

Muhammad Deva Alfila Majid

2502042656

Assignment 2 – Business Application Development

### 1. Explain and illustrate 3 layout managers in Java!

In Java, layout managers are used to define the arrangement and positioning of components within a container (such as a JFrame or JPanel) in a graphical user interface (GUI). They help maintain a consistent and organized appearance of components across different screen sizes and resolutions. Here, I'll explain and illustrate three commonly used layout managers in Java: BorderLayout, GridLayout, and FlowLayout.

#### A. BorderLayout:

BorderLayout divides the container into five regions: NORTH, SOUTH, EAST, WEST, and CENTER. Components are added to these regions, and they expand to fill the available space within their respective regions. The CENTER region takes up any remaining space not occupied by the other regions.

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

public class BorderLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("BorderLayout Example");

        JPanel panel = new JPanel(new BorderLayout());

        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);

        frame.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setVisible(true);
    }
}
```

#### B. GridLayout:

GridLayout arranges components in a grid with a specified number of rows and columns. Each cell in the grid is of equal size, and components are added row by row. The grid expands to accommodate additional components.

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

public class GridLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("GridLayout Example");

        JPanel panel = new JPanel(new GridLayout(3, 2));

        panel.add(new JButton("Button 1"));
        panel.add(new JButton("Button 2"));
        panel.add(new JButton("Button 3"));
        panel.add(new JButton("Button 4"));
        panel.add(new JButton("Button 5"));
        panel.add(new JButton("Button 6"));

        frame.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

#### C. FlowLayout:

FlowLayout arranges components in a single row or column, and when the available space is filled, it wraps components to the next row or column. It's ideal for creating a simple linear arrangement of components.

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

public class FlowLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("FlowLayout Example");

        JPanel panel = new JPanel(new FlowLayout());

        panel.add(new JButton("Button 1"));
        panel.add(new JButton("Button 2"));
        panel.add(new JButton("Button 3"));
        panel.add(new JButton("Button 4"));
    }
}
```

```
        frame.add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150);
        frame.setVisible(true);
    }
}
```

**2. From the following GUI, explain what components, containers, and layouts are used!**

**A. Components:**

- JLabel: Used for displaying labels like "Name:", "Phone:", "Address:".
- JTextField: Allows users to enter text (for name, phone, and address).
- JComboBox: Provides a drop-down list for selecting size and style.
- JCheckBox: Enables users to select toppings.
- JButton: Triggers the order action(OK).

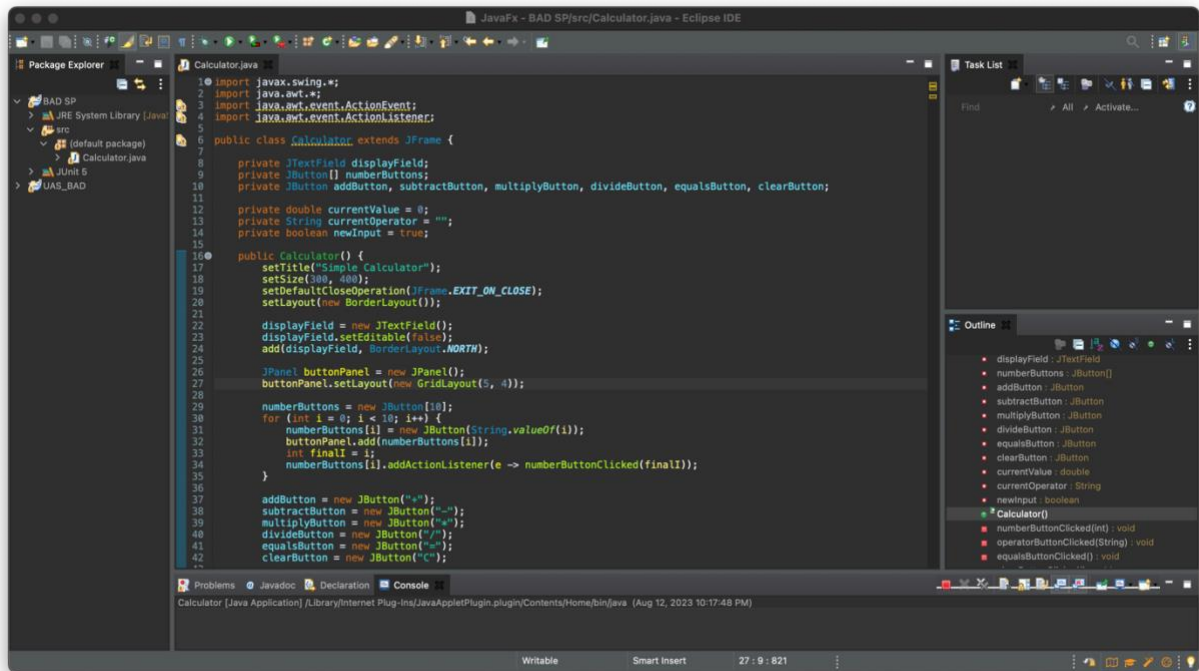
**B. Containers:**

- JFrame: The main application window.
- JPanel: Contains the input components (labels, text fields, combo boxes, check boxes) organized using GridLayout.

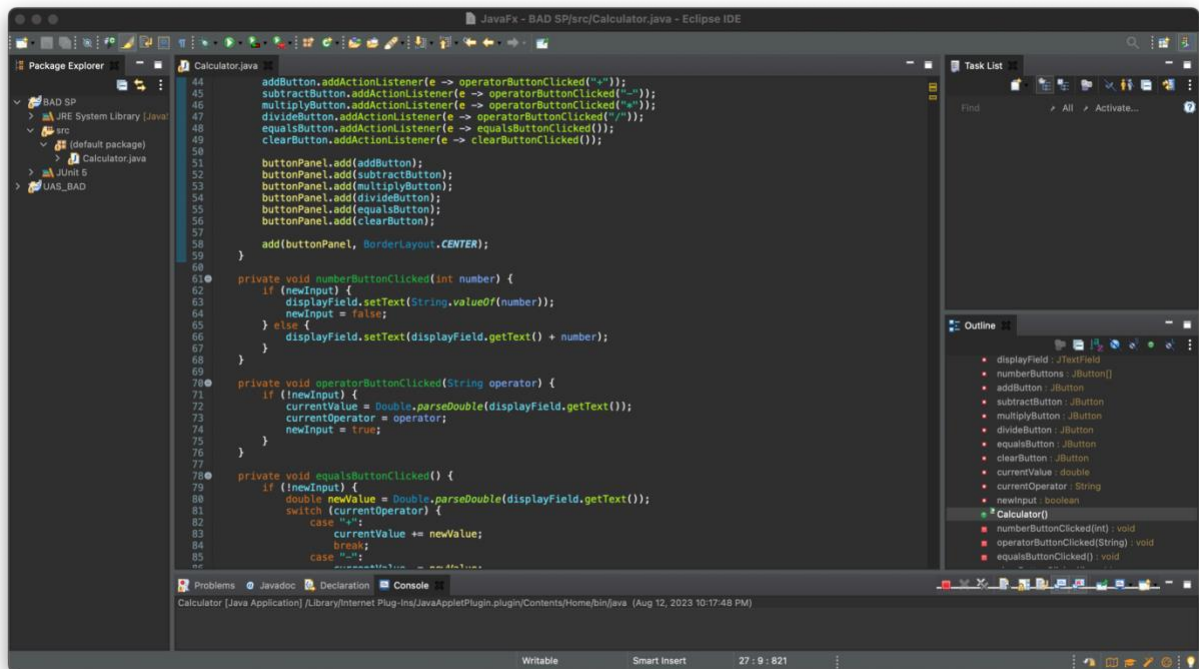
**C. Layouts:**

- BorderLayout: Used for the main application layout. Input components are placed in the CENTER region, and the "OK" button is placed in the SOUTH region.
- GridLayout: Used within the input JPanel to arrange labels and corresponding input fields in a grid.

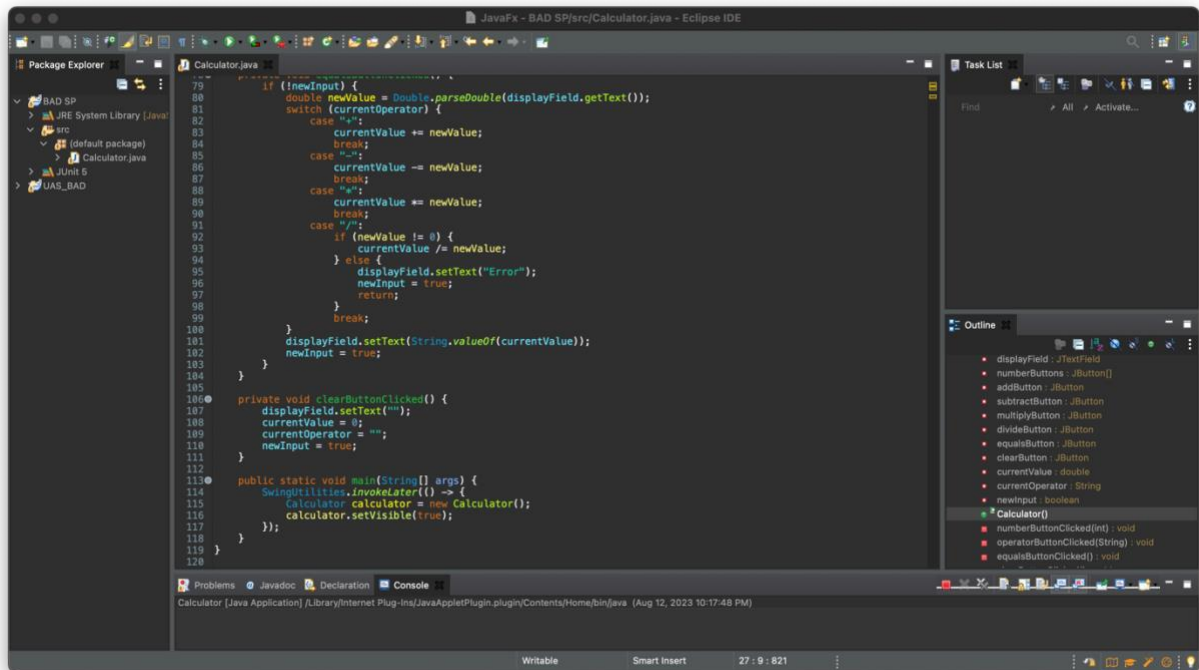
### 3. Create a simple calculator with GUI in Java! Input:



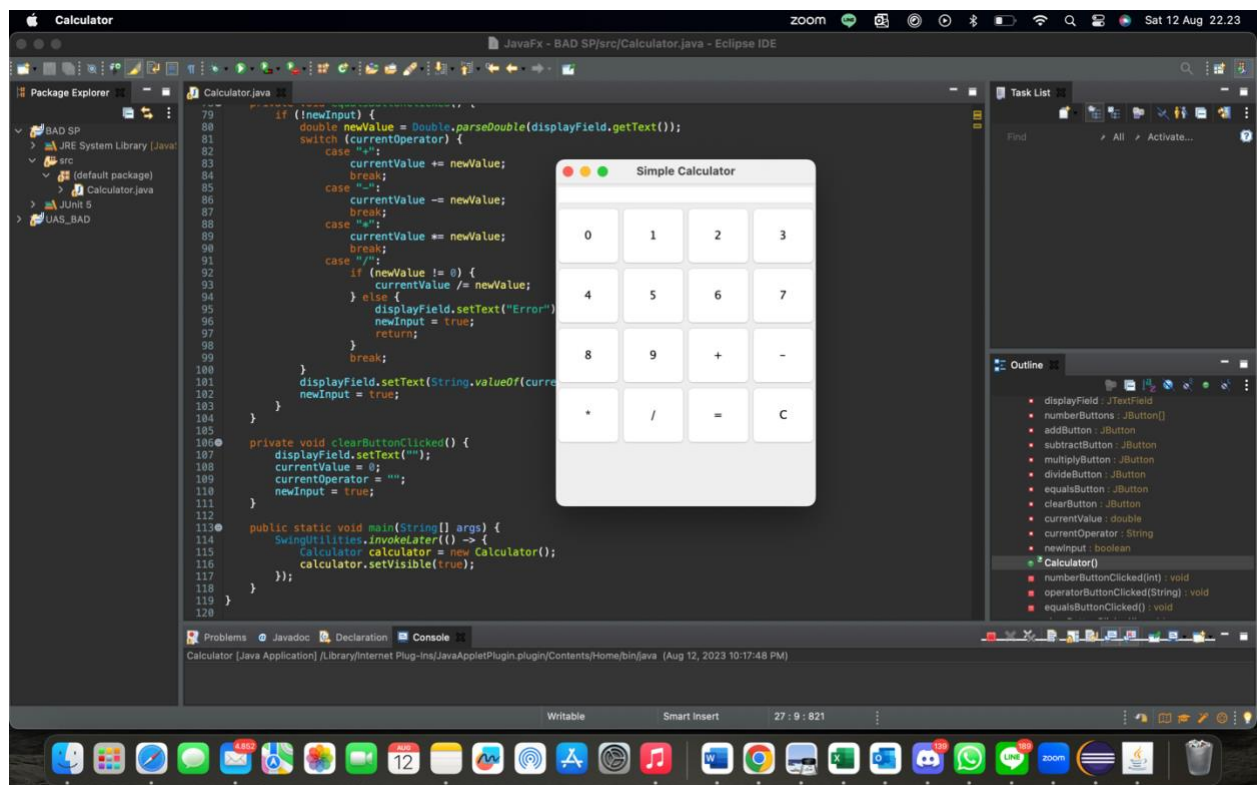
```
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 public class Calculator extends JFrame {
7
8     private JTextField displayField;
9     private JButton[] numberButtons;
10    private JButton addButton, subtractButton, multiplyButton, divideButton, equalsButton, clearButton;
11
12    private double currentValue = 0;
13    private String currentOperator = "";
14    private boolean newInput = true;
15
16    public Calculator() {
17        setTitle("Simple Calculator");
18        setSize(300, 400);
19        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20        setLayout(new BorderLayout());
21
22        displayField = new JTextField();
23        displayField.setEditable(false);
24        add(displayField, BorderLayout.NORTH);
25
26        JPanel buttonPanel = new JPanel();
27        buttonPanel.setLayout(new GridLayout(5, 4));
28
29        numberButtons = new JButton[10];
30        for (int i = 0; i < 10; i++) {
31            numberButtons[i] = new JButton(String.valueOf(i));
32            buttonPanel.add(numberButtons[i]);
33            int finalI = i;
34            numberButtons[i].addActionListener(e -> numberButtonClicked(finalI));
35        }
36
37        addButton = new JButton("+");
38        subtractButton = new JButton("-");
39        multiplyButton = new JButton("*");
40        divideButton = new JButton("/");
41        equalsButton = new JButton("=");
42        clearButton = new JButton("C");
```



```
44 addButton.addActionListener(e -> operatorButtonClicked("+"));
45 subtractButton.addActionListener(e -> operatorButtonClicked("-"));
46 multiplyButton.addActionListener(e -> operatorButtonClicked("*"));
47 divideButton.addActionListener(e -> operatorButtonClicked("/"));
48 equalsButton.addActionListener(e -> equalsButtonClicked());
49 clearButton.addActionListener(e -> clearButtonClicked());
50
51 buttonPanel.add(addButton);
52 buttonPanel.add(subtractButton);
53 buttonPanel.add(multiplyButton);
54 buttonPanel.add(divideButton);
55 buttonPanel.add(equalsButton);
56 buttonPanel.add(clearButton);
57
58 add(buttonPanel, BorderLayout.CENTER);
59
60 private void numberButtonClicked(int number) {
61     if (newInput) {
62         displayField.setText(String.valueOf(number));
63         newInput = false;
64     } else {
65         displayField.setText(displayField.getText() + number);
66     }
67 }
68
69 private void operatorButtonClicked(String operator) {
70     if (newInput) {
71         currentValue = Double.parseDouble(displayField.getText());
72         currentOperator = operator;
73         newInput = true;
74     }
75 }
76
77 private void equalsButtonClicked() {
78     if (newInput) {
79         double newValue = Double.parseDouble(displayField.getText());
80         switch (currentOperator) {
81             case "+":
82                 currentValue += newValue;
83                 break;
84             case "-":
85                 currentValue -= newValue;
```



Output:



I also attached a calculator java file in the zip