GUI

JFrame

Five Steps to Displaying a Frame

1. `JFrame frame = new JFrame();`
2. `frame.setSize(300,400);`
3. `frame.setTitle("An Empty Frame");`
4. `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
5. `frame.setVisible(true);`

Don't forget to import javax.swing.JFrame

Adding Components

- You cannot directly add to the JFrame
- Instead, construct an object and add it to the frame
  - Examples
    - JComponent
    - JPanel
    - JTextComponent
    - JLabel

```
// Extend the JComponent Class
public class RectangleComponent extends JComponent {

    // Override JComponent's paintComponent method
    @Override
    public void paintComponent(Graphics g) {
        // Drawing instructions go here
    }
}
```

Adding Panels

- Add components into a panel (a container for other UI components) and then add the panel to the frame

```
JButton button = new JButton("Click me");
JPanel panel new JPanel();
```

```
panel.add(button);
```

Using Inheritance to Customize Frames

- For complex frames
  - Design a subclass of `JFrame`
  - Store the components as instance variables
  - Initialize them in the constructor of your variables

```
public class FilledFrame extends JFrame {
    private JButton button;
    private JLabel label;
    private static final int FRAME_WIDTH = 400;
    private static final int FRAME_HEIGHT = 300;

    private void createComponents() {
        button = new JButton("Click me");
        label = new JLabel("Hello world");
        JPanel = new JPanel();
        panel.add(button);
        panel.add(label);
        add(panel);
    }
}

public class FilledFrameViewer {
    public static void main(String[] args) {
        JFrame frame = new FilledFrame();
        frame.setTitle("A frame");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

> **Tip**
>
> You can add the main method directly to the class(FilledFrame in this case)

Events and Event Handling

- In a modern GUI program, the user controls the program through the mouse and keyboard
- The user can perform actions via mouse and keyboard, and the program can be set up to receive and handle inputs from the mouse and keyboard

Action Listener

```
// The ActionListener interface has one method
public interface ActionListener {
    void actionPerformed(ActionEven event);
}

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ClickListener implements ActionListener {

    public void actionPerformed(ActionEvent event) {
        System.out.println("I was clicked");
    }
}
```

Registering the Action Listener

A `ClickListener` object must be created, and then registered to a specific event source

```
ActionListener listener = new ClickListener();
button.addActionListener(listener);
// When the button object is clicked, it will call listener.ActionPerformed,
passing it the event as a parameter
```

> ⚠ **Warning**
>
> Add the `ActionListener` after adding the components to the panel and adding the panel to frame

Inner Class Listener

```
public class ButtonFrame2 extends JFrame {
    private JButton button;
    private JLabel label;

    class ClickListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            label.setText("I was clicked");
        }
    }
}
```

> ⚠ **Warning**
>
> Don't forget to attach the listener

Text Areas

- …
- The `append` method adds text to the end of a text area
  - Use newline characters to separate lines
  - `textArea.append(account.getBalance() + interest)`
- Use the `setEditable` method to control user input
- `textArea setEditable(false)`

JTextField and JTextArea

- The append method is declared in the `JTextArea` class
- …
- Use `JScrollPane` to add scroll bars
  - `JScrollPane srollPane = new JScrollPane(textArea)`
  -

Creating Drawings

- You cannot draw directly on a `JFrame` objects
- Instead, construct an object and add it to the frame
  - A few examples objects to draw on are
    - `JComponent`
    - `JPanel`
    - `JTextComponent`
    - `JLabel`

paintComponent`

- Called automatically when
  - The component is shown for the first time
  - Every time the wind is resized or after being hidden

```
public class chartComponent extends JComponent {
    public void paintComponent(Graphics G) {
        // fillRect(x,y,length,width)
        // draws from the corner
        g.fillRect(0,10,200,10);
```

```
        g.fillRect(0,30,300,10);
        g.fillRect(0,50,100,10);
        // creates a bar chart
    }
}
```

```
public class RectangleComponent extends JComponent
{
    public void paintComponent(Graphics g) {
        Graphics2D = (Graphics2D) g;

    }
}
```

Ovals, Lines, Text, and Color

Ovals

- Ellipses are drawn inside a **bounding box** in the same way that you specify a rectangle
  - Provide the x and y coordinates of the top left corner
  - Provide the width and height of the bounding box
  - Use the `Graphics` class `drawOval` method to create an ellipse
  - …

Text

- …

Advanced GUI

Frame Windows

UI components are arrangements by placing them in a swing `Container` object

- `JFrame` and `JPanel`
- So far we have used `JPanel` for left to right layout

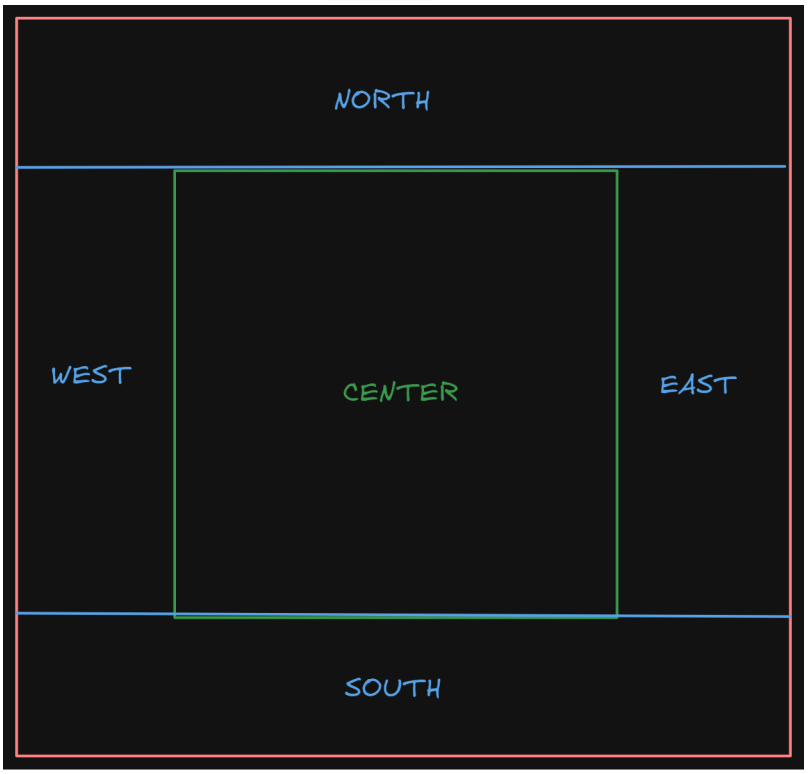
Flow Layout

- `JPanel` uses a flow layout (left to right) by default
- New row started when current row fills

Border Layout

- Five container areas: NORTH, EAST, SOUTH, WEST, CENTER

- The content pane of a `JFrame` uses border layout by default



Grid Layout

- Components are placed in boxes in a simple table arrangement
  - Specify the size (rows then columns) of the grid on `JPanel`
  - `buttonPanel.setLayout(newGridLayout(4,3))`

Nested Panels

- Create complex layouts by nesting panels

```
JPanel keypadPanel = new JPanel();
keypadPanel.setLayout(new BorderLayout());
buttonPanel = new JPanel();
buttonPanel.setLayout(new GridLayout(4,3));
buttonPanel.add(button7);
buttonPanel.add(button8);
//...
keypadPanel.add(buttonPanel, BorderLayout.CENTER);
JTextField ...
```

> ⚠ **Warning**
>
> If multiple items are added to one `BorderLayout` region directly, only the most recently added one will show

Choices

Radio Buttons

For a small set of mutually exclusive choices

- Use a panel for each set of radio buttons

```
JPanel panel = new JPanel();
panel.add(sButton);
panel.add(mButton);
panel.add(lButton);
panel.setBorder(new TitledBorder(new EtchedBorder(), "Size"));
```

```
//setting selected state
sButton.setSelected(true)
// customary to set one as true as a default state
if (sButton.isSelected()) {
    // check if selected
    // ...
}
```

Button Group

- Allows only one in the group to be selected at once

```
JRadioButton sButton = new JRadioButton("S");
JRadioButton mButton = new JRadioButton("M");
JRadioButton lButton = new JRadioButton("L");
ButtonGroup group = new ButtonGroup();
group.add(sButton);
group.add(mButton);
group.add(lButton);
```

Check Boxes

For a binary choice/choices

-

Combo Boxes (Dropdown)

For a large set of choices