed, resized, etc. : **`WindowListener`**; one of the methods: **`windowClosing(WindowEvent e)`** method.
- Text changed in a text field: **`TextListener; textValueChanged(TextEvent e)`**

Event Listeners

- Mouse clicks, releases and mouse entering/exiting component: **`MouseListener`**; one of the methods: **`mousePressed(MouseEvent e). MouseEvents`** have **`getX()`** and **`getY()`** methods.
- Mouse movement (mouse dragged, mouse moved): **`MouseMotionListener`**; e.g. **`mouseDragged(MouseEvent e)`**
- Keyboard event (key stroke): **`KeyListener;`** one of the methods: **`keyPressed(KeyEvent e). KeyEvent`** object has **`getKeyChar()`** method.
- And about 50 more: see subinterfaces of **`EventListener`**.

Plan of the lecture

- Graphics (painting and animation) in Java
- Applets

Painting

- Each Component has **`paint()`** method.
- For AWT component, that's the method you overwrite if you need to change the component's appearance.
- For Swing components, you overwrite **`paintComponent()`** method. **`paint()`** in Swing calls **`paintComponent()`** but also does lots of other work (e.g. double buffering). Don't overwrite it.
- If you just need a particular look for a button etc., chances are you can do it in Swing by using additional functionality of Swing components (images, borders, look and feel…).

`Graphics`

`Graphics` class provides methods for drawing and using different fonts and colours. It is an abstract class. Subclassed for different platforms/graphics toolkits.

A **`Graphics`** object encapsulates the state information needed for painting a particular component on the screen: such as current colour, font, translation origin etc.

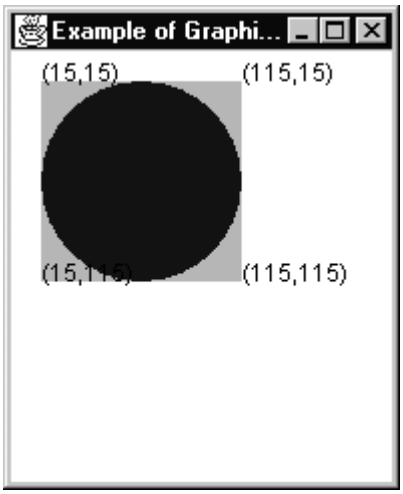
`paint(Graphics g)` and **`paintComponent(Graphics g)`** take a **`Graphics`** object as a parameter.

Some Graphics methods

- Methods to set colours and fonts.
- Methods to draw things: **`drawArc()`**, **`drawLine()`**, **`drawImage`** (takes an Image), **`drawRect()`**, **`drawOval()`**, **`drawPolygon()...`**
- Methods to fill figures: **`fillOval()`**, **`fillRect()`**, **...**

Coordinates are abstract and infinitely thin, the “pen” which the **`Graphics`** uses is pixel-sized. The origin is in the top left corner of the component where the drawing takes place.

Example



Outline of the example program

- Import **`javax.swing.*`** for the components, **`java.awt.Graphics`** and **`java.awt.Color`** for drawing, **`java.awt.event.*`** if going to handle events.
- Create a frame (just to hold the picture)
- Overwrite **`JPanel`** or some other **`Component;`** write a new **`paintComponent(Graphics g)`** method for it. All drawing is done in that method.
- Add the panel to the frame.

Painting the JPanel

```
public void paintComponent(Graphics g){
  super.paintComponent(g);
  g.setColor(Color.green);
  g.fillRect(15,15,100,100);
  g.setColor(Color.blue);
  g.fillOval(15, 15, 100,100);
  g.setColor(Color.black);
  g.drawString("(15,15)", 15,15);
  g.drawString("(15,115)", 15,115);
  g.drawString("(115,15)", 115,15);
  g.drawString("(115,115)", 115,115);
}
```


The whole extended class

```
class GraphicsPanel extends JPanel {

  GraphicsPanel(){
     this.setBackground(Color.white);
  }

  public void paintComponent(Graphics g){
    . . .
  }
}
```

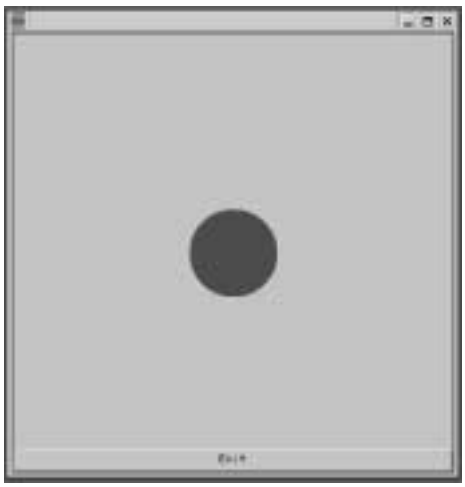
The main class

```
public class Example extends JFrame {
   GraphicsPanel panel = new GraphicsPanel();
   public Example() {
      super("Example of Graphics Methods");
      this.getContentPane().add(panel);
      this.setSize(200,150);
      this.setVisible(true);
   }
   public static void main(String[] args){
      new Example();
   }
}
```

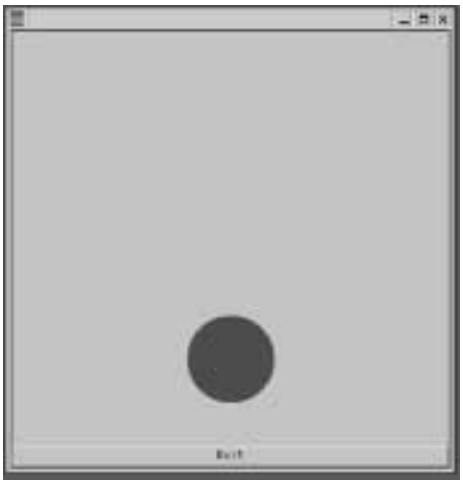

Animation

- Animation loop:
  - draw frame
  - wait a fixed amount of time
  - increment the frame number
- Simplest case: alternating between two frames

BouncingBall



BouncingBall



How to write an animation loop

- Good old days (AWT): somewhere, e.g. in the main(), have a loop which calls Thread.sleep() for a given number of milliseconds, then changes the animation frame and calls repaint().
- Swing: graphics can only be changed from the event thread. So we need to generate some kind of events at fixed intervals of time, in response to which the animation frame is changed and the component told to repaint itself.
- **`javax.swing.Timer`** class does just that.

Example of using Timer

```
class BBPanel extends JPanel implements
  ActionListener {
   int frameNumber = 0;

   BBPanel(int delay) {
      Timer timer = new Timer(delay, this);
      timer.start();
   }
   public void actionPerformed(ActionEvent e) {
      frameNumber++;
      repaint();
   }
```

Bouncing ball painting

```
public void paintComponent(Graphics g){
  super.paintComponent(g);
  setBackground(Color.white);
  g.setColor(Color.red);
  Dimension d = getSize();
  if (frameNumber % 2 == 1) {
    g.fillOval(0,0,d.width/2,d.height/2);
  } else {
    g.fillOval(0,d.height/2,d.width/2,
                              d.height/2);
  }
```


Applets

- What are Applets
- How do they work
- How to create one
- What one can do with applets

What are applets

- All Java programs we wrote so far are referred to as applications.
- Applets are a different kind of a Java program. They are designed to run in a browser and what they can do is restricted by the browser's security manager.
- Applets don's have a **`main()`** method (although you can write one and make the same program runnable both as an applet and as an application).
- Should implement at least one of **`init(), start()`** or **`paint()`** methods.

What are applets continued

- **`Applet`** and **`JApplet`** classes are part of **`java.awt`** and **`javax.swing`**, so they are relatives of frames and panels
- **`java.awt.Applet`** extends **`java.awt.Panel`** class and **`javax.swing.JApplet`** extends **`Applet`**.

Applet HTML tag

Applets are embedded in html pages using an applet tag:

```
<APPLET CODE="HelloWorld.class" WIDTH=500 HEIGHT=25>
</APPLET>
```

Here, HelloWorld.class should be in the same directory as the html page (you can also give a path to the class file).

HelloWorld applet code (AWT!)

```
import java.applet.Applet;
import java.awt.*;
public class HelloWorld extends Applet {
  public void paint(Graphics g) {
    Font font = new Font ("SansSerif",
                          Font.BOLD, 26);
    g.setFont(font);
    g.drawString("Hello world!", 50, 25);
  }
}
```

What the Browser does

On encountering the APPLET tag, the browser

- downloads the class file
- creates an object of that class
- gives it a region on the web page to control
- invokes its **`init()`** method
- invokes its **`start()`** method
- invokes its **`paint()`** method

Life Cycle of an Applet

An applet can

- initialise: **`init()`**
- start running: **`start()`**
- stop running: **`stop()`**
- perform final cleanup, free up resources: **`destroy()`**

All these methods are called by the application (browser, appletviewer) which is running the applet.

Security Restrictions on Applets

Those differ from browser to browser, but usually applets are forbidden to:

- load non-Java code
- read or write files on the host where it is running
- make network connections except to the host that it came from
- start any program on the host where it is running
- read certain system properties

Windows that an applet brings up look different than windows that an application brings up.

Why example applets are in AWT

- I gave AWT code for the example applets to make sure that they run in older browsers.
- You can also read Sun tutorial on writing GUI (and Applets) in Swing.

Summary

- Applets extend the Applet class (from java.applet package)
- Swing applets similar to awt applets.
- Applets don't have the main() method. Instead they must implement paint(), init() or start().
- They can be run in a browser or applet viewer.
- They should be embedded in HTML using the APPLET tag.
- Applets inherit the ability to draw and handle events from Component, the ability to have other components inside and use layout manager from Container.