Welcome to ICS 111

Introduction to GUI Programming

4/2 - 4/4

Introduction to GUI Programming

- Critical in software development for creating intuitive and visually appealing applications for different benefits:

Enhanced User Experience - GUIs help leveager feedback to help guide users more effectively.

Increased Productivity - GUIs provide menu bars and toolbars for quick access to commonly used functions

Broader Accessibility - Visual cues help cater to users with a varying level of technical expertise

Common GUI Frameworks

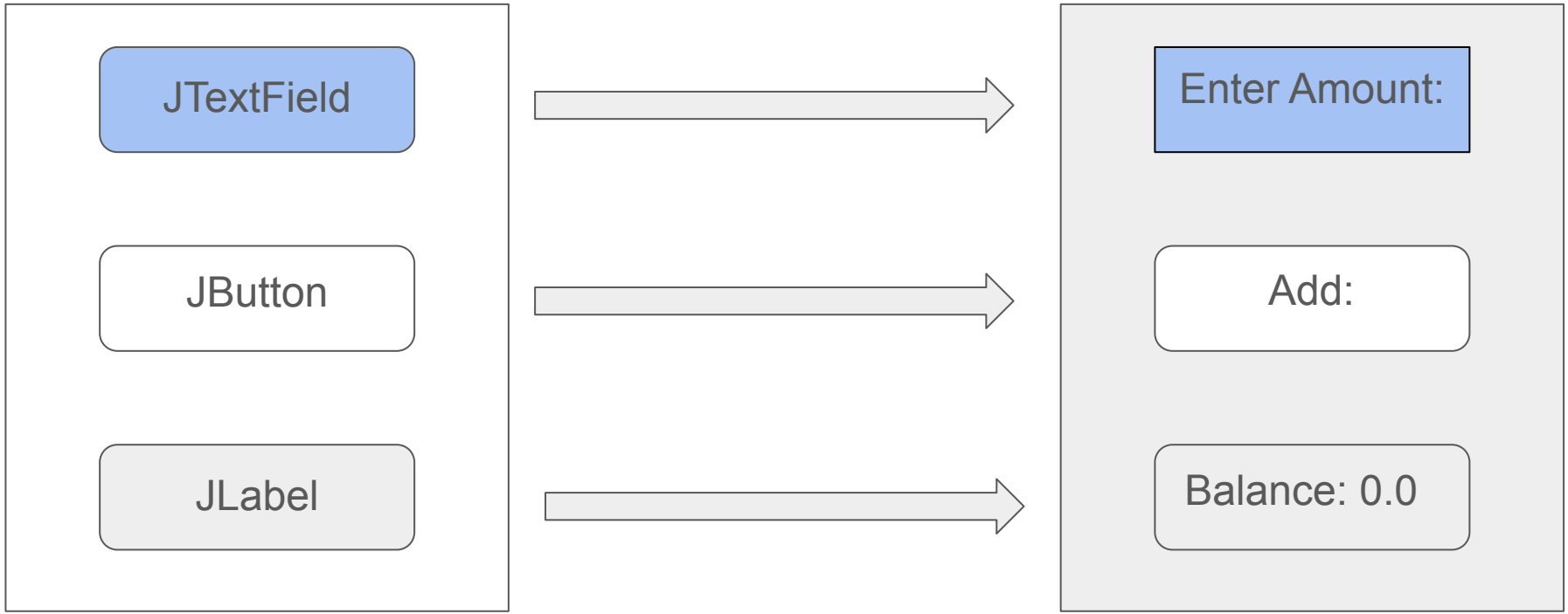
- GUI frameworks are essential for developers to create GUIs efficiently.
- Swing is a powerful GUI toolkit apart of the Java Foundation Classes (JFC) and includes key features:

I. Extensive Library: comprehensive components such as buttons, labels, text fields, and other features for creating sophisticated GUIs

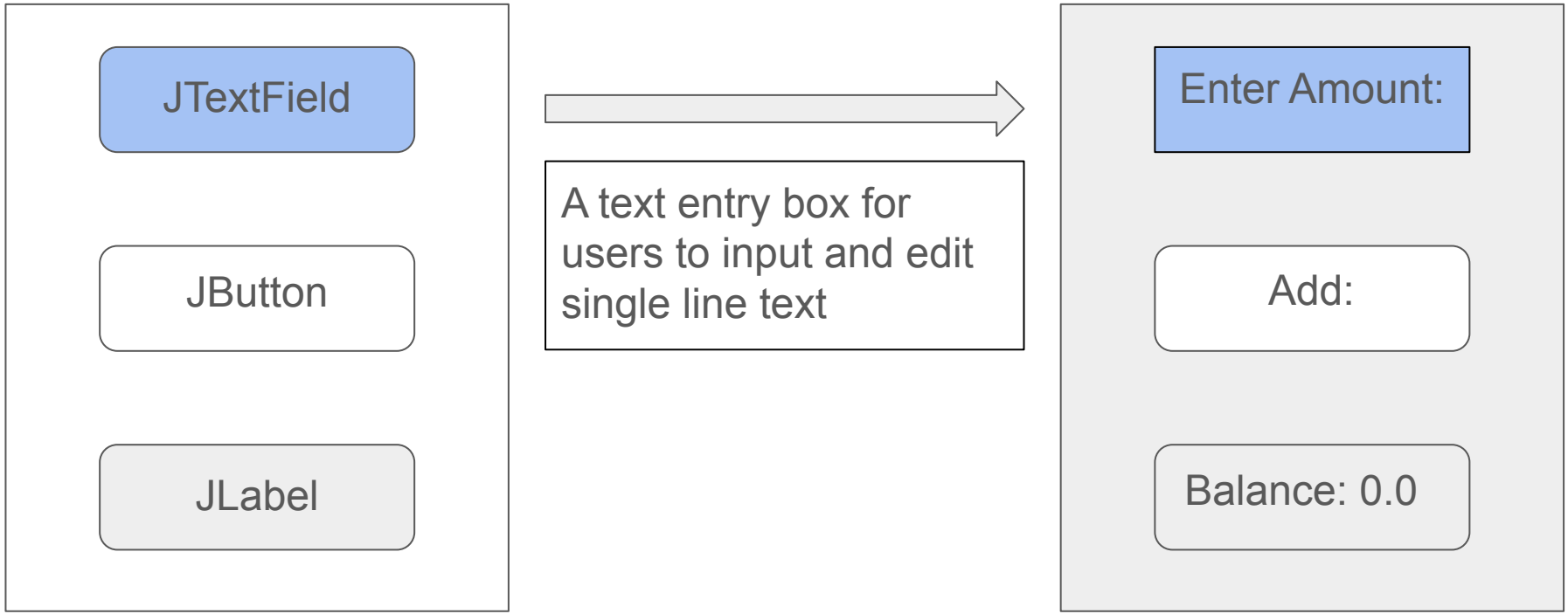
II. Variety of Layout Managers: BorderLayout, GridLayout, and FlowLayout to arrange information within containers more dynamically.

III. Event Handling: button clicks and other user interactions corresponding to an action are handled by event listeners as apart of the GUIs components.

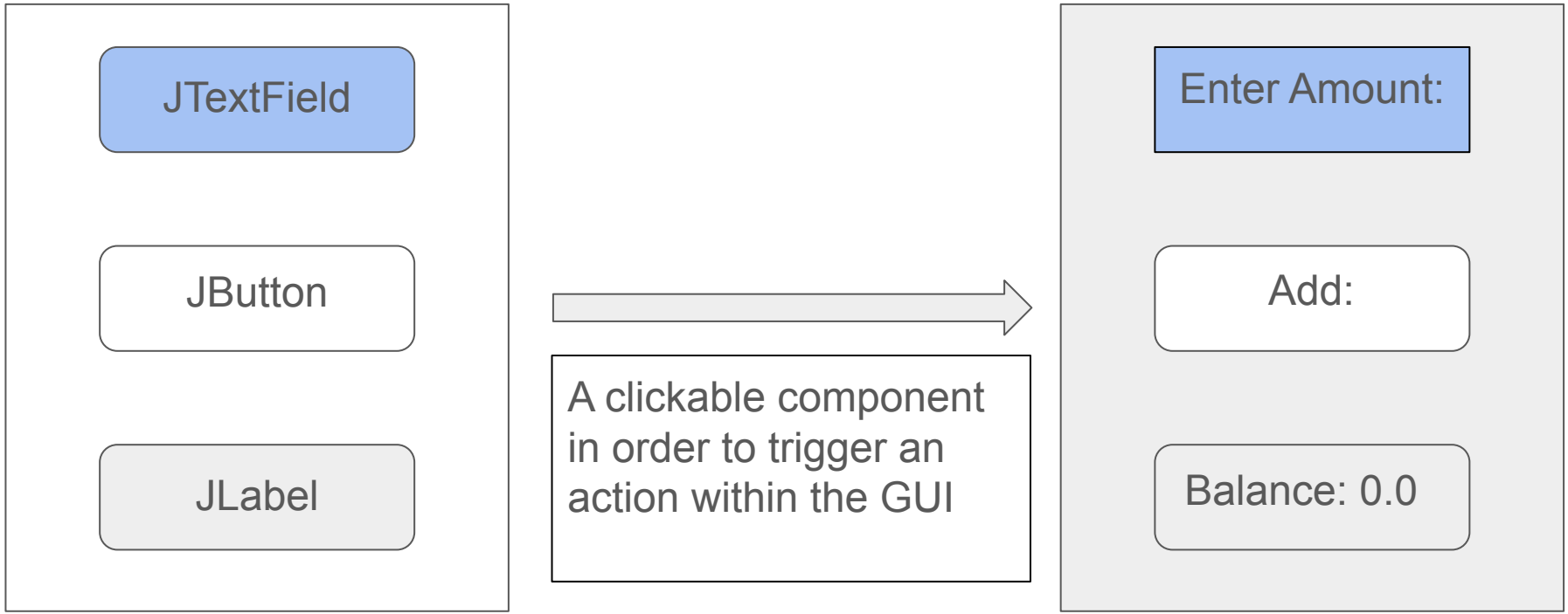
Components and Layout of JPanel



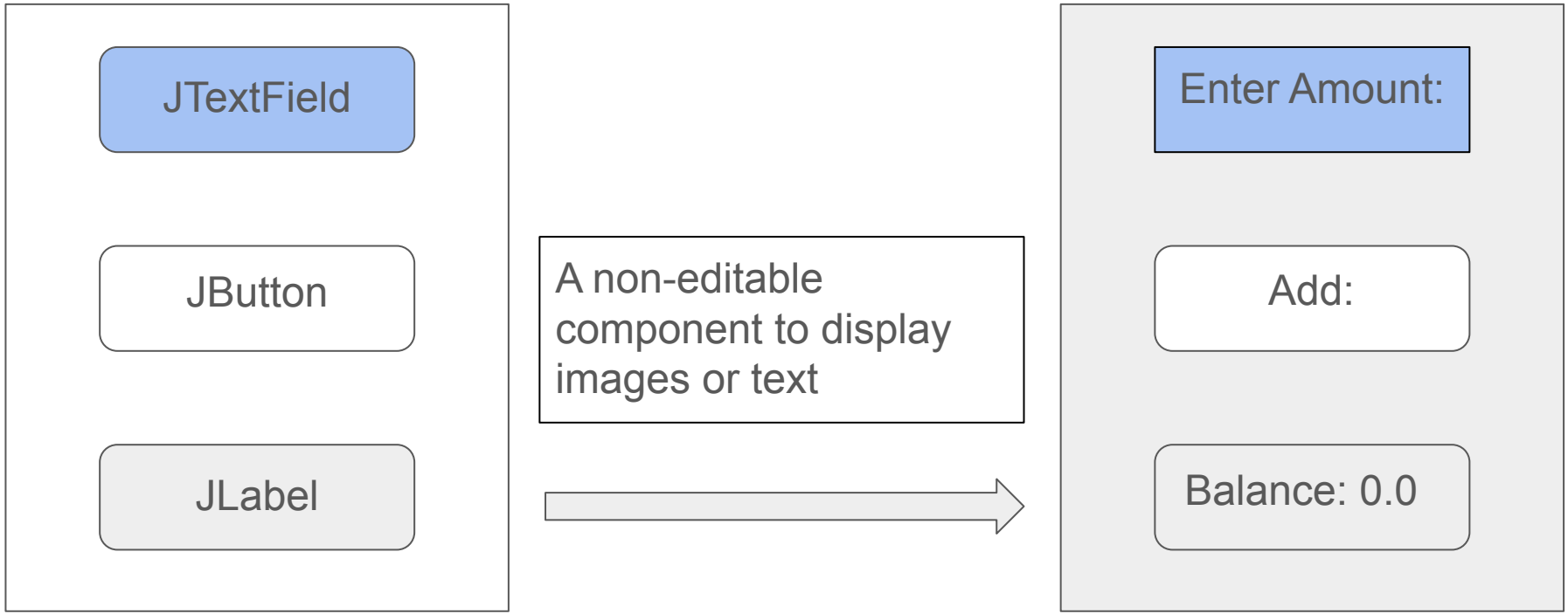
Components and Layout of JPanel



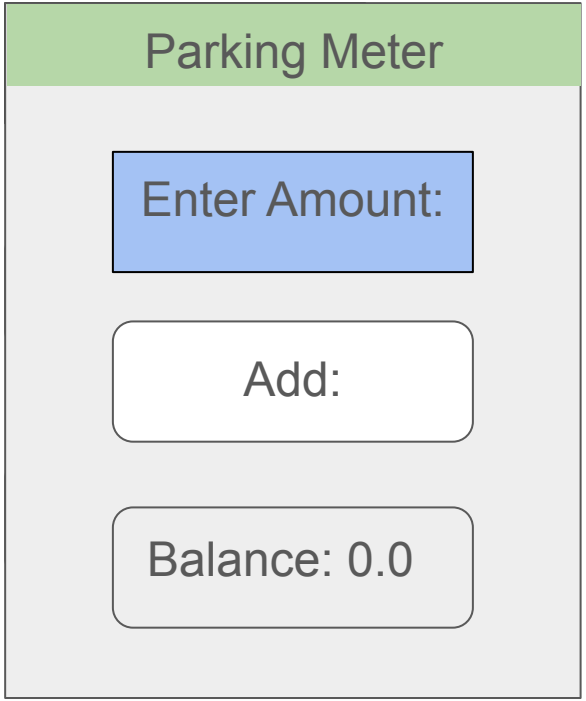
Components and Layout of JPanel



Components and Layout of JPanel

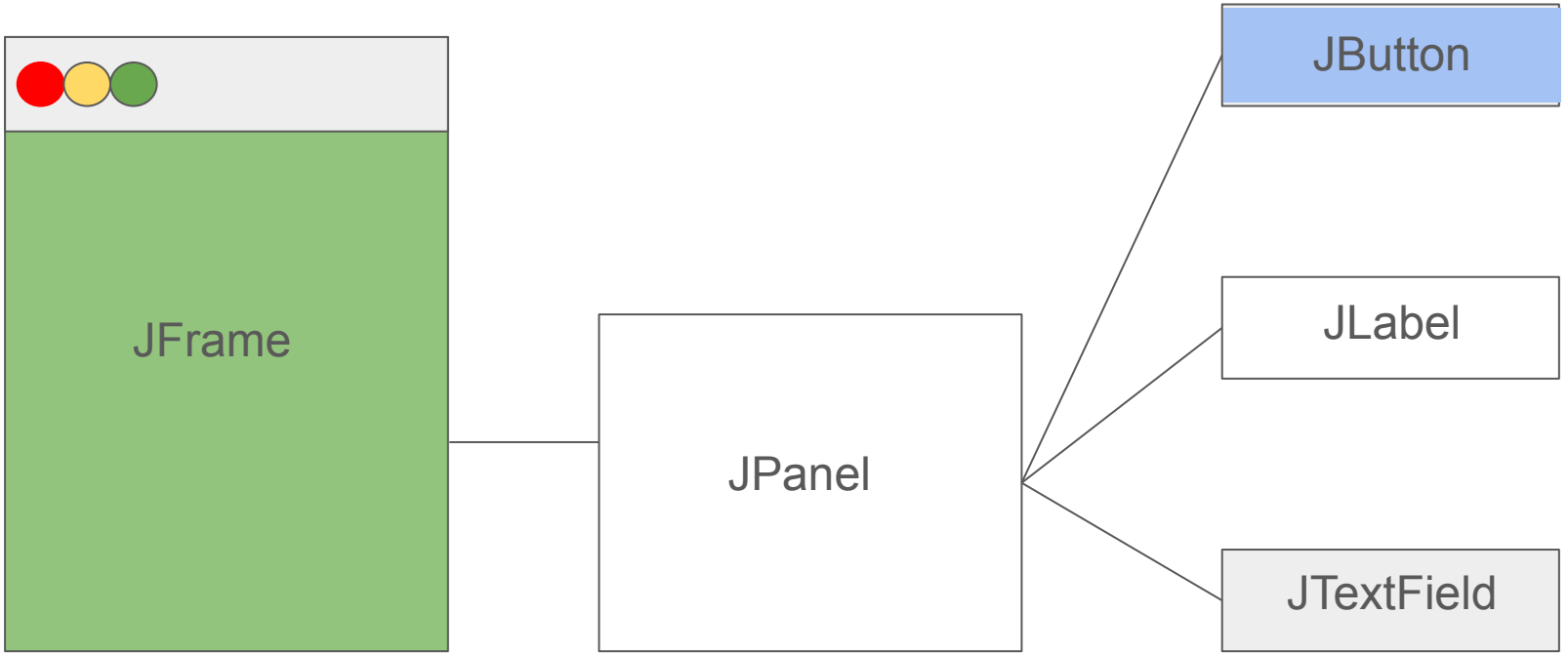


JPanel, JButton, JTextField, JLabel code breakdown

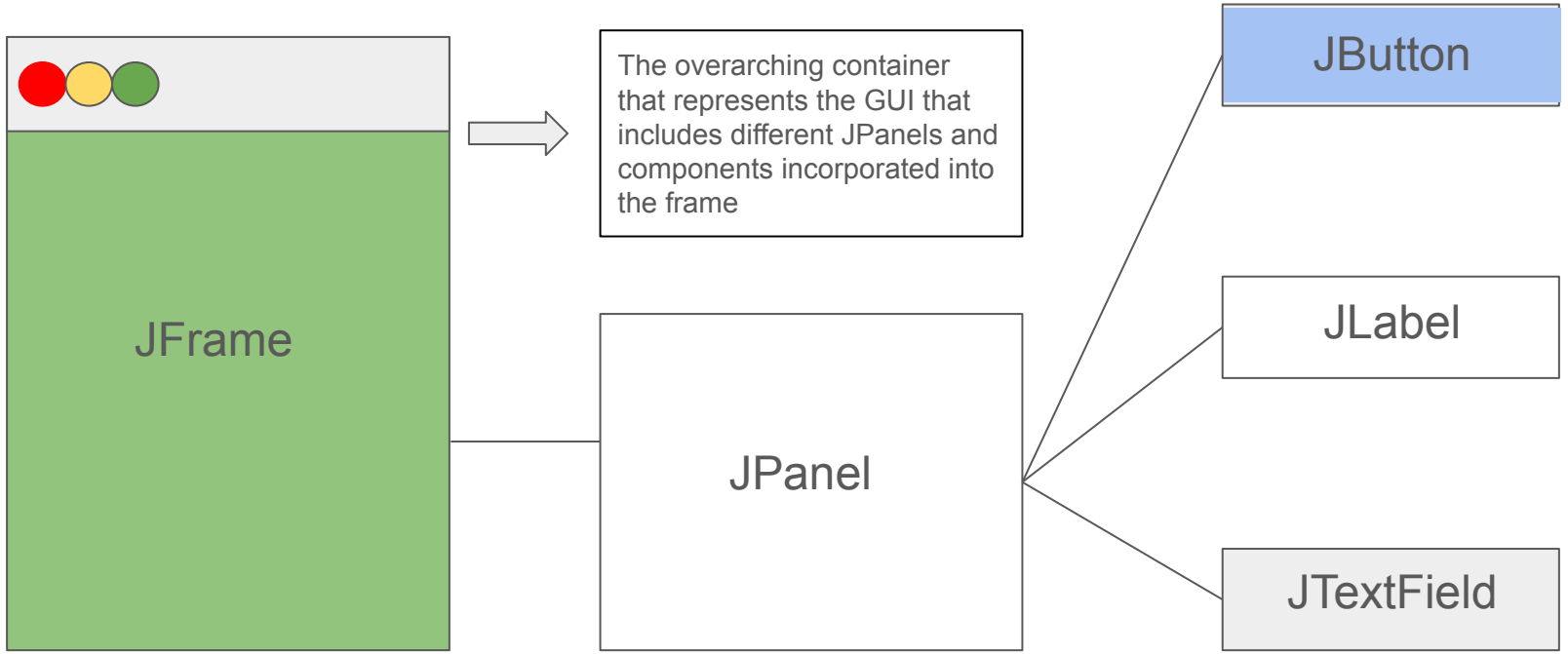



```
Import javax.swing.JFrame;
Import javax.swing.JPanel;
Import javax.swing.JButton;
Import javax.swing.JTextField;
Import javax.swing.JLabel;

double balance = 0;
JFrame frame = new JFrame();
JPanel mainPanel = new JPanel();
JButton addButton = newJButton(“Add”);
JTextField amount = new JTextField(5);
JLabel balanceLabel = new JLabel(“Balance: “ + balance);

mainPanel.add(amount);
mainPanel.add(addButton);
mainPanel.add(balanceLabel);

frame.add(mainPanel);
frame.setSize(150,200);
frame.setTitle(“Parking Meter”);
```

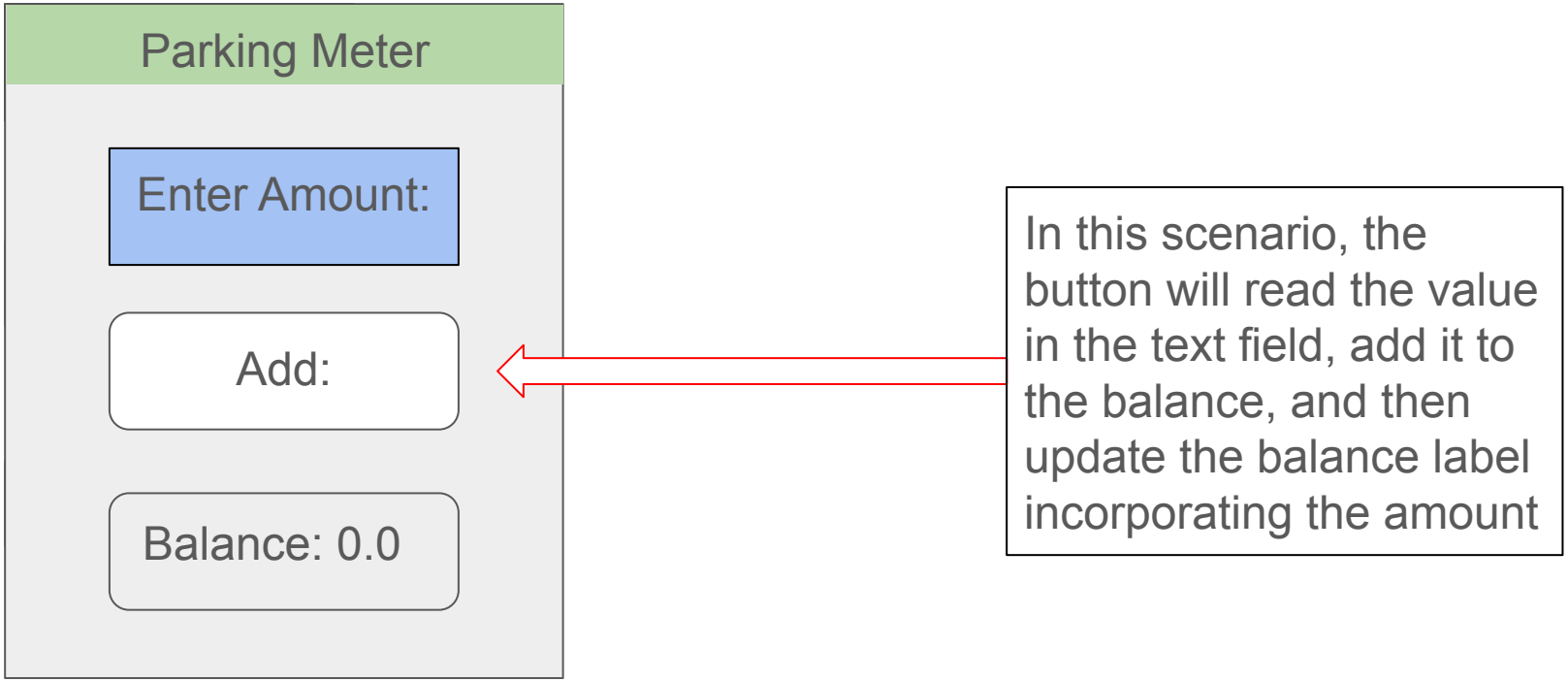
Components and Layout of JFrame GUI



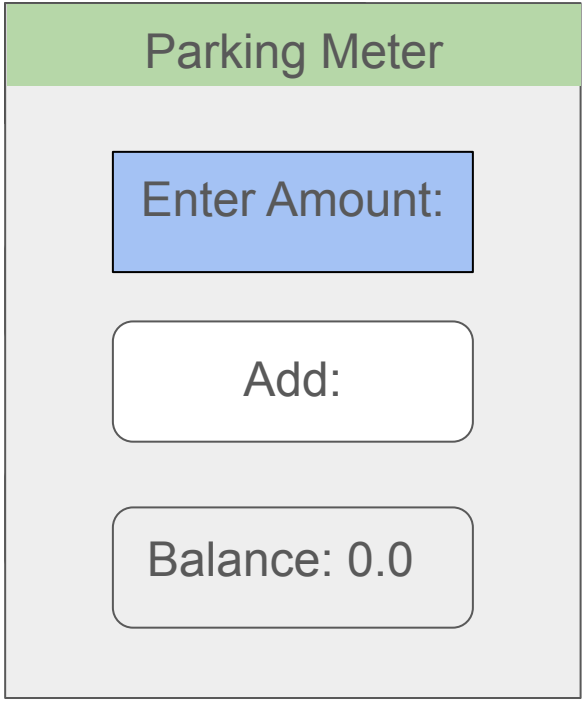
Components and Layout of JFrame GUI



Listeners


Parking Meter
Enter Amount:
Add:
Balance: 0.0
In this scenario, the button will read the value in the text field, add it to the balance, and then update the balance label incorporating the amount

Code for Listeners with a JButton

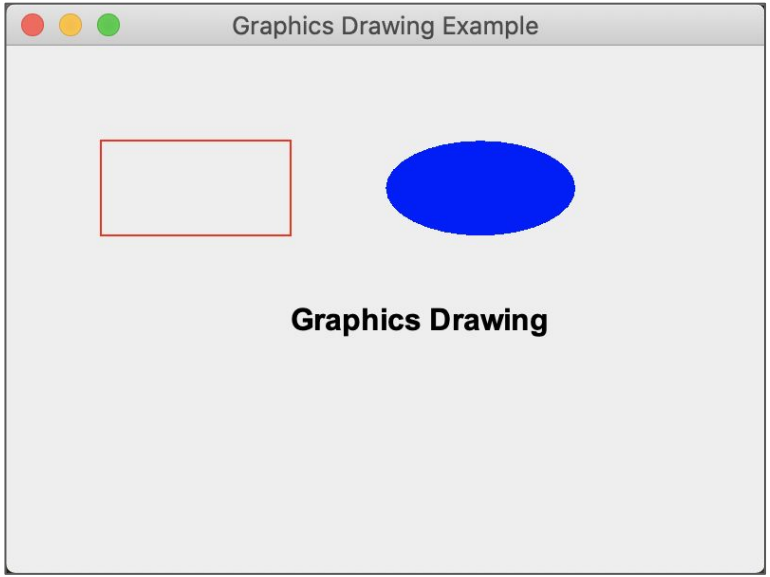



```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class AddListener implements ActionListener {
 public void actionPerformed(ActionEvent event) {
  amount = amountField.getText();
  balance = balance +
Double.parseDouble(amount);
  balanceLabel.setText("Balance: " + balance);
 }
}
addButton.addActionListener(addListener);
```

Graphics Drawing Key Concepts

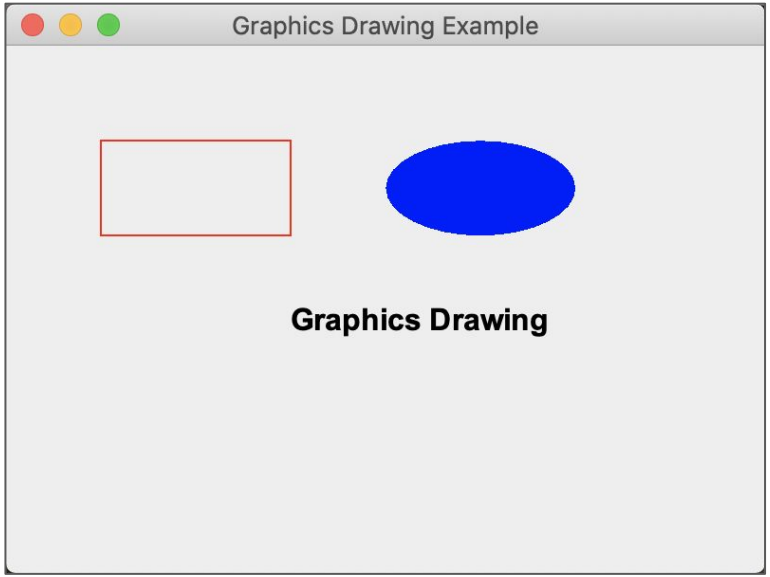
- Coordinate System: Swing uses Cartesian coordinates with positive x-values increasing to the right and positive y-values increasing downwards.
- Shapes & Lines: Methods like drawLine, drawRect, and drawPolygon can be implemented to draw specific geometrical shapes.
- Drawing Text: The method drawString to specify the text content, font, color, and position

Graphics Drawing Example




```
Import javax.swing.*;
Import javax.awt.*;

public class graphicsDrawingExample {
        public static void main(String[] argos) {
                JFrame frame = new JFrame("Graphics Drawing Example
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(400,300)
```

Graphics Drawing Example




```
JPanel panel = new JPanel();
        protected void paintComponent(Graphics g)
                super.paintComponent(g)

                g.setColor(Color.RED);
                g.drawRect(50,50,100,50);

                g.setColor(Color.BLUE);
                g.drawOval(100,50,100,50);

                g.setColor(Color.BLACK);
                g.setFont(new Font("Arial", Font.BOLD, 16);
                g.drawString("Graphics Drawing",150,150)
        }
}
frame.add(panel);
frame.setVisible(true);
```

Layout Managers

I. BorderLayout:
- The default layout.
- Divides the into 5 regions (North, East, South, West, and Center), and is ideal for resizing layouts with a main component at the center and additional components on the edges

II. FlowLayout
- Arranges components in a row or column which are added one after the other while maintaining their preferred size.
- Suitable for building toolbars and simple button layouts

III. GridLayout
- Components are organized in a grid-like orientation with a pre-determined and specified number of rows and columns which can be useful for creating forms and tables

BorderLayout Code Example

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

public class BorderLayoutExample {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("BorderLayout Example");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            // Create a JPanel with BorderLayout
            JPanel panel = new JPanel();
            panel.setLayout(new BorderLayout());

            // Add buttons to different regions
            panel.add(new JButton("North"), BorderLayout.NORTH);
            panel.add(new JButton("South"), BorderLayout.SOUTH);
            panel.add(new JButton("East"), BorderLayout.EAST);
            panel.add(new JButton("West"), BorderLayout.WEST);
            panel.add(new JButton("Center"), BorderLayout.CENTER);

            // Add the panel to the frame
            frame.add(panel);

            frame.setSize(400, 300);
            frame.setVisible(true);
        });
    }
}
```