

How to make a GUI program from scratch

Because E.O. asked

Applications

- There are two main ways to make a GUI program in Java
 - applications (run on the computer)
 - applets (run in a browser)
- We'll look at how to make an application

Basics of an Application

- Parts
 - A “Driver” class
 - Frames to represent the different windows our program has
 - Layouts for where to place buttons and other graphical components
 - “Panel” classes for custom drawing areas

Basic Driver

- Purpose: To create our frame and put it on the screen so it is visible

```
public class Driver
{
    public static void main(String[] args)
    {
    }
}
```

Driver continued

- JFrame is a class that represents a window

```
import javax.swing.*;

public class Driver
{
    public static void main(String[] args)
    {
        Driver d = new Driver();
        d.init();
    }

    public void init()
    {
        JFrame frame = new JFrame();
        frame.setSize(300,300);
        frame.setTitle("My First Window");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

More Driver

- Each frame contains a **content pane**
- This is the area to which we can add GUI elements such as button and custom drawing areas
- To make sure that these GUI elements appear in the right places, we use things called **layout managers**

Content Pane

- The content pane is of type Container

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

public class Driver
{
    public static void main(String[] args)
    {
        Driver d = new Driver();
        d.init();
    }

    public void init()
    {
        JFrame frame = new JFrame();
        frame.setSize(300,300);
        frame.setTitle("My First Window");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        Container c = frame.getContentPane();
        //now to add some things...
    }
}
```

Layouts

- There are several ways to layout items in a content pane.
 - FlowLayout
 - BorderLayout
 - GridLayout
 - BoxLayout
 - GridBagLayout (confusing!)
 - SpringLayout (very useful but difficult)

FlowLayout

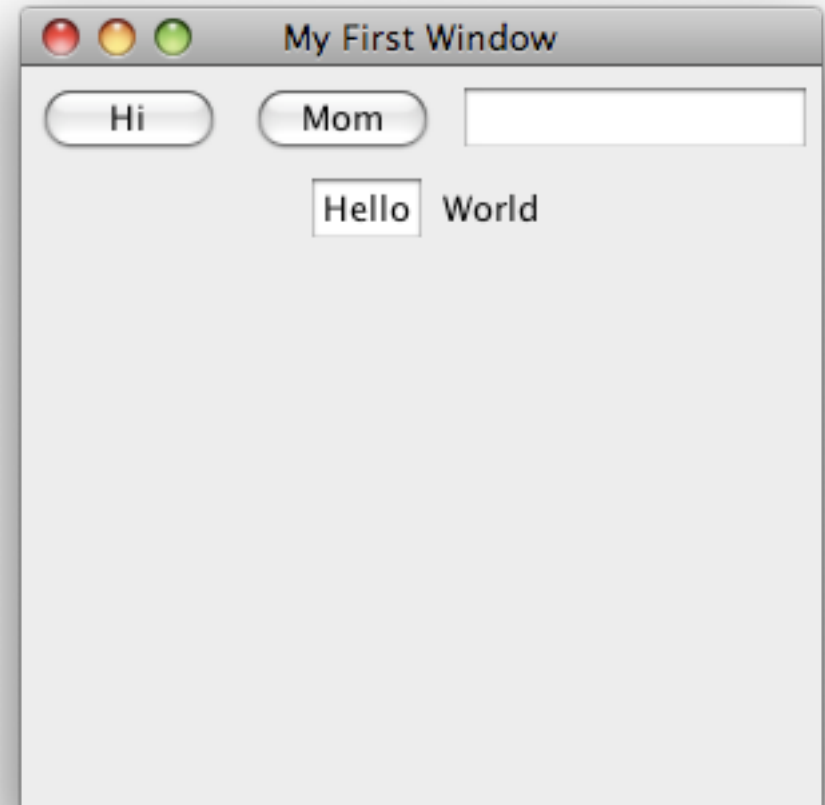
```
//...
Container c = frame.getContentPane();
c.setLayout(new FlowLayout());

JButton buttonOne = new JButton("Hi");
JButton buttonTwo = new JButton("Mom");
JTextField fieldOne = new JTextField(10);
JTextField fieldTwo = new JTextField("Hello");
JLabel label = new JLabel("World");

c.add(buttonOne);
c.add(buttonTwo);
c.add(fieldOne);
c.add(fieldTwo);
c.add(label);

c.validate();

}
}
```



FlowLayout

- By default, items are added from left to right and centered
- If the next item won't fit on the current "line", it flows to the next line and continues to be centered
- Look up "FlowLayout Java 1.5" to find out ways to change how a FlowLayout behaves
- Do similar searches for changes to JButton's, JTextField's, and JLabel's

BorderLayout

```
//...
Container c = frame.getContentPane();
c.setLayout(new BorderLayout());

JButton buttonOne = new JButton("Hi");
JButton buttonTwo = new JButton("Mom");
JTextField fieldOne = new JTextField(10);
JTextField fieldTwo = new JTextField("Hello");
JLabel label = new JLabel("World");

c.add(buttonOne, BorderLayout.EAST);
c.add(buttonTwo, BorderLayout.WEST);
c.add(fieldOne, BorderLayout.NORTH);
c.add(fieldTwo, BorderLayout.SOUTH);
c.add(label, BorderLayout.CENTER);

c.validate();
}
```



BorderLayout

- Breaks content pane into five regions: EAST, WEST, SOUTH, NORTH, and CENTER
- Added GUI elements stretch to fill the entire region they are added to
- Most useful layout when doing nesting (later)

GridLayout

```
//...
Container c = frame.getContentPane();
c.setLayout(new GridLayout(3,2));

JButton buttonOne = new JButton("Hi");
JButton buttonTwo = new JButton("Mom");
JTextField fieldOne = new JTextField(10);
JTextField fieldTwo = new JTextField("Hello");
JLabel label = new JLabel("World");

c.add(buttonOne);
c.add(buttonTwo);
c.add(fieldOne);
c.add(fieldTwo);
c.add(label);

c.validate();

}
```



GridLayout

- Breaks the content pane into r rows by c columns
- Added items go from left to right
- Added items are stretched to completely fill their cell

Nesting

- Putting GUI elements within nested layouts can help you place buttons and the like where you want them
- JPanel's allow you to do such nesting

Using a JPanel

```
//...
Container c = frame.getContentPane();
c.setLayout(new BorderLayout(3,2));

JPanel south = new JPanel(new FlowLayout());

JButton buttonOne = new JButton("Hi");
JButton buttonTwo = new JButton("Mom");

south.add(buttonOne);
south.add(buttonTwo);

c.add(south, BorderLayout.SOUTH);

c.validate();
}
}
```



Registering for Events

- Buttons when clicked generate “ActionEvents”.
- The object to receive the events has to be marked as an ActionListener
- The object to receive the events must register itself with the button

Handling an Event

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

public class Driver implements ActionListener
{
    private JButton buttonOne;

    public static void main(String[] args)
    {
        Driver d = new Driver();
        d.init();
    }
}
```

```
public void init()
{
    JFrame frame = new JFrame();
    frame.setSize(300,300);
    frame.setTitle("My First Window");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);

    Container c = frame.getContentPane();
    c.setLayout(new FlowLayout());

    buttonOne = new JButton("Hi");
    c.add(buttonOne);

    buttonOne.addActionListener(this);

    c.validate();
}

public void actionPerformed(ActionEvent e)
{
    if(e.getSource() == buttonOne)
    {
        //do something
    }
}
}
```

Custom Drawing Areas

- To add your own custom GUI elements, you should create a subclass of JPanel
- Then if you override the paint method, you can make it draw whatever you want!
- You can also register yourself for mouse and keyboard events

Custom Drawing Area

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

public class Driver
{
    public static void main(String[] args)
    {
        Driver d = new Driver();
        d.init();
    }

    public void init()
    {
        JFrame frame = new JFrame();
        frame.setSize(300,300);
        frame.setTitle("My First Window");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        Container c = frame.getContentPane();
        c.setLayout(new BorderLayout());

        DrawingPanel panel = new DrawingPanel();
        c.add(panel, BorderLayout.CENTER);

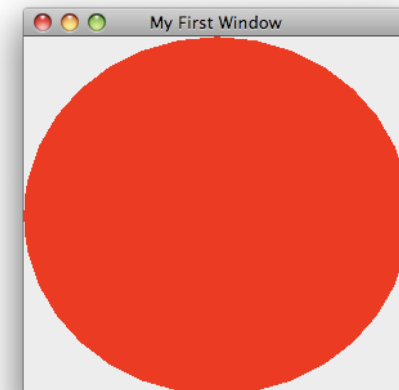
        c.validate();
    }
}
```

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

public class DrawingPanel extends JPanel
{
    public DrawingPanel()
    {
    }

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

        g.setColor(Color.RED);
        g.fillOval(0, 0, getWidth(), getHeight());
    }
}
```



Custom Drawing Area With Events

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

```
public class DrawingPanel extends JPanel implements MouseListener
```

```
{  
    public DrawingPanel()  
    {  
        addMouseListener(this);  
    }  
  
    public void paint(Graphics g)  
    {  
        super.paint(g);  
  
        g.setColor(Color.RED);  
        g.fillOval(0, 0, getWidth(), getHeight());  
    }  
  
    public void mousePressed(MouseEvent e)  
    {  
    }  
  
    public void mouseReleased(MouseEvent e){}  
    public void mouseClicked(MouseEvent e){}  
    public void mouseEntered(MouseEvent e){}  
    public void mouseExited(MouseEvent e){}  
}
```

Timer Events

- Need to create a timer and start it
- Also need to set up the panel as an ActionListener

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

```
public class DrawingPanel extends JPanel implements ActionListener
{
    private Timer timer;

    public DrawingPanel()
    {
        timer = new Timer(10, this);
        timer.start();
    }
}
```

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

    g.setColor(Color.RED);
    g.fillOval(0, 0, getWidth(), getHeight());
}

public void actionPerformed(ActionEvent e)
{
}
}
```