Graphical User Interface Programming

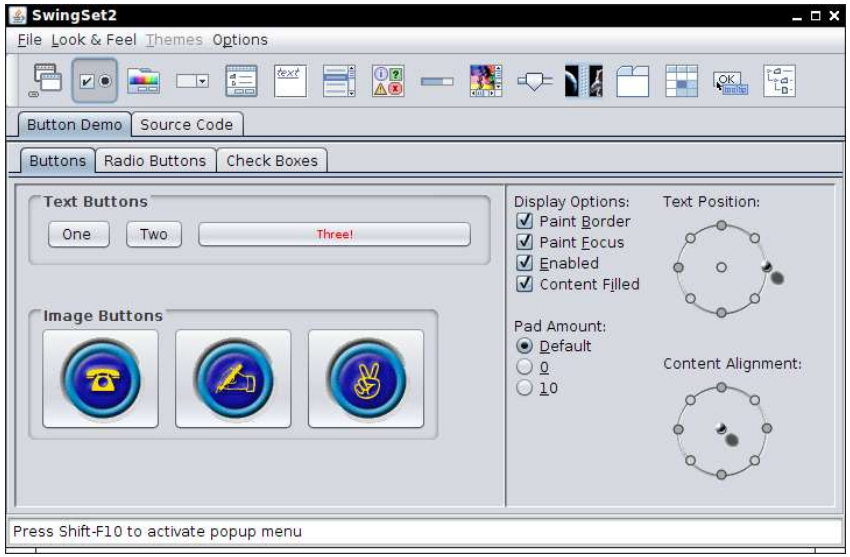
Part 1 – Basics of GUI

Chapter 10, Core Java, Volume I

Contents

Introducing Java GUI Frameworks

- **Java 1.0 had the Abstract Window Toolkit (AWT):**
  - Peer-based GUI toolkit (lowest common denominator approach)
  - Worked on Windows, Mac OS, Linux and so on.
  - Inconsistent in look and feel, behavior.
- **Java 1.2 put a new UI toolkit, called Swing:**
  - Paints almost all UI widgets from scratch (light-weight components)
  - Pluggable look and feel can mimic the host platform or provide a cross-platform look.
  - Works on the foundations of AWT (e.g. AWT event model)
- **Other graphical user interface toolkits:**
  - Java FX
  - SWT (for eclipse)
  - Android UI Framework



Displaying Frames

- Frame = top-level window with a title bar (**Frame** class in AWT).

- JFrame is the Swing version of frame; one of the few Swing components that is not painted by Swing—window decorations come from host OS.

- Extend **JFrame** class:

```
class SimpleFrame extends JFrame
{
      public SimpleFrame() { setSize(300, 200); }
}
```

- Create a frame, set the properties and show the frame:

```
JFrame frame = new SimpleFrame();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
```

Example : SimpleFrame (Listing 10.1)

```
package simpleFrame;

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

public class SimpleFrameTest
{
   public static void main(String[] args)
   {
        SimpleFrame frame = new SimpleFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
   }
}
```

Example : SimpleFrame (Listing 10.1)

```
class SimpleFrame extends JFrame
{
  private static final int DEFAULT_WIDTH = 300;
  private static final int DEFAULT_HEIGHT = 200;

  public SimpleFrame()
  {
    setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
  }
}
```

Event Dispatch Thread

- The event dispatch thread(EDT) is a special thread on which Swing event handling code runs. (e.g. `actionPerformed` methods in event handlers)
- Most code that invokes Swing methods should be run on this EDT.
- This is necessary because most Swing object methods are not "thread safe":
- Invoking them from multiple threads risks thread interference or memory consistency errors.

- Revisit the `main` method of `SimpleFrameTest` class:

```
EventQueue.invokeLater(
    () -> {
        SimpleFrame frame = new SimpleFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
);
```

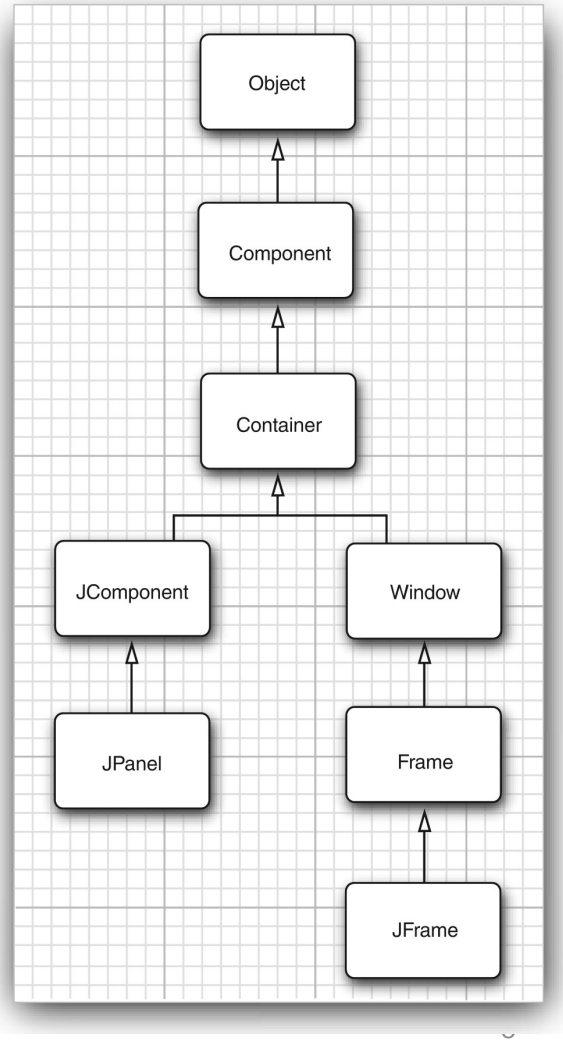
Frame Properties and Methods

- JFrame inherits properties and methods from it's the various superclasses.
- Position and appearance:
  - setLocation/getLocation
  - setSize/getSize
  - setBounds/getBounds
  - setIconImage/getIconImage
  - setTitel/getTitle
  - setResizable/isResizable
  - setVisible/isVisible
  - ...
- To set the size of a frame using screen dimension:

```
Toolkit kit = Toolkit.getDefaultToolkit();
Dimension screenSize = kit.getScreenSize();
```

- Alternatively, fill frame with UI elements, and then call:

```
frame.pack(); // use preferred sizes of components
```



Example : SizedFrame

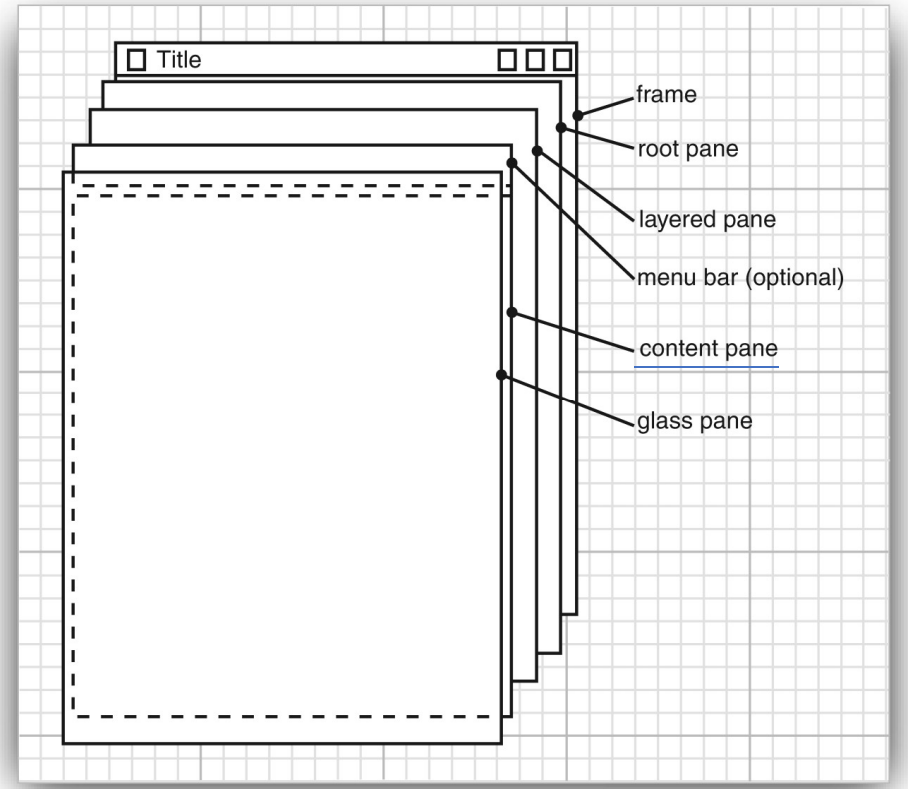
```
package sizedFrame;

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

public class SizedFrameTest
{
   public static void main(String[] args)
   {
      EventQueue.invokeLater(() ->
         {
            JFrame frame = new SizedFrame();
            frame.setTitle("SizedFrame");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
         });
   }
}
```

Example : SizedFrame

```
class SizedFrame extends JFrame
{
  public SizedFrame()
  {
    // get screen dimensions

    Toolkit kit = Toolkit.getDefaultToolkit();
    Dimension screenSize = kit.getScreenSize();
    int screenHeight = screenSize.height;
    int screenWidth = screenSize.width;

    // set frame width, height and let platform pick
    // screen location

    setSize(screenWidth / 2, screenHeight / 2);
    setLocationByPlatform(true);

    // set frame icon

    Image img = new ImageIcon("icon.gif").getImage();
    setIconImage(img);
  }
}
```

Drawing on a Component

- Draw onto a component, not directly onto a frame.



- Add one or more components to the frame:

```
frame.add(comp);  // placed into the content pane
```

- A component extends the JComponent class:

```
class MyComponent extends JComponent
{
      public void paintComponent(Graphics g)
      {
          code for drawing
      }
}
```

- Use the Graphics object for drawing:

```
g.drawString("Not a Hello World program", 75, 100);
```

- A component should tell how big it wants to be:

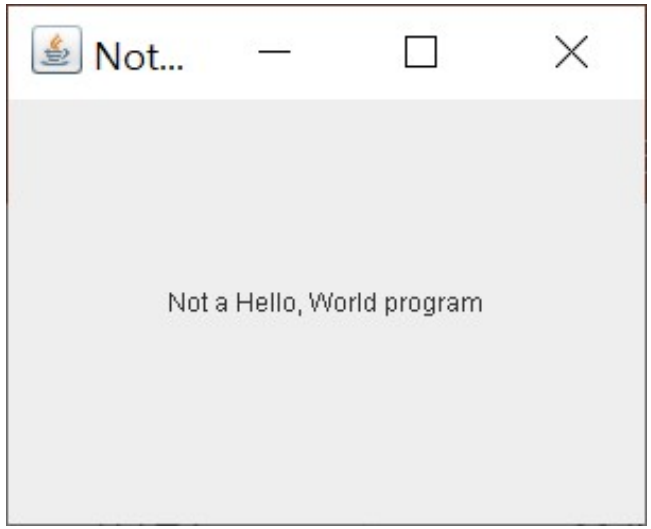
```
public Dimension getPreferredSize() { return new Dimension(300, 200); }
```

paintComponent() method and Graphics Object

- Each time a window needs to be redrawn, the `paintComponent` methods of all components in the window are executed.
  - the size of the window is changed
  - the window is minimized and then restored
  - an overlaid window disappears
  - a window is displaced for the first time
  - etc.
- Never call the `paintComponent` method yourself. It is called automatically.
- The `paintComponent` method takes one parameter of type `Graphics`.
- The `Graphics` object has a collection of settings for drawing images and text, such as the font or color.
- Measurement on a `Graphics` object for screen display is done in pixels.
  - (0,0) coordinate denotes the top left corner of the component.
  - e.g. `g.drawstring(text, x, y);`

Example : NotHelloWorld (Listing 10.2)

```
package notHelloWorld;

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

public class NotHelloWorld
{
   public static void main(String[] args)
   {
      EventQueue.invokeLater(() ->
         {
            JFrame frame = new NotHelloWorldFrame();
            frame.setTitle("NotHelloWorld");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
         });
   }
}
```



Example : NotHelloWorld (Listing 10.2)

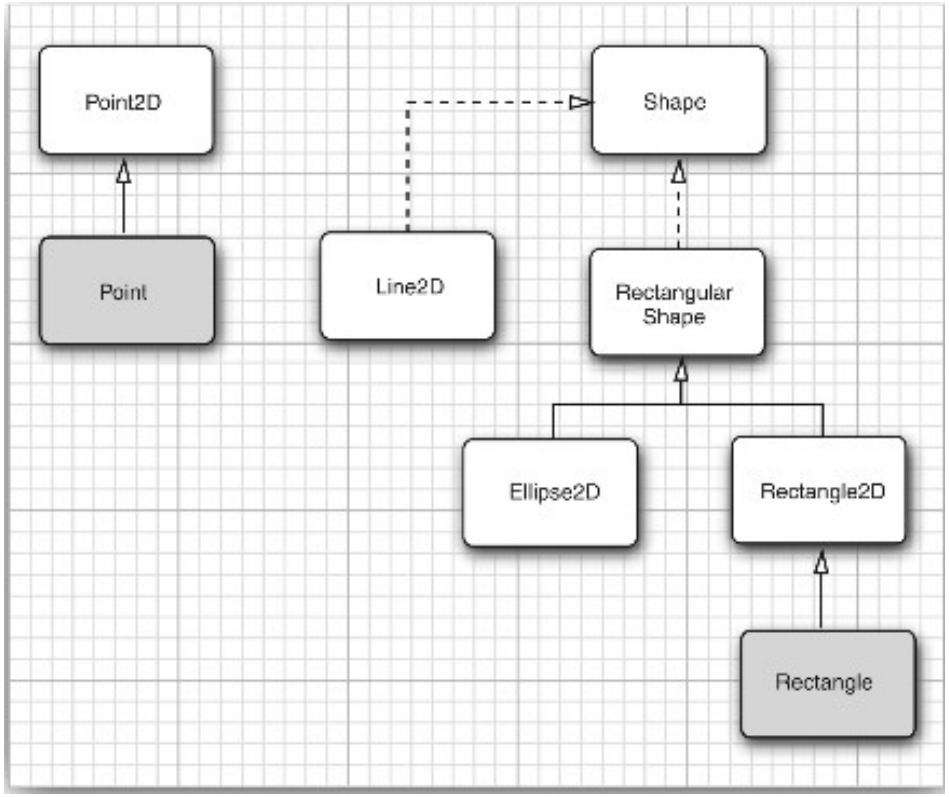
```
/**
 * A frame that contains a message panel
 */
class NotHelloWorldFrame extends JFrame
{
   public NotHelloWorldFrame()
   {
      add(new NotHelloWorldComponent());
      pack();  // layout manager determines the location and uses preferred sizes of components
   }
}
```

Example : NotHelloWorld (Listing 10.2)

```
/**
 * A component that displays a message.
 */
class NotHelloWorldComponent extends JComponent
{
   public static final int MESSAGE_X = 75;
   public static final int MESSAGE_Y = 100;

   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 200;

   public void paintComponent(Graphics g)
   {
      g.drawString("Not a Hello, World program", MESSAGE_X, MESSAGE_Y);
   }
   public Dimension getPreferredSize() { return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT); }
}
```

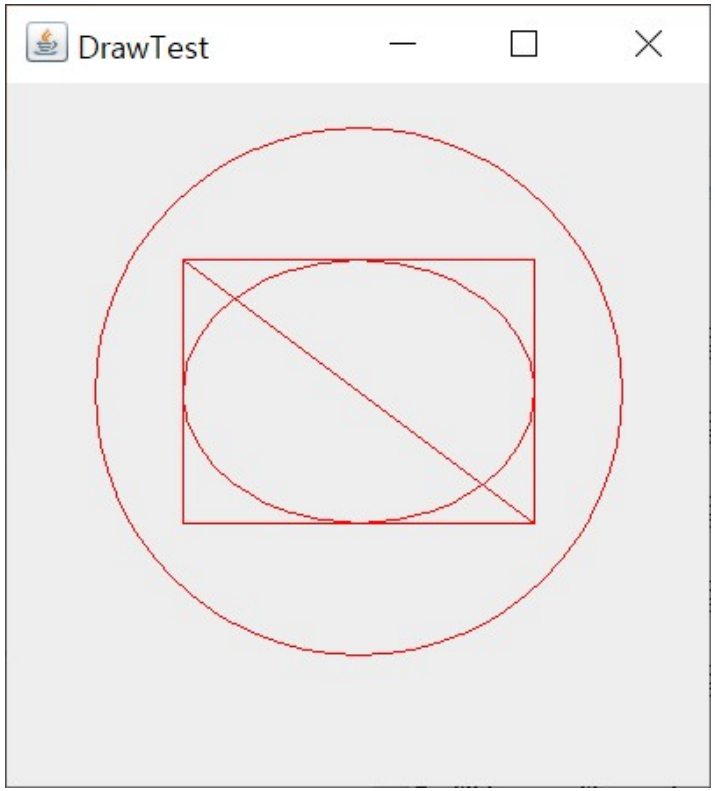
Displaying Graphical Shapes

- Displays geometrical shapes
- Displays texts
- Handles colors and fonts
- Displays images



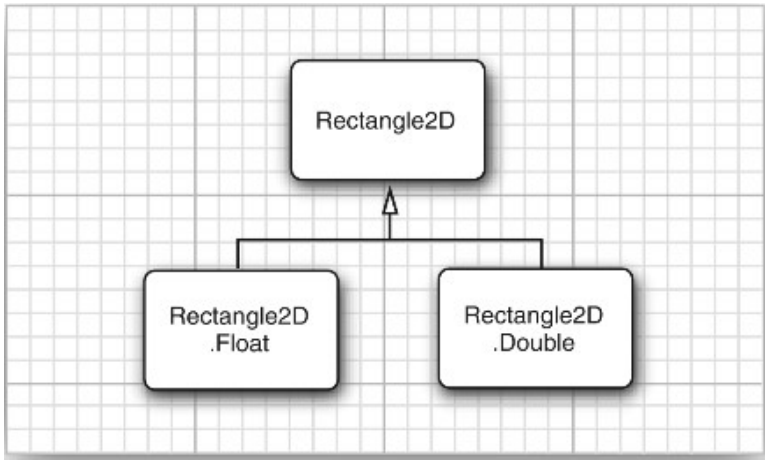
Listing 10.3: DrawTest

```
package draw;
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;
public class DrawTest
{
   public static void main(String[] args)
   {
      EventQueue.invokeLater(() ->
         {
            JFrame frame = new DrawFrame();
            frame.setTitle("DrawTest");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
         });
   }
}
```



Listing 10.3: DrawTest

```
class DrawFrame extends JFrame
{
   public DrawFrame()
   {
      add(new DrawComponent());
      pack();
   }
}
```




```
class DrawComponent extends JComponent
{
   private static final int DEFAULT_WIDTH = 400;
   private static final int DEFAULT_HEIGHT = 400;

   public void paintComponent(Graphics g)
   {
      Graphics2D g2 = (Graphics2D) g;

      // draw a rectangle
      double leftX = 100;
      double topY = 100;
      double width = 200;
      double height = 150;

      Rectangle2D rect =
            new Rectangle2D.Double(leftX, topY, width, height);
      g2.draw(rect);
```

Listing 10.3: DrawTest

```
    // draw the enclosed ellipse
    Ellipse2D ellipse = new Ellipse2D.Double();
    ellipse.setFrame(rect);
    g2.draw(ellipse);

    // draw a diagonal line
    g2.draw(new Line2D.Double(leftX, topY, leftX + width, topY + height));

    // draw a circle with the same center
    double centerX = rect.getCenterX();
    double centerY = rect.getCenterY();
    double radius = 150;

    Ellipse2D circle = new Ellipse2D.Double();
    circle.setFrameFromCenter(centerX, centerY, centerX + radius, centerY + radius);
    g2.draw(circle);
  }
  public Dimension getPreferredSize() { return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT); }
}
```