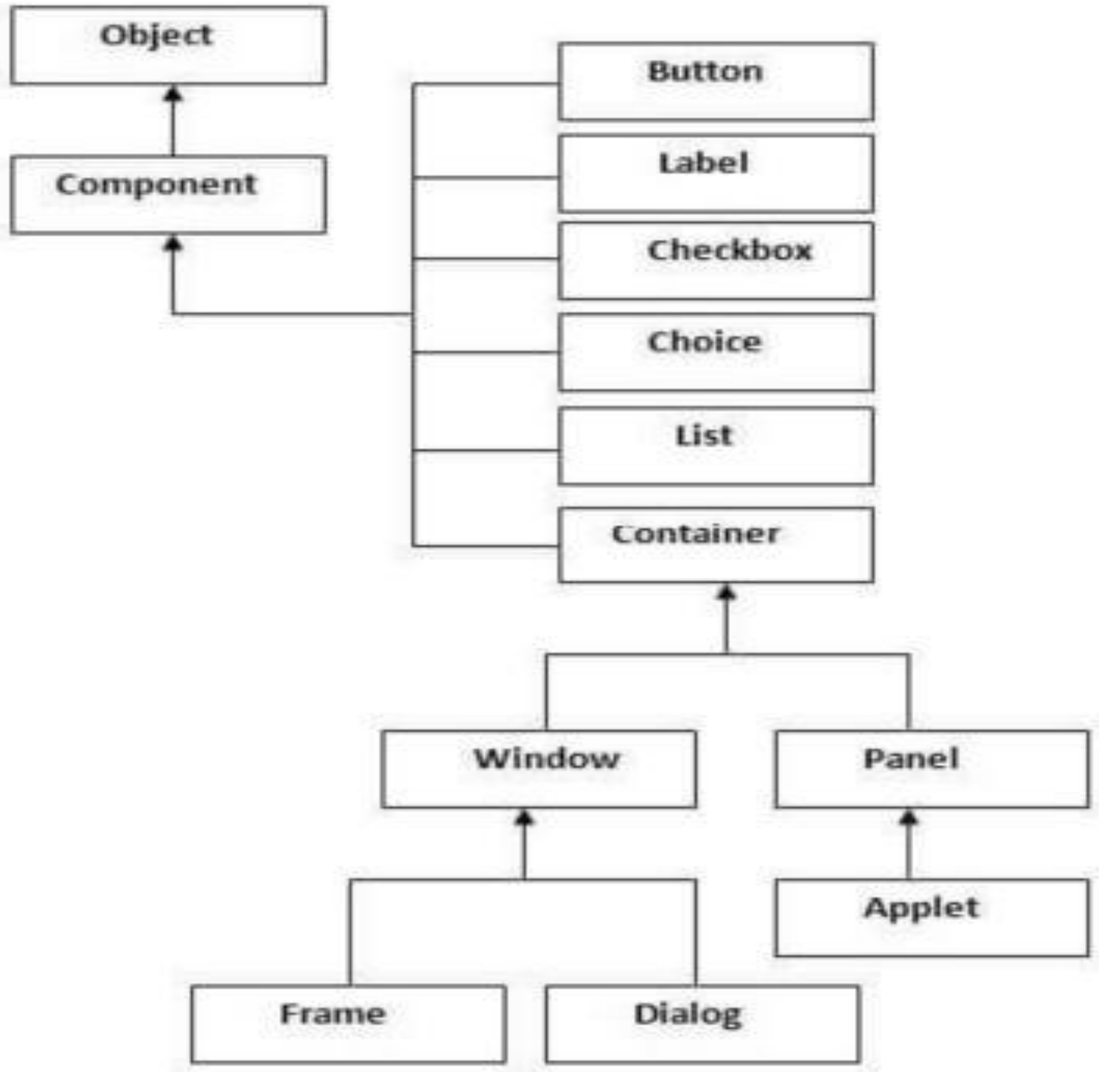
Unit 11
Java Applications

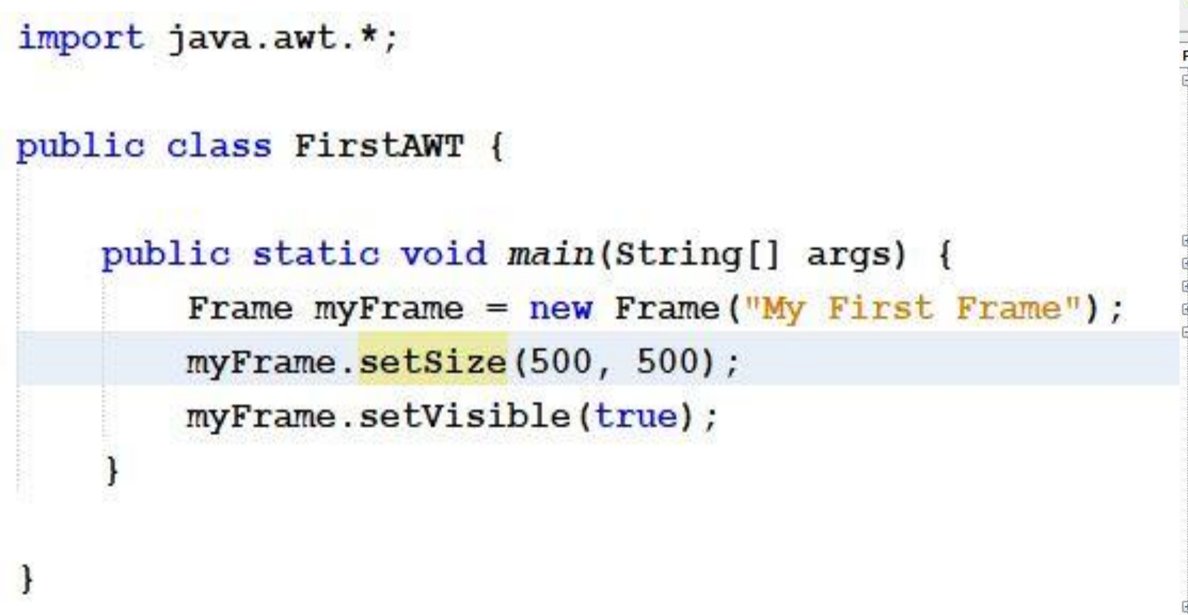
Java AWT

- ❖ Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
- ❖ Java AWT components are **platform-dependent i.e**. components are displayed according to the view of operating system.
- ❖ AWT is heavyweight, consuming more resource

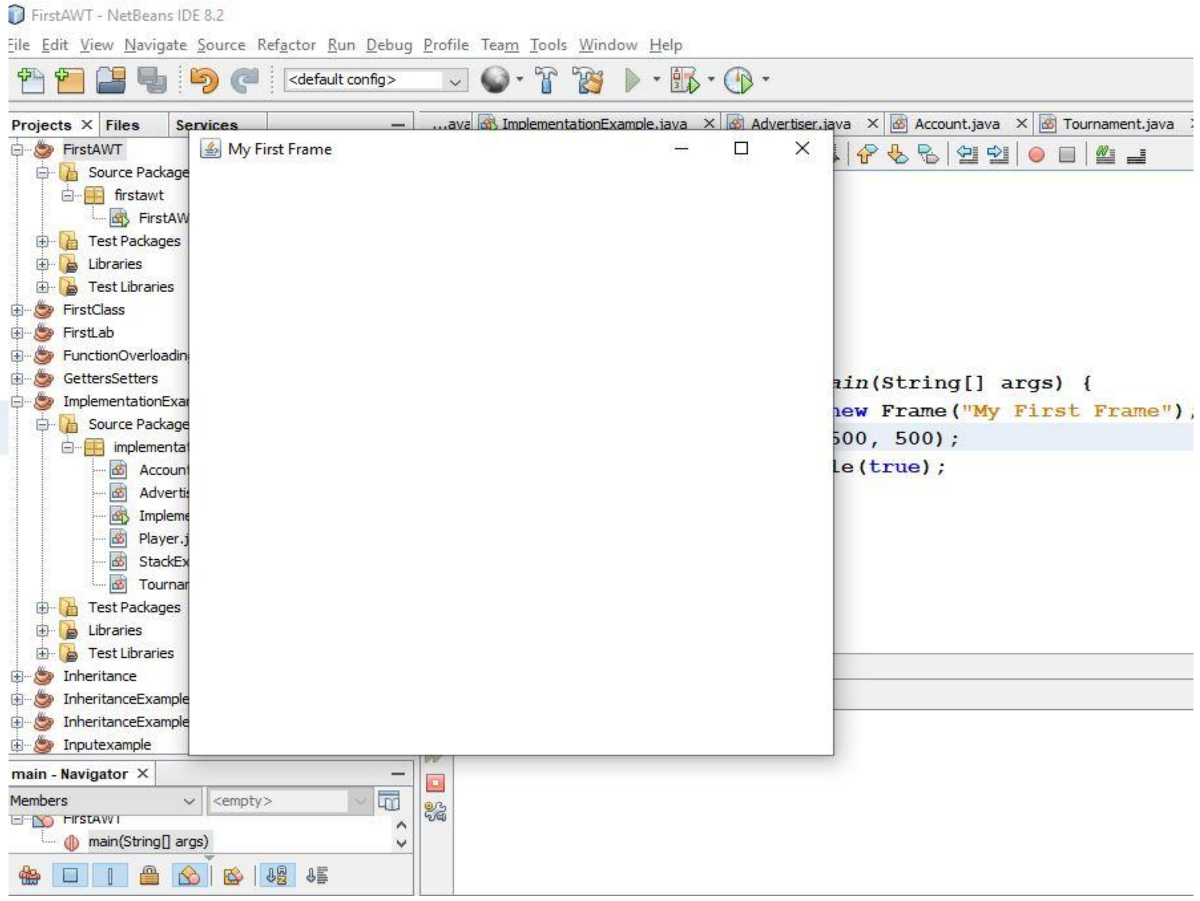
Java AWT Class Hierarchy



First Java AWT Program

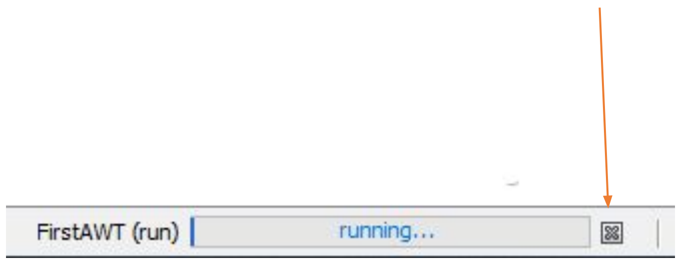


```
import java.awt.*;

public class FirstAWT {

    public static void main(String[] args) {
        Frame myFrame = new Frame("My First Frame");
        myFrame.setSize(500, 500);
        myFrame.setVisible(true);
    }

}
```



```
ain(String[] args) {
new Frame("My First Frame");
500, 500);
le(true);
```

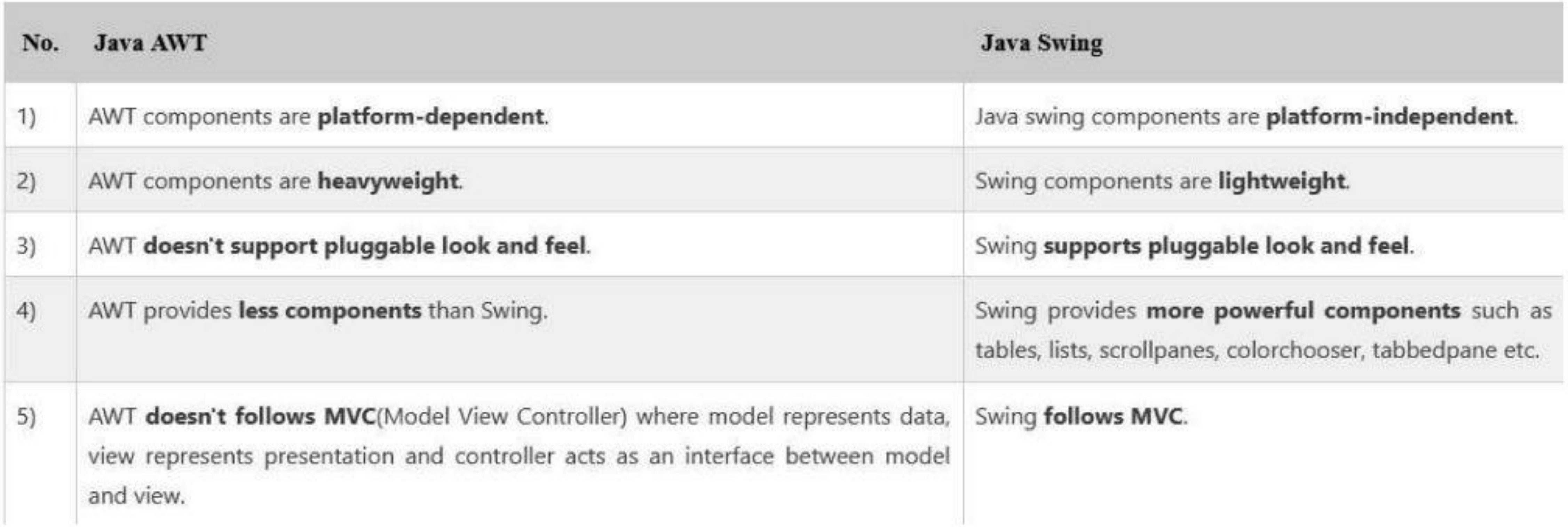
Note: The close button of the frame will not work as we have not handled this event (which can be done).To close the frame we can do □



Java Swing

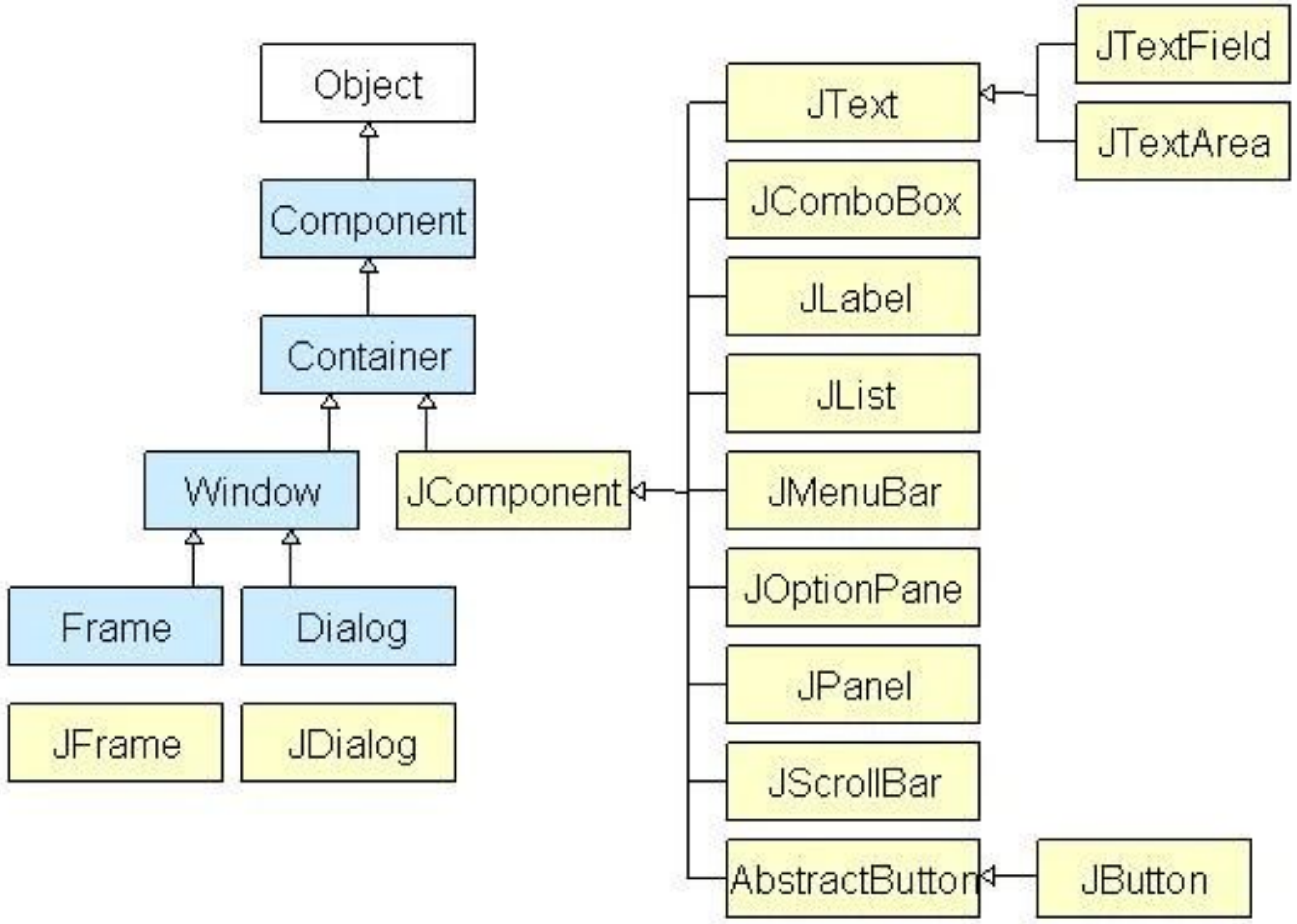
- ❖ Java Swing tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications.
- ❖ The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.
- ❖ It is built on the top of AWT API and entirely written in java.
- ❖ **Unlike AWT, Java Swing provides platform-independent and lightweight components**
- ❖ We need to import javax.swing.*;

AWT Vs Swing

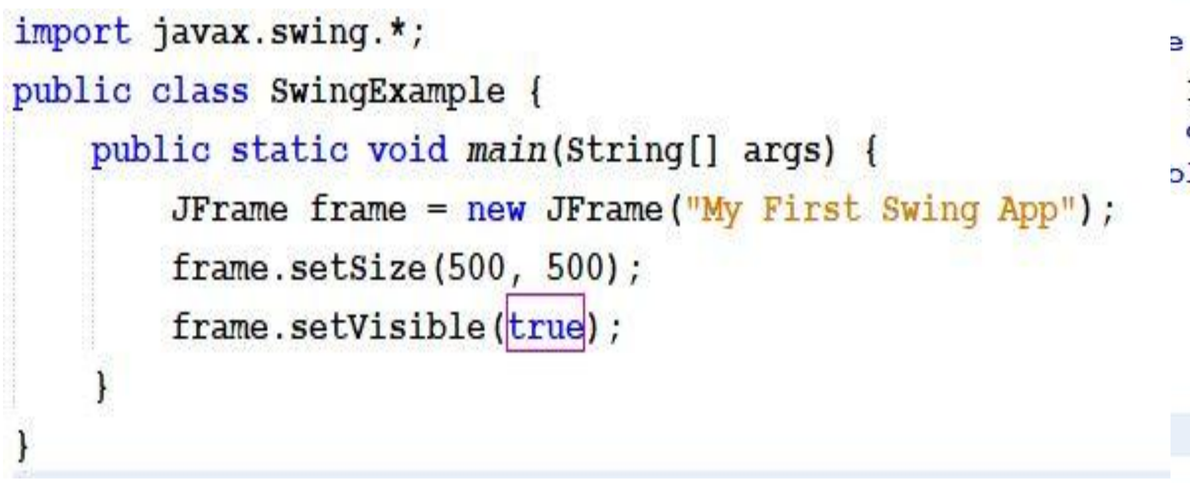


No.	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

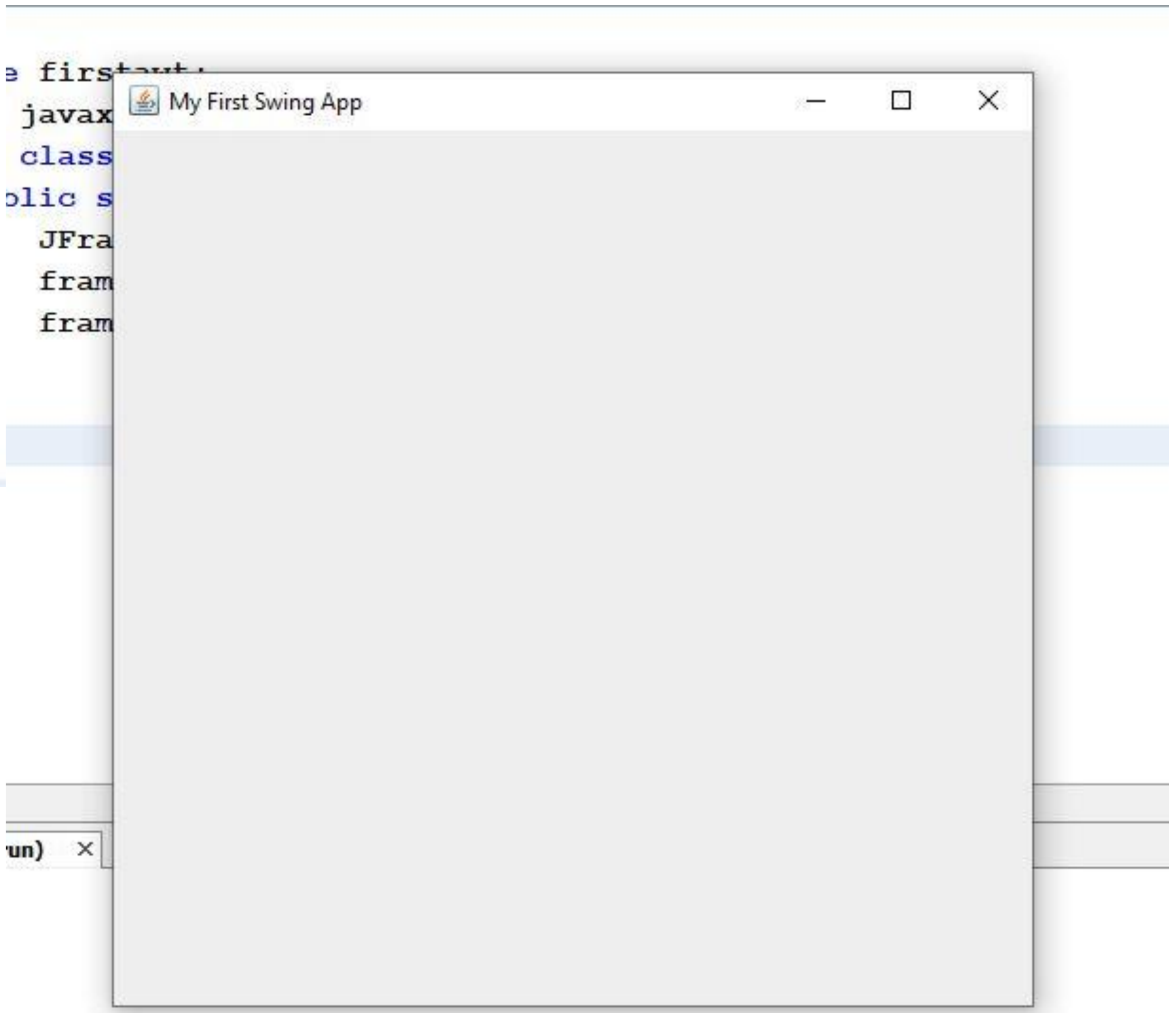
Java Swing Hierarchy



Java Swing

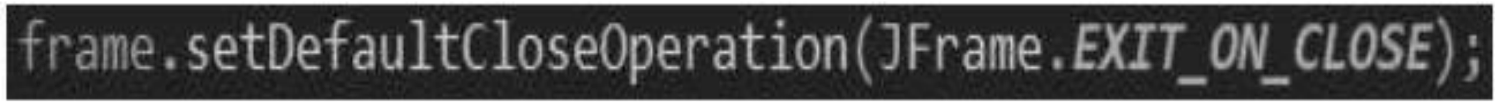


```
import javax.swing.*;
public class SwingExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My First Swing App");
        frame.setSize(500, 500);
        frame.setVisible(true);
    }
}
```



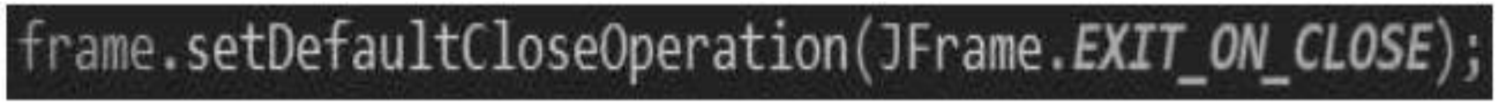
JFrame

- ❖ The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class.
- ❖ JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.
- ❖ JFrame Class is the top-level container that contains **content pane.** All visible components are contained in the content pane.
- ❖ Unlike Frame, JFrame has the option to hide or close the window with the help of *setDefaultCloseOperation(int)* method.
- ❖ Closing the frame will not close the java program, just hide the frame. To close the application,



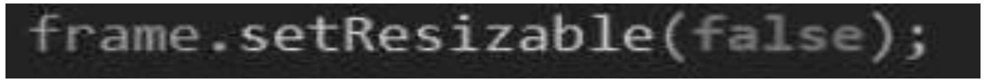
```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JFrame



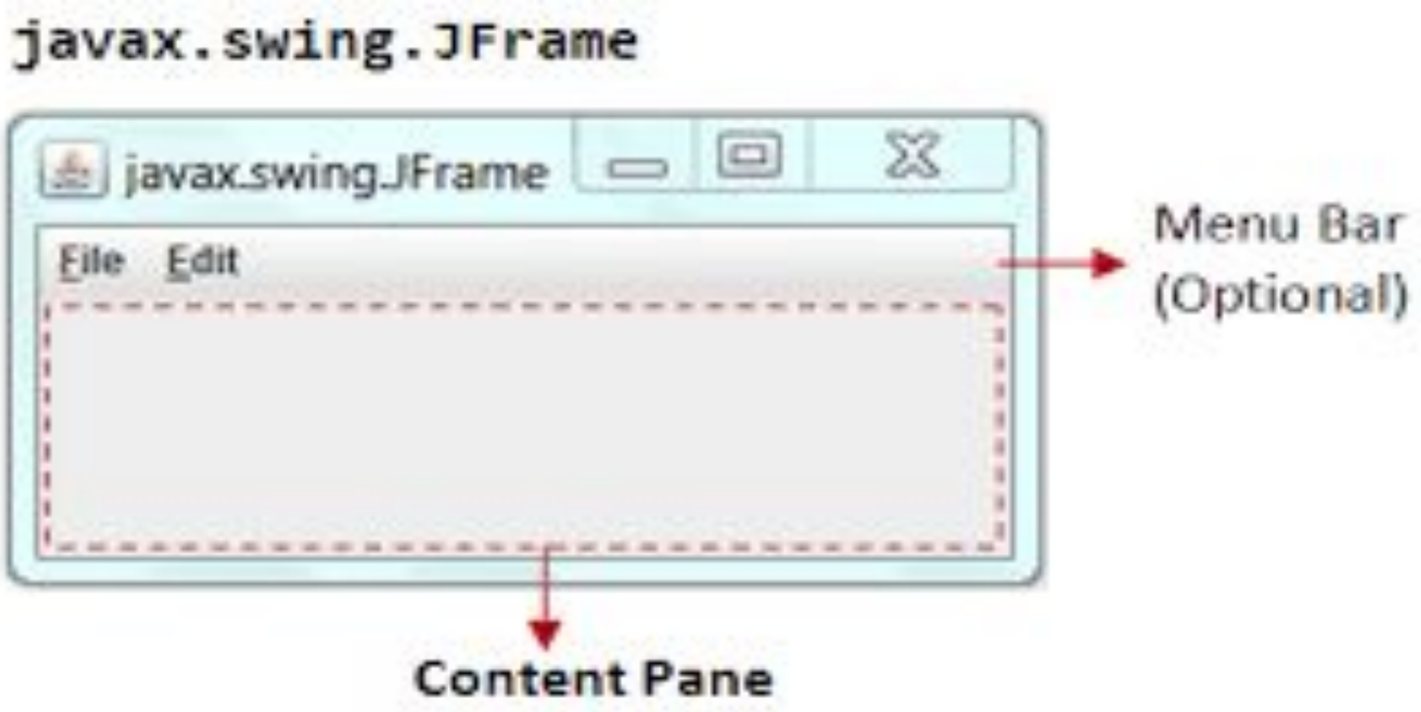
```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- ❖ By default, it uses *HIDE_ON_CLOSE,* we can also write *DO_NOTHING_ON_CLOSE* which does not allow user to close the window.
- ❖ We can also prevent user from resizing the frame

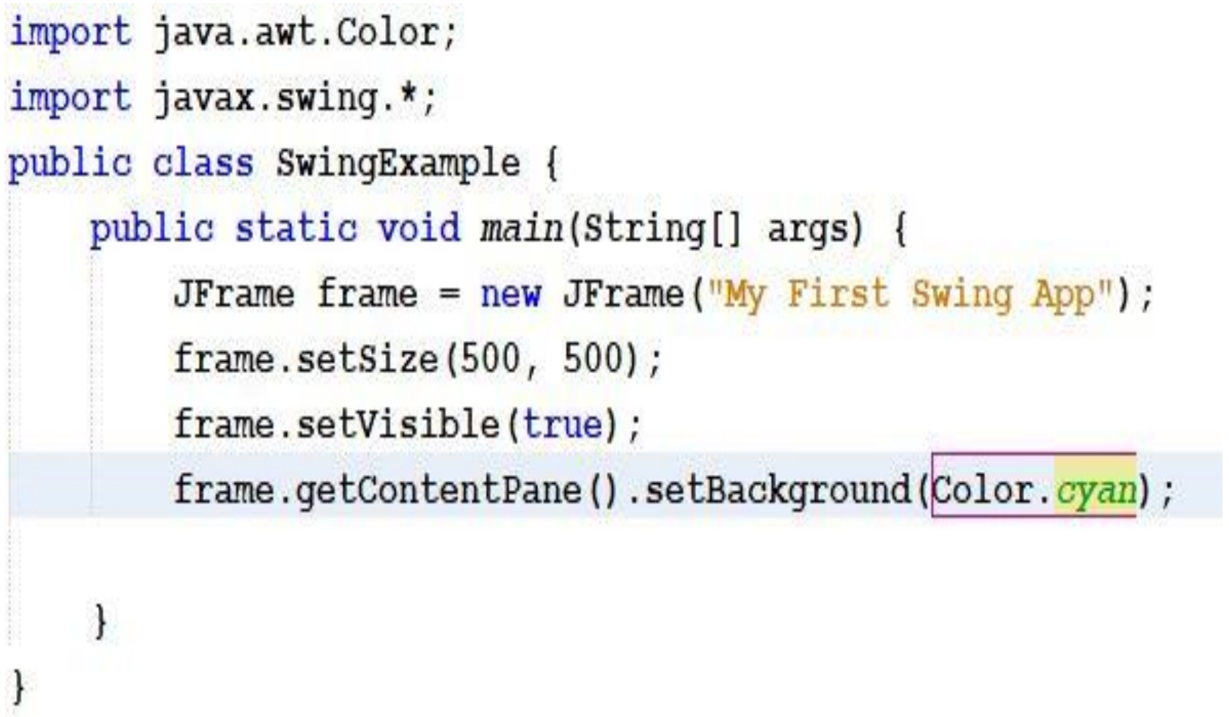


```
frame.setResizable(false);
```

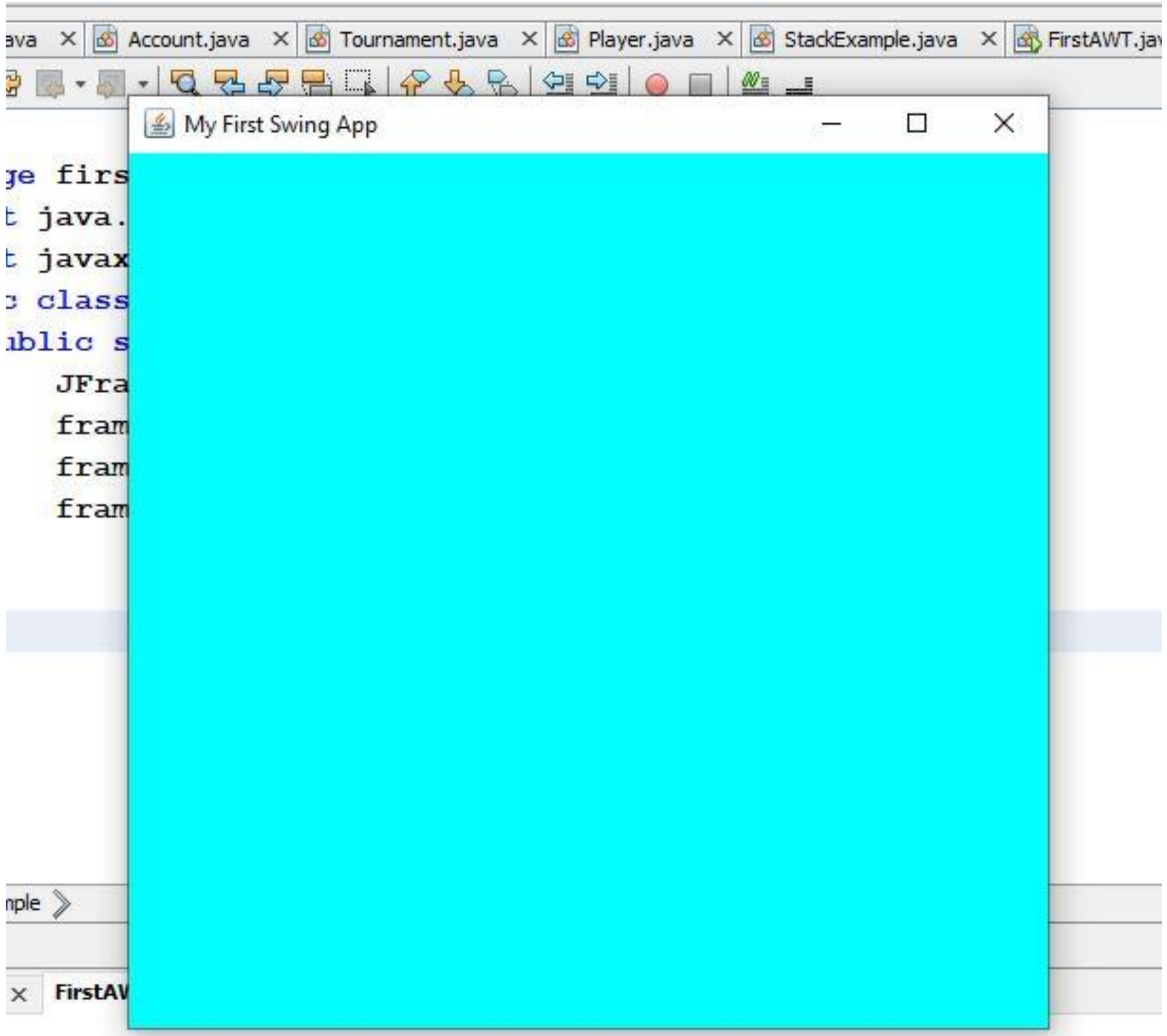
Jframe-Content Pane



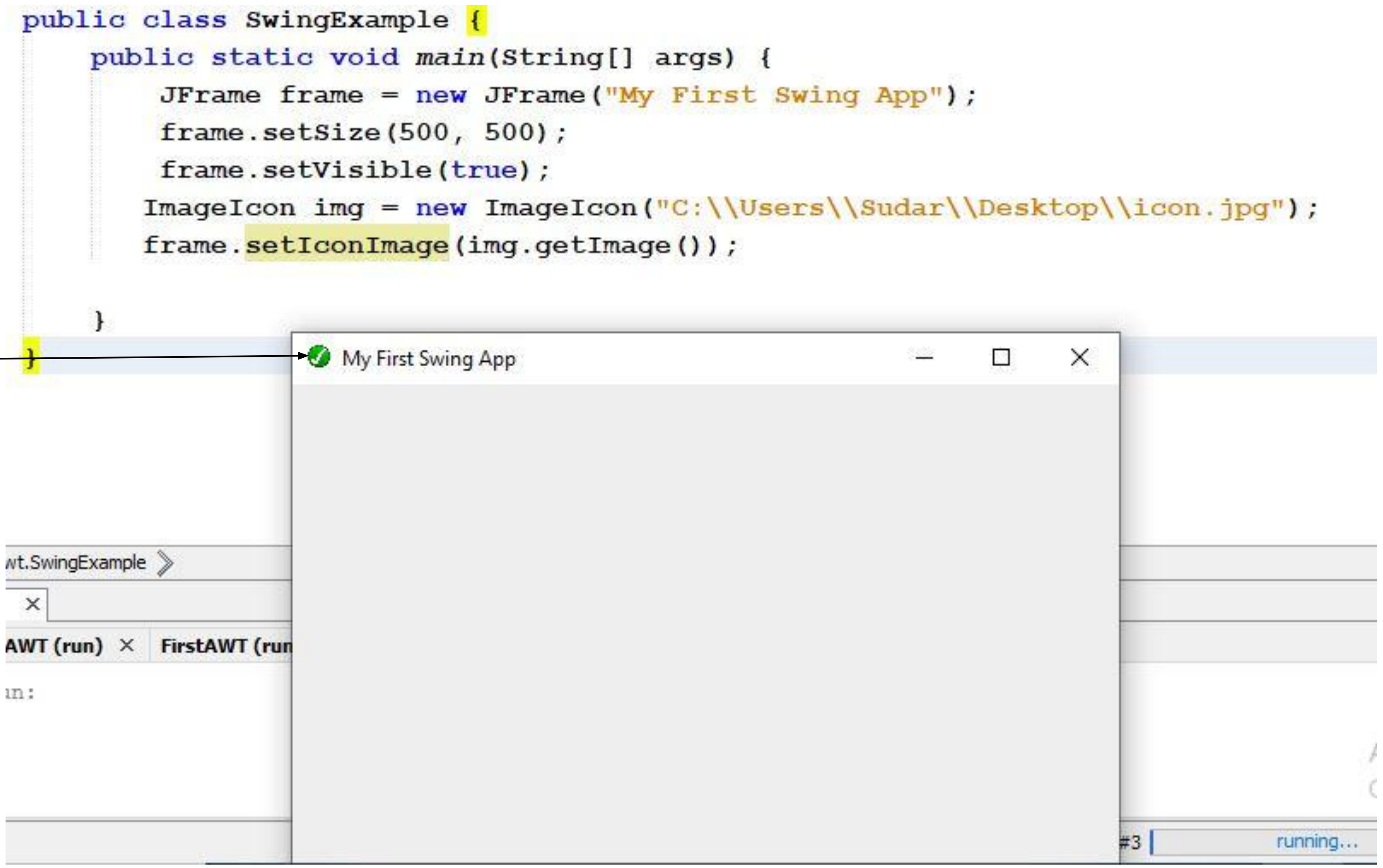
Jframe-Setting Background color



```java
import java.awt.Color;
import javax.swing.*;
public class SwingExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My First Swing App");
        frame.setSize(500, 500);
        frame.setVisible(true);
        frame.getContentPane().setBackground(Color.cyan);

    }
}
```



JFrame-Changing Icon on the top

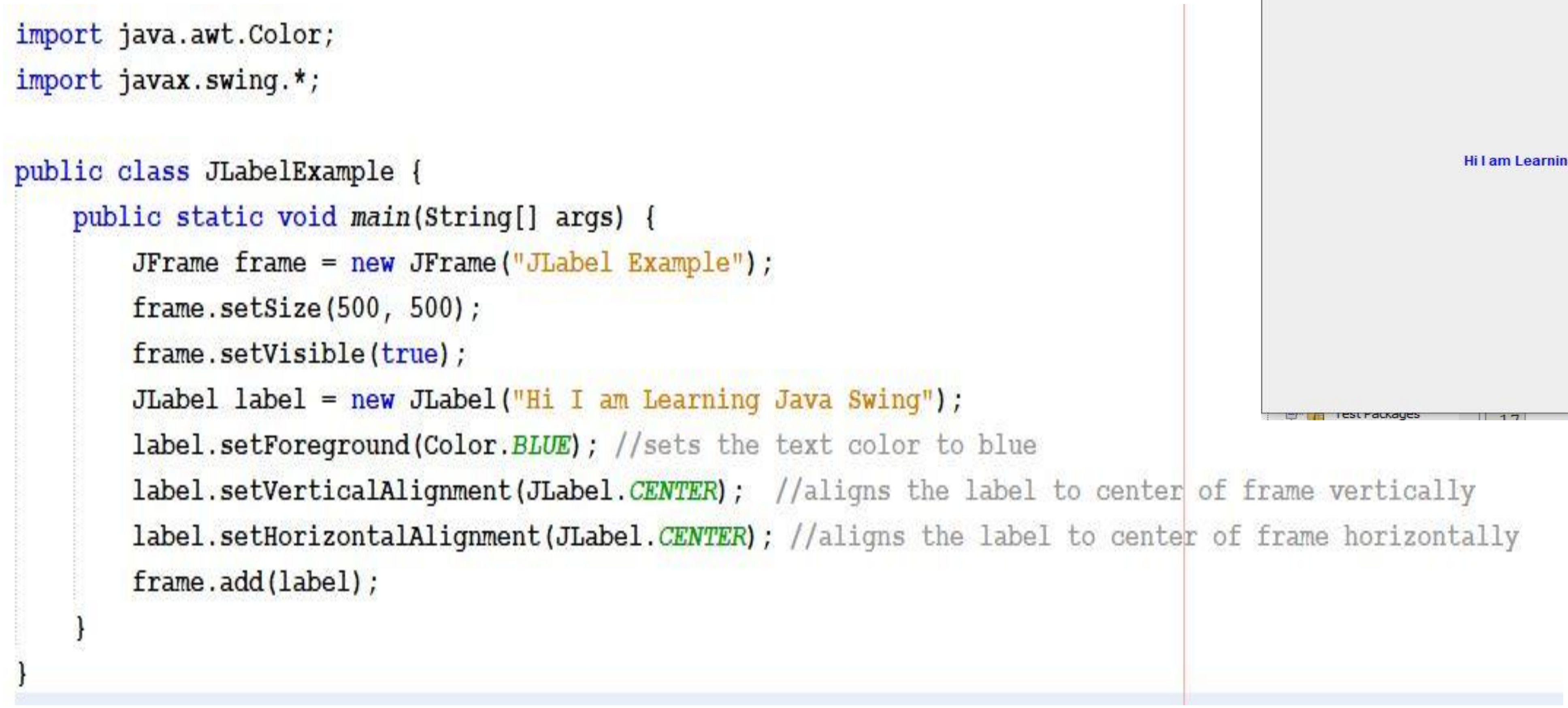


```
public class SwingExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My First Swing App");
        frame.setSize(500, 500);
        frame.setVisible(true);
       ImageIcon img = new ImageIcon("C:\\Users\\Sudar\\Desktop\\icon.jpg");
       frame.setIconImage(img.getImage());

    }
}
```

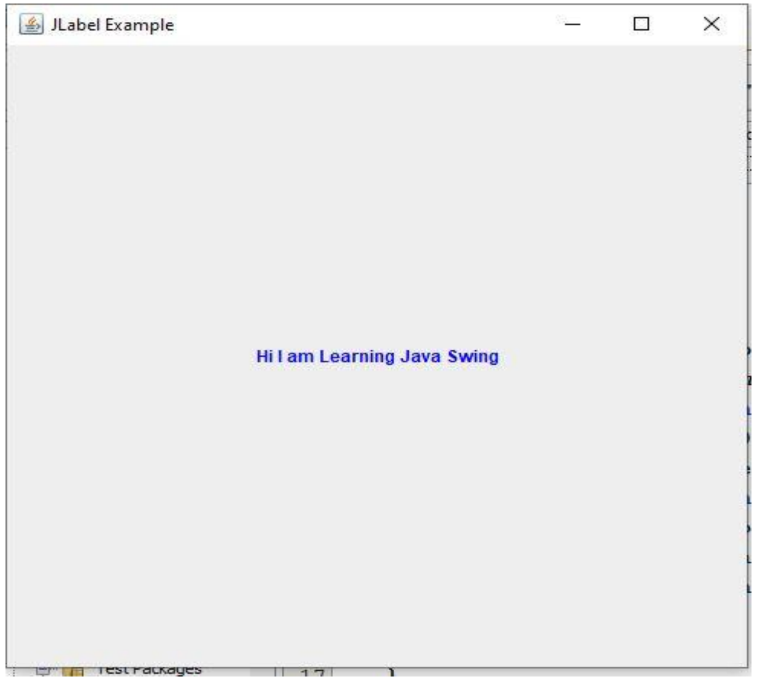
Icon Changed

Swing Components-JLabel

- ❖ It is used to display a single line of read only text.
- ❖ We define a JLabel and add it to our frame.

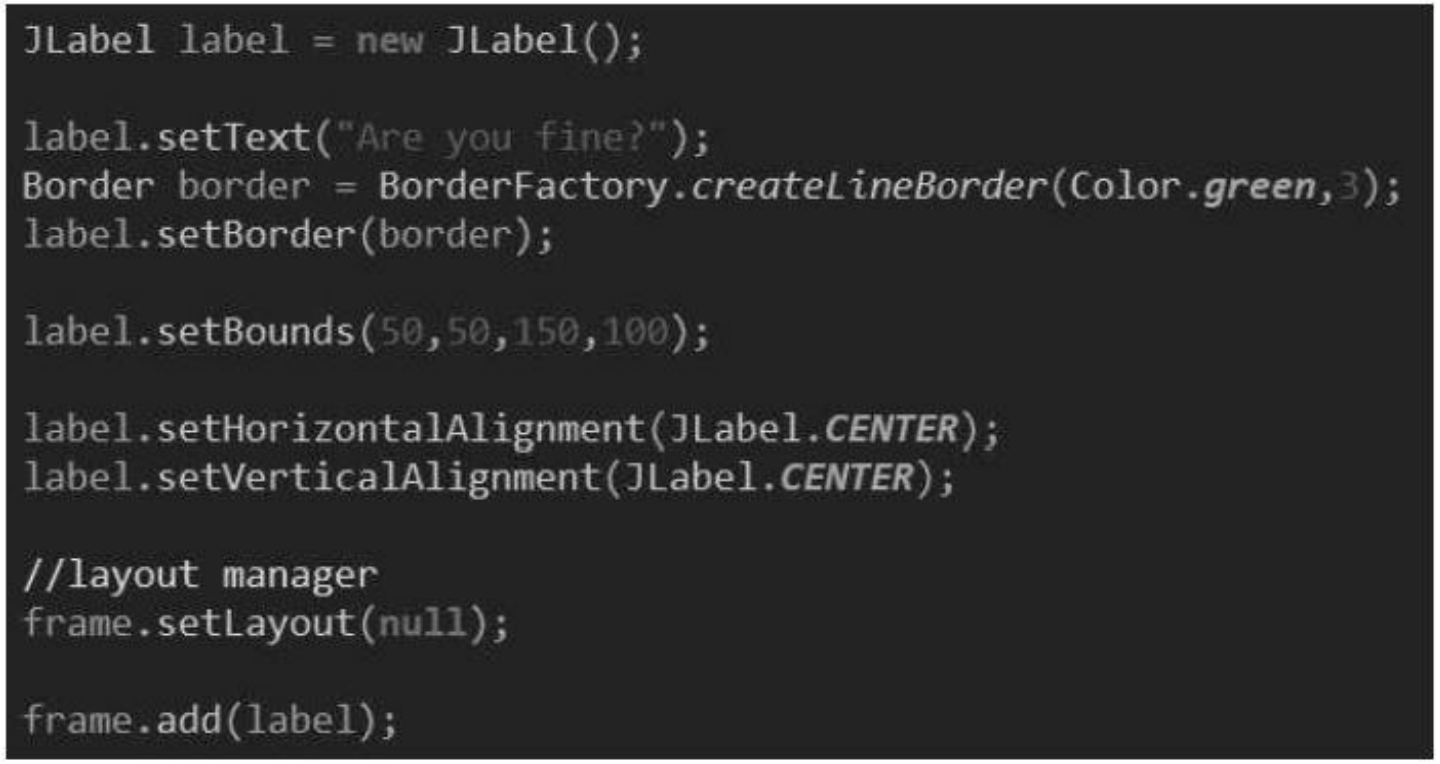


```
import java.awt.Color;
import javax.swing.*;

public class JLabelExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("JLabel Example");
        frame.setSize(500, 500);
        frame.setVisible(true);
        JLabel label = new JLabel("Hi I am Learning Java Swing");
        label.setForeground(Color.BLUE); //sets the text color to blue
        label.setVerticalAlignment(JLabel.CENTER);  //aligns the label to center of frame vertically
        label.setHorizontalAlignment(JLabel.CENTER); //aligns the label to center of frame horizontally
        frame.add(label);
    }
}
```

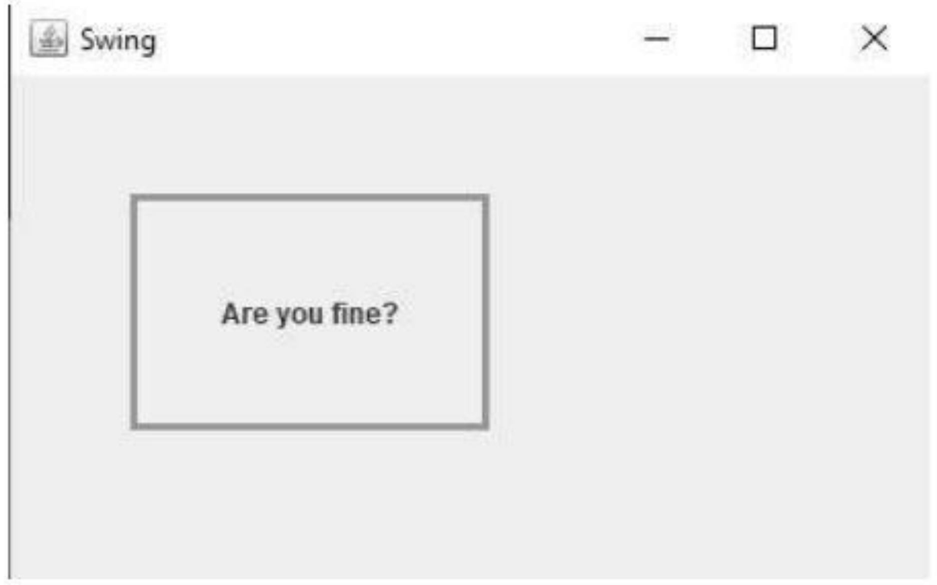


```
Hi I am Learning Java Swing
```

Swing Components-Jlabel Example



```
JLabel label = new JLabel();

label.setText("Are you fine?");
Border border = BorderFactory.createLineBorder(Color.green,3);
label.setBorder(border);

label.setBounds(50,50,150,100);

label.setHorizontalAlignment(JLabel.CENTER);
label.setVerticalAlignment(JLabel.CENTER);

//layout manager
frame.setLayout(null);

frame.add(label);
```

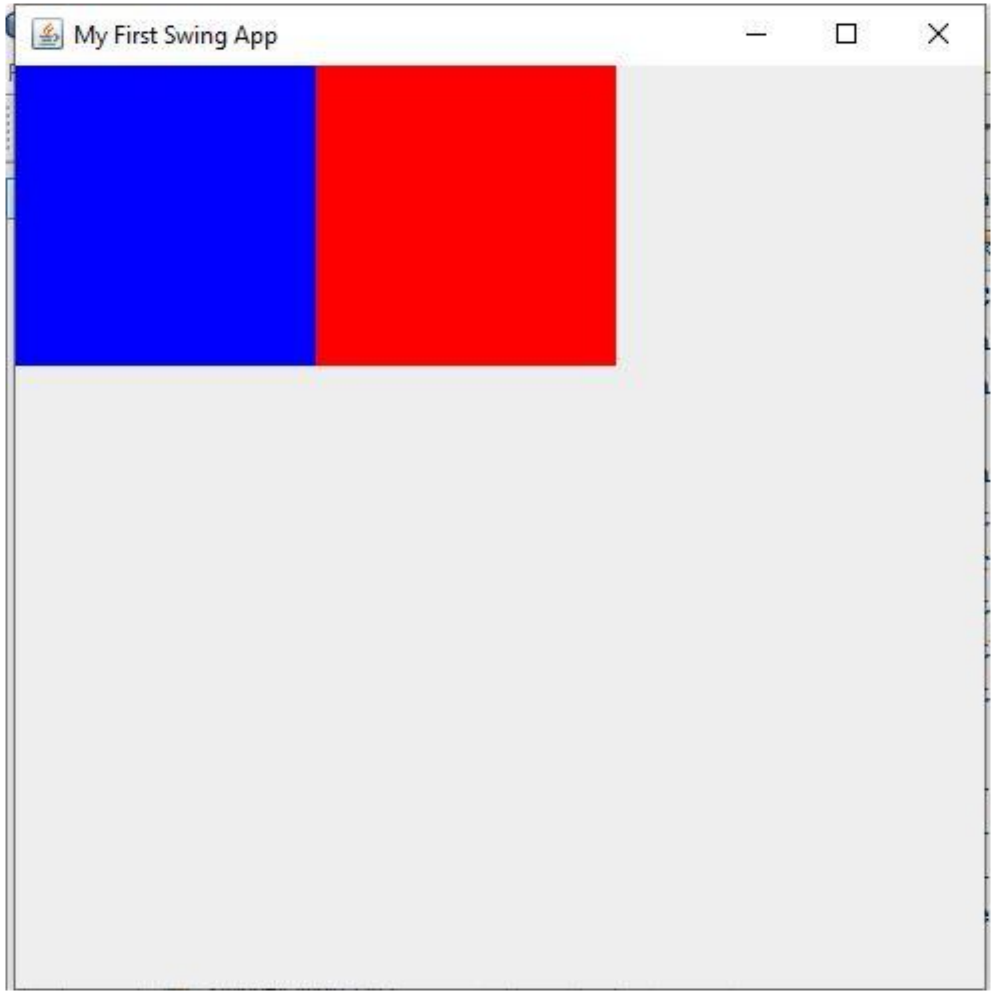


setBounds() method: The **setBounds()** method needs four arguments. The first two arguments are **x and y coordinates** of the **top-left corner** of the component, the third argument is the **width** of the component and the fourth argument is the **height** of the component.

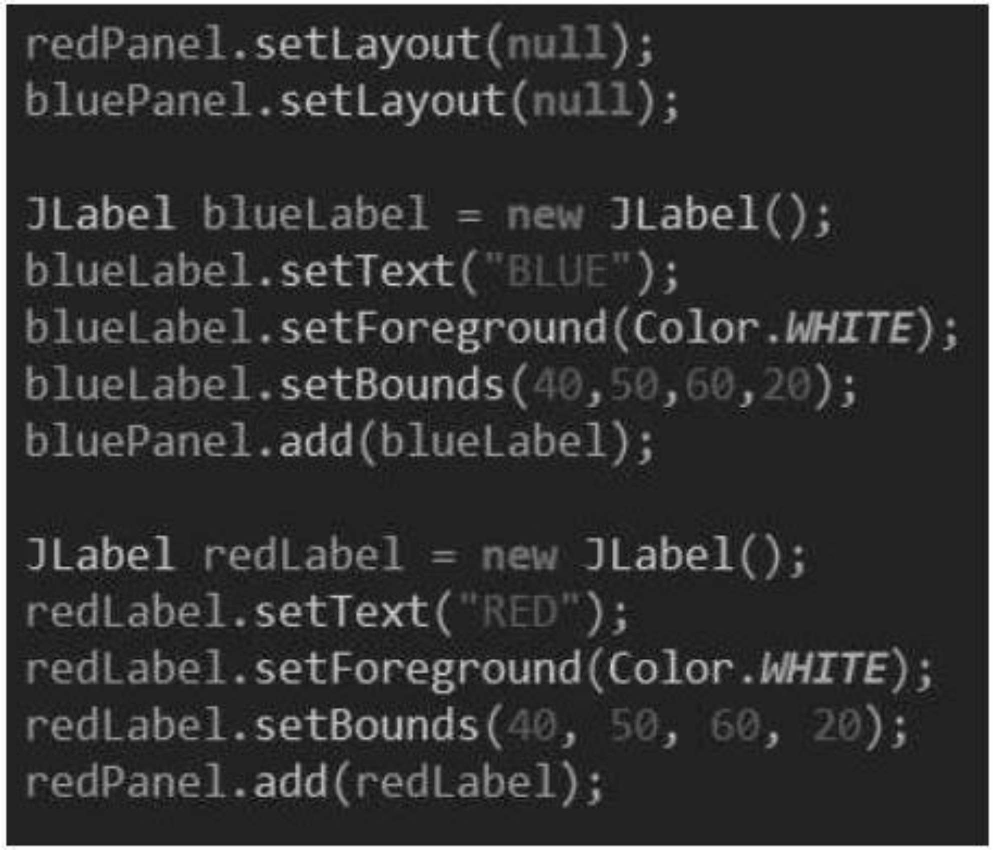
Swing Components-JPanel Example

- The JPanel is a simplest container class.
- It provides space in which an application can attach any other component.

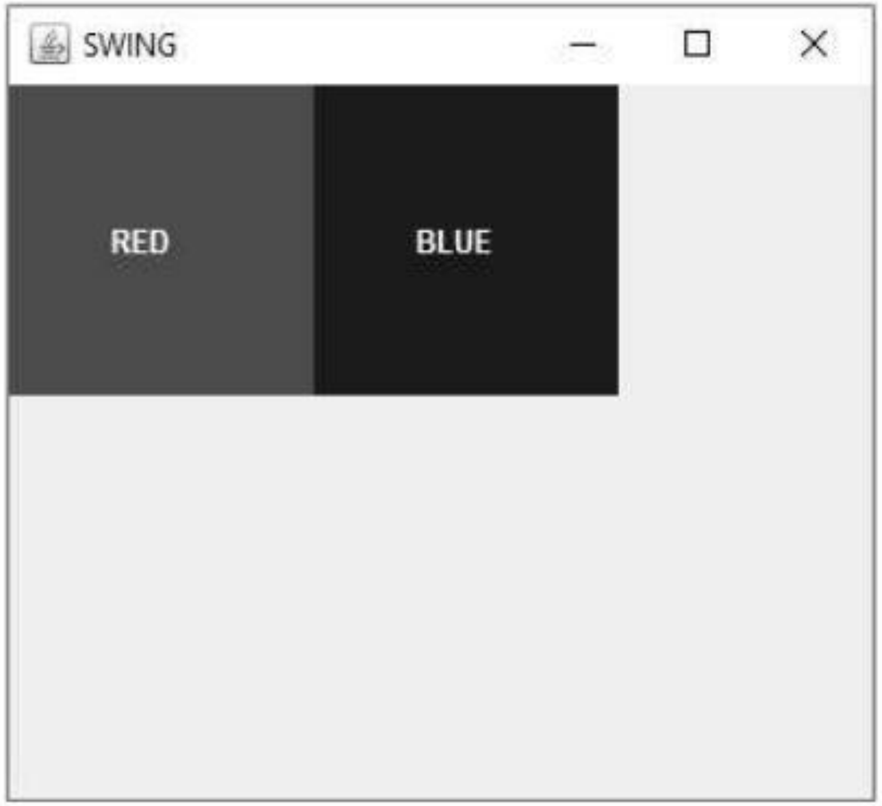
```
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JPanelExample {
        public static void main(String[] args) {
        JFrame frame = new JFrame("My First Swing App");
        frame.setSize(500, 500);
        frame.setVisible(true);
        frame.setLayout(null); //to set the layout

        JPanel bluePanel = new JPanel();
        bluePanel.setBackground(Color.BLUE);
        bluePanel.setBounds(0, 0, 150, 150);
        JPanel redPanel = new JPanel();
        redPanel.setBackground(Color.RED);
        redPanel.setBounds(150, 0, 150, 150);
        frame.add(bluePanel);
        frame.add(redPanel);
    }
}
```



Swing Components-JPanel Example

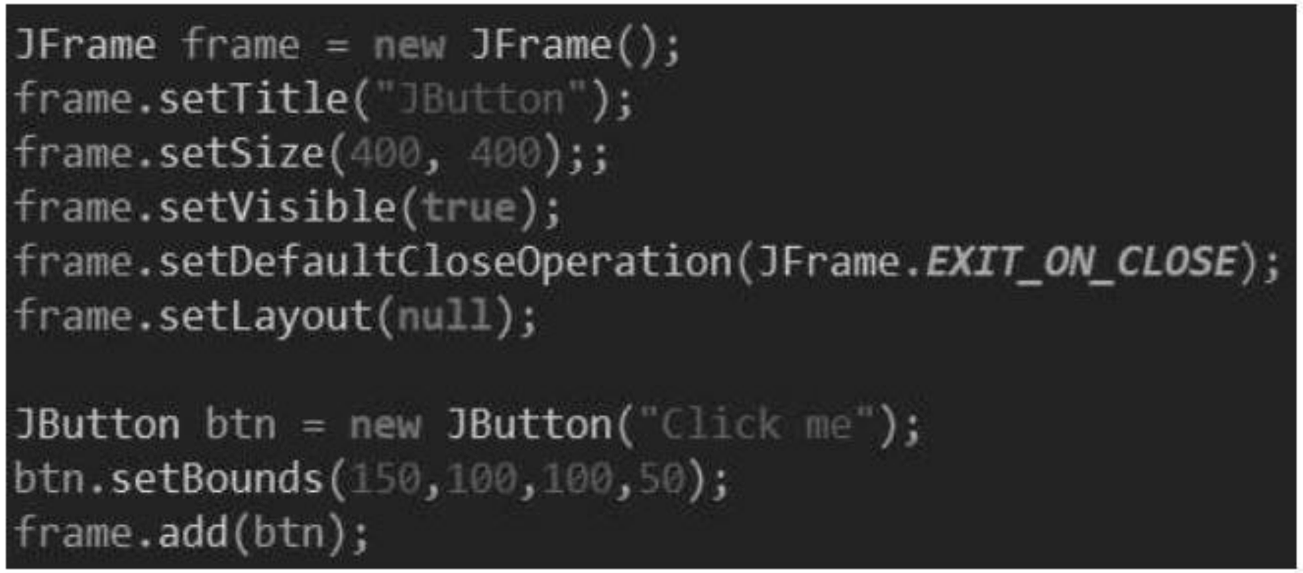


```
redPanel.setLayout(null);
bluePanel.setLayout(null);

JLabel blueLabel = new JLabel();
blueLabel.setText("BLUE");
blueLabel.setForeground(Color.WHITE);
blueLabel.setBounds(40,50,60,20);
bluePanel.add(blueLabel);

JLabel redLabel = new JLabel();
redLabel.setText("RED");
redLabel.setForeground(Color.WHITE);
redLabel.setBounds(40, 50, 60, 20);
redPanel.add(redLabel);
```

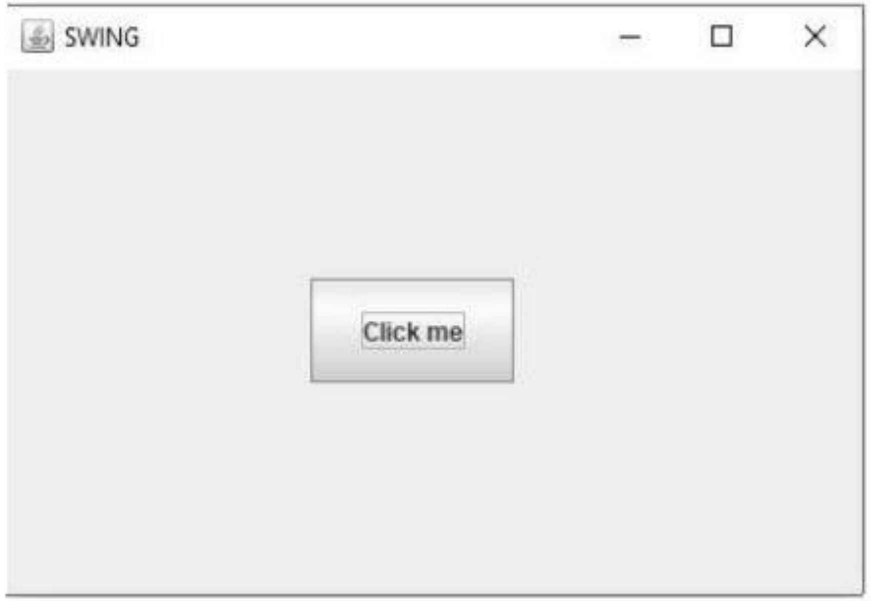


Swing Components-JButton

❖ The JButton class is used to create a labeled button that has platform independent implementation.



```
JFrame frame = new JFrame();
frame.setTitle("JButton");
frame.setSize(400, 400);;
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout(null);

JButton btn = new JButton("Click me");
btn.setBounds(150,100,100,50);
frame.add(btn);
```



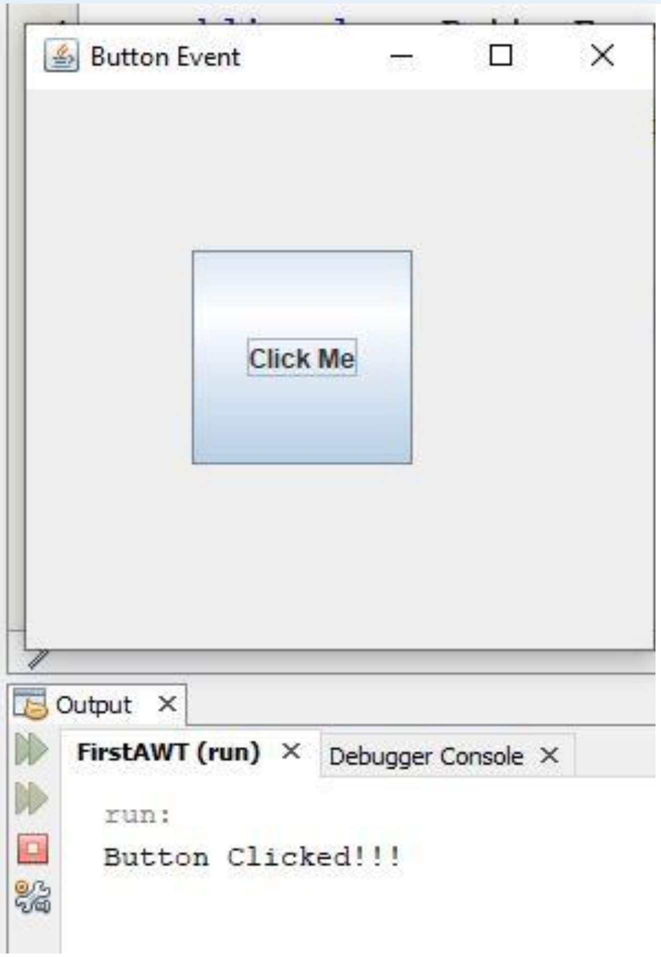
Event Handling in Java Swing

1. **Event:** An event is a signal to the program that something has happened. It can be triggered by typing in a text field, selecting an item from the menu etc.
2. **Event handler:** The code that performs a task in response to an event. is called event handler.
3. **Event handling:** It is process of responding to events that can occur at any time during execution of a program.
4. **Event Source:** It is an object that generates the event(s). Usually the event source is a button or the other component that the user can click but any Swing component can be an event source.
5. **Event Listener:** It is an object that watch for (i.e. listen for) events and handles them when they occur.
6. **Listener interface**: It is an interface which contains methods that the listener must implement and the source of the event invokes when the event occurs.

JButton Events

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

public class JButtonEvent implements ActionListener{
    public JButtonEvent(){
        JFrame frame = new JFrame("Button Event");
        frame.setSize(300,300);
        frame.setVisible(true);
        frame.setLayout(null);

        JButton btn = new JButton("Click Me");
        btn.setBounds(75, 75, 100, 100);
        btn.addActionListener(this);
        frame.add(btn);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Button Clicked!!!");
    }
}
```

```java
public class ButtonEventMain {
    public static void main(String[] args) {
        new JButtonEvent();
    }
}
```

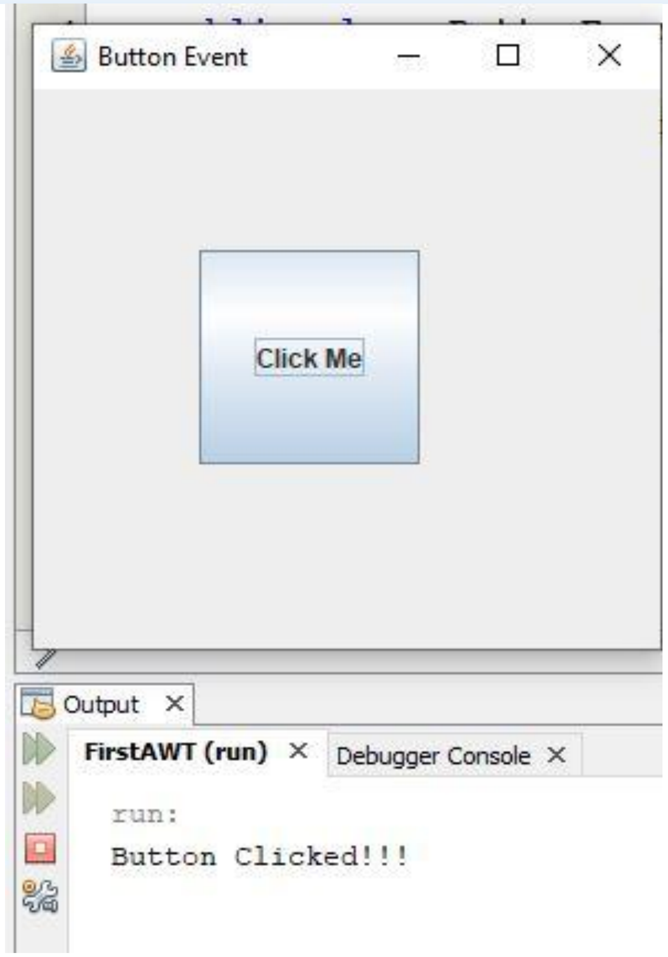


```
run:
Button Clicked!!!
```

JButton Events

```java
public class ButtonEventMain {
    public static void main(String[] args) {
        new JButtonEvent();
    }
}
```

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

public class JButtonEvent extends JFrame implements ActionListener{
    public JButtonEvent(){
        new JFrame("Button Event");
        this.setSize(300,300);
        this.setVisible(true);
        this.setLayout(null);

        JButton btn = new JButton("Click Me");
        btn.setBounds(75, 75, 100, 100);
        btn.addActionListener(this);
        this.add(btn);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Button Clicked!!!");
    }
}
```



```
run:
Button Clicked!!!
```

Layout Managers

- ❖ The Layout Managers are used to arrange components in a particular manner
- ❖ LayoutManager is an interface that is implemented by all the classes of layout managers.
- ❖ There are following classes that represents the layout managers:

a. java.awt.BorderLayout
b. java.awt.FlowLayout
c. java.awt.GridLayout
d. java.awt.CardLayout
e. java.awt.GridBagLayout
f. javax.swing.BoxLayout
g. javax.swing.GroupLayout
h. javax.swing.ScrollPaneLayout
i. javax.swing.SpringLayout

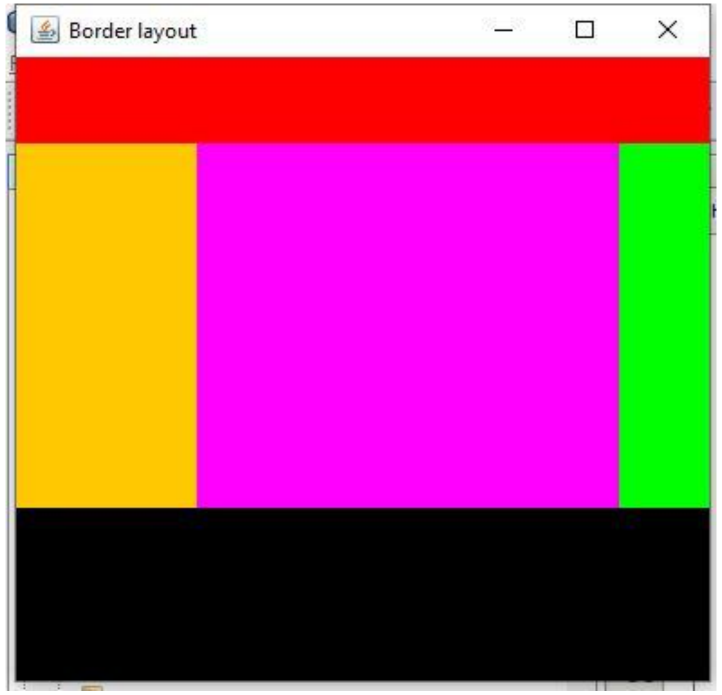
Border Layout

- ❖ The BorderLayout is used to arrange the components in five regions: north, south, east, west and center.
- ❖ All extra space is placed in center area
- ❖ Each region (area) may contain one component only. It is the default layout of frame or window.
- ❖ BorderLayout provides five constants for each region: NORTH, SOUTH, EAST, WEST
- ❖ On frame resize, top and bottom will expand horizontally ; left and right will expand vertically

Border Layout-Example

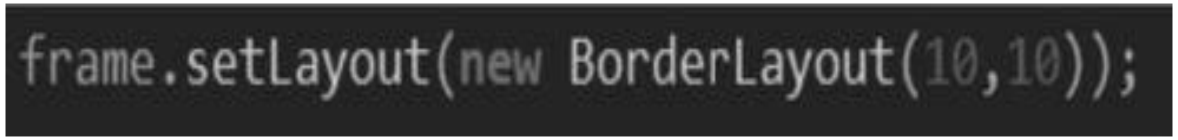
```java
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JavaBorderLayout {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Border layout");
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.setLayout(new BorderLayout());

        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        JPanel p3 = new JPanel();
        JPanel p4 = new JPanel();
        JPanel p5 = new JPanel();
```

```java
        p1.setBackground(Color.red);
        p1.setPreferredSize(new Dimension(100,50)); //setting dimension of p1
        p2.setBackground(Color.BLACK);
        p2.setPreferredSize(new Dimension(50,100));
        p3.setBackground(Color.GREEN);
        p3.setPreferredSize(new Dimension(50,100));
        p4.setBackground(Color.ORANGE);
        p4.setPreferredSize(new Dimension(100,50));
        p5.setBackground(Color.MAGENTA);

        frame.add(p1, BorderLayout.NORTH);
        frame.add(p2, BorderLayout.SOUTH);
        frame.add(p3, BorderLayout.EAST);
        frame.add(p4, BorderLayout.WEST);
        frame.add(p5, BorderLayout.CENTER);
    }
}
```

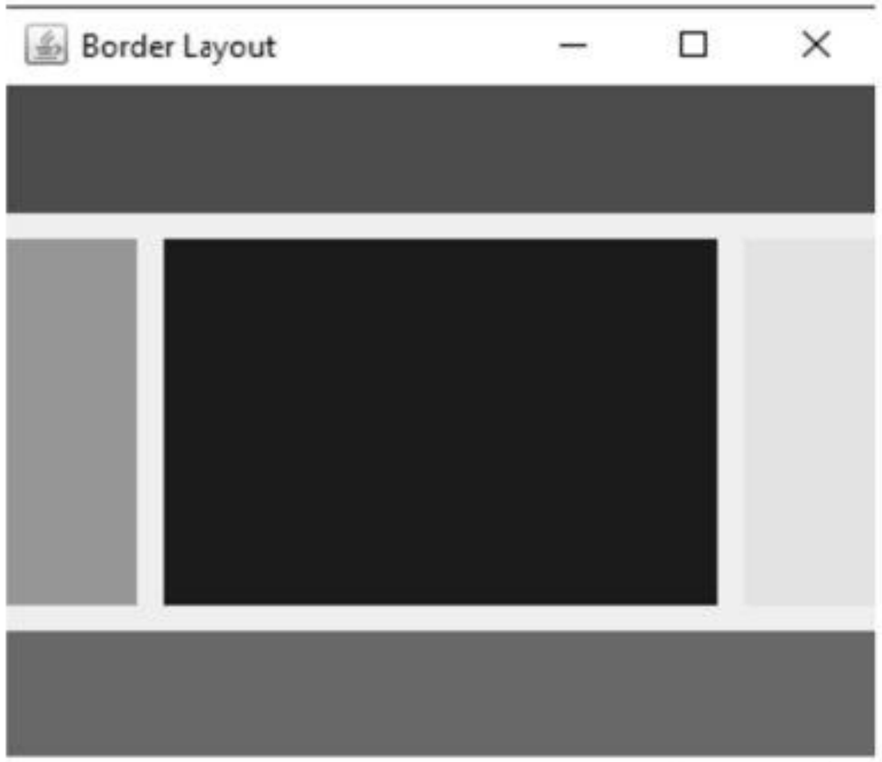


Border Layout-Specifying Gaps between border

❖ In order to specify gaps between components, we need to pass horizontal and vertical gap (in pixels) in the BorderLayout Constructor.



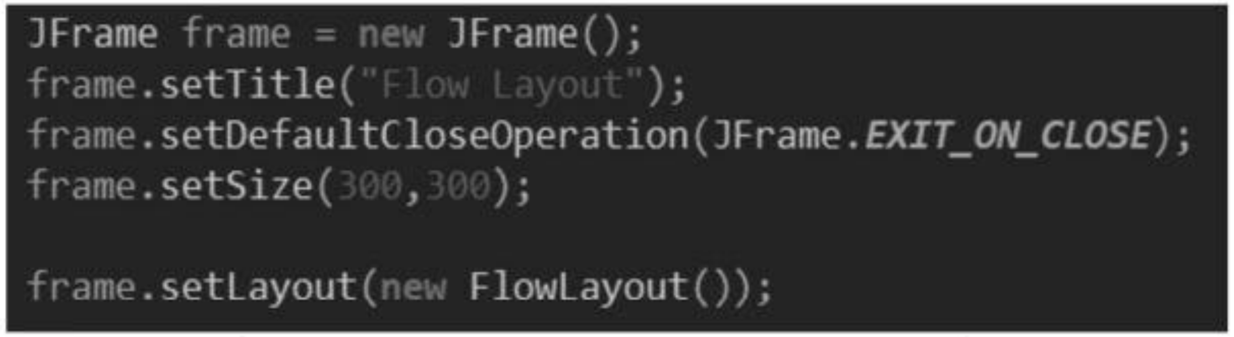
```
frame.setLayout(new BorderLayout(10,10));
```



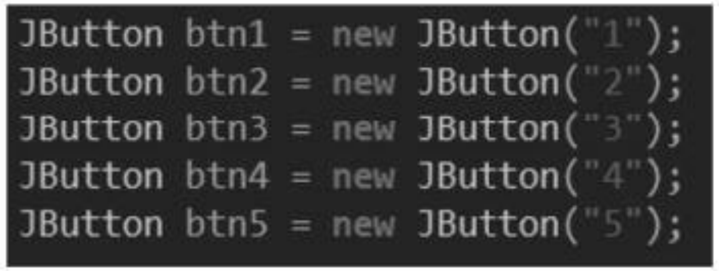
Flow Layout

- ❖ The FlowLayout is used to arrange the components in a row, one aêer another (in a flow). It is the default layout of applet or panel
- ❖ FlowLayout provides five constants for each region: LEFT, RIGHT, CENTER, LEADING, TRAILING
- ❖ Constructors of FlowLayout class
  a. FlowLayout(): creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
  b. FlowLayout(int align): creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
  c. FlowLayout(int align, int hgap, int vgap): creates a flow layout with the given alignment and the given horizontal and vertical gap.

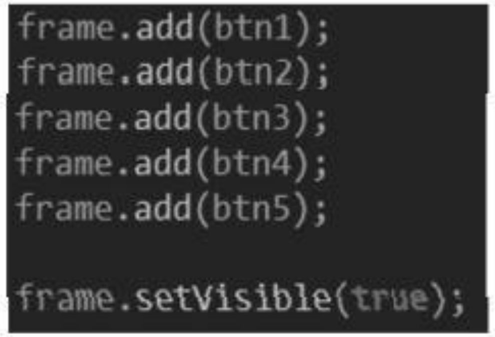
Flow Layout



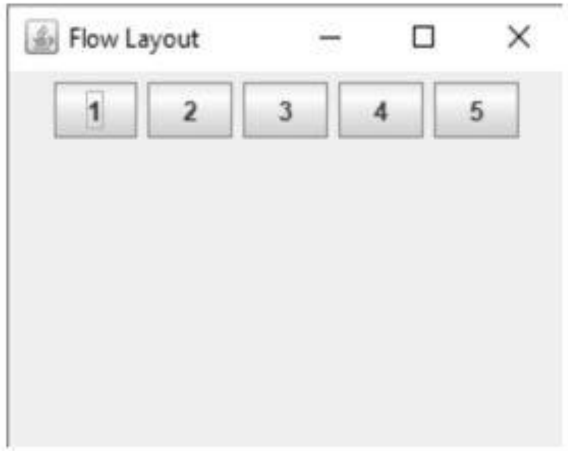
```
JFrame frame = new JFrame();
frame.setTitle("Flow Layout");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(300,300);

frame.setLayout(new FlowLayout());
```

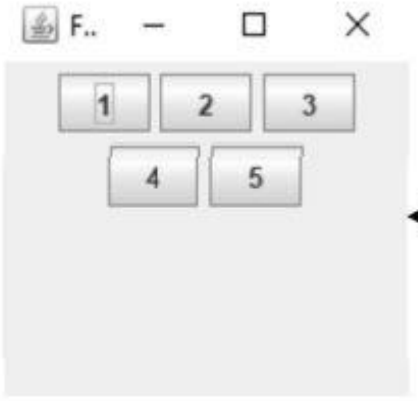


```
JButton btn1 = new JButton("1");
JButton btn2 = new JButton("2");
JButton btn3 = new JButton("3");
JButton btn4 = new JButton("4");
JButton btn5 = new JButton("5");
```



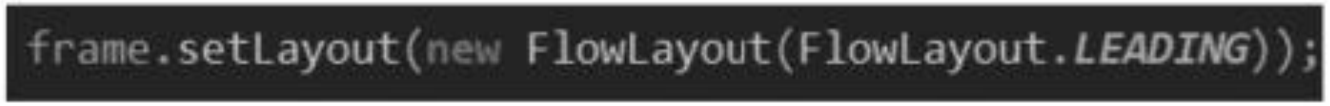
```
frame.add(btn1);
frame.add(btn2);
frame.add(btn3);
frame.add(btn4);
frame.add(btn5);

frame.setVisible(true);
```

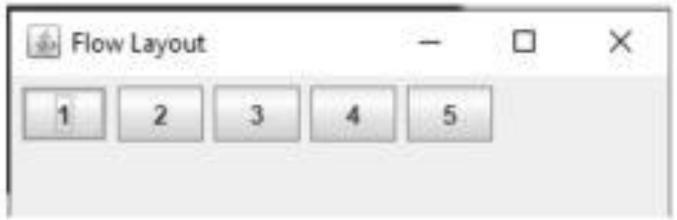


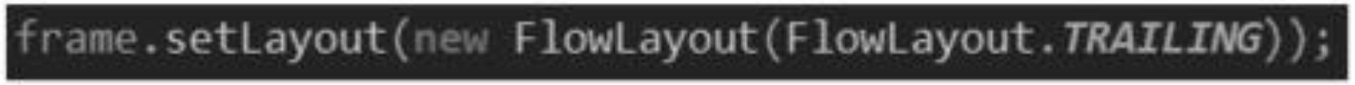


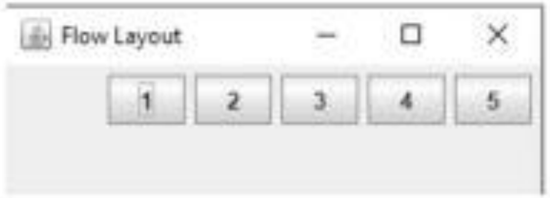
Moves to next row
when not enough spa

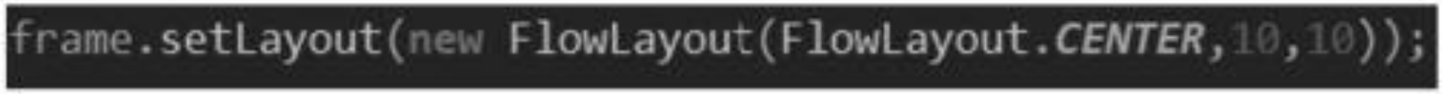
Flow Layout

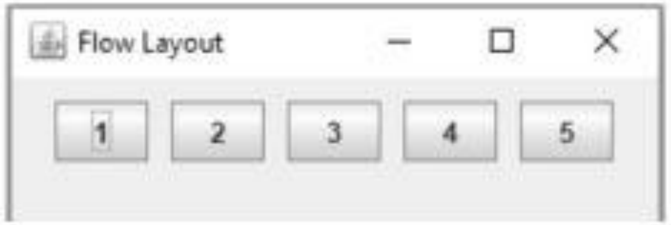

```
frame.setLayout(new FlowLayout(FlowLayout.LEADING));
```




```
frame.setLayout(new FlowLayout(FlowLayout.TRAILING));
```




```
frame.setLayout(new FlowLayout(FlowLayout.CENTER,10,10));
```



Grid Layout

- ❖ The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle.
- ❖ Constructors of GridLayout class

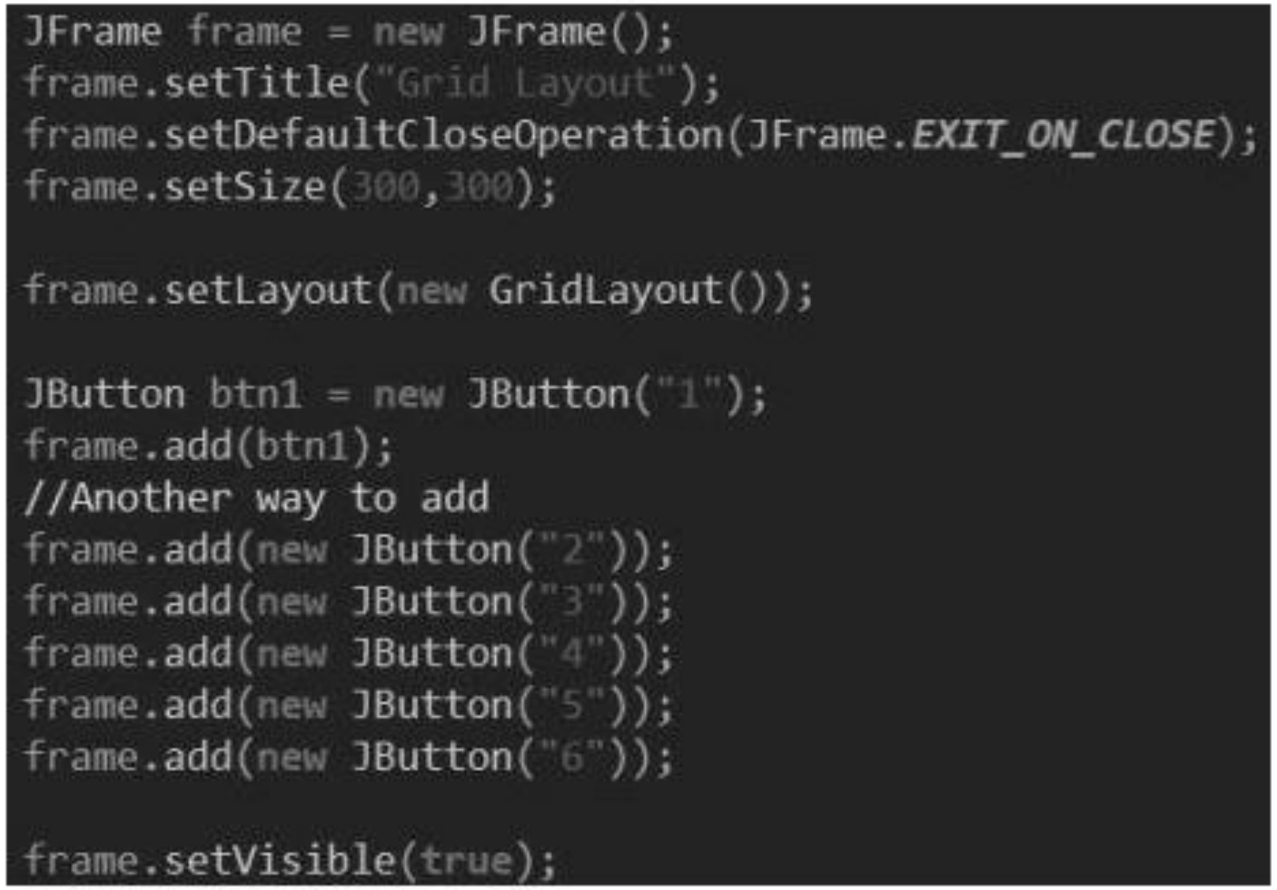
a. GridLayout(): creates a grid layout with one column per component in a row.

b. GridLayout(int rows, int columns): creates a grid layout with the given rows and columns but no gaps between the components.

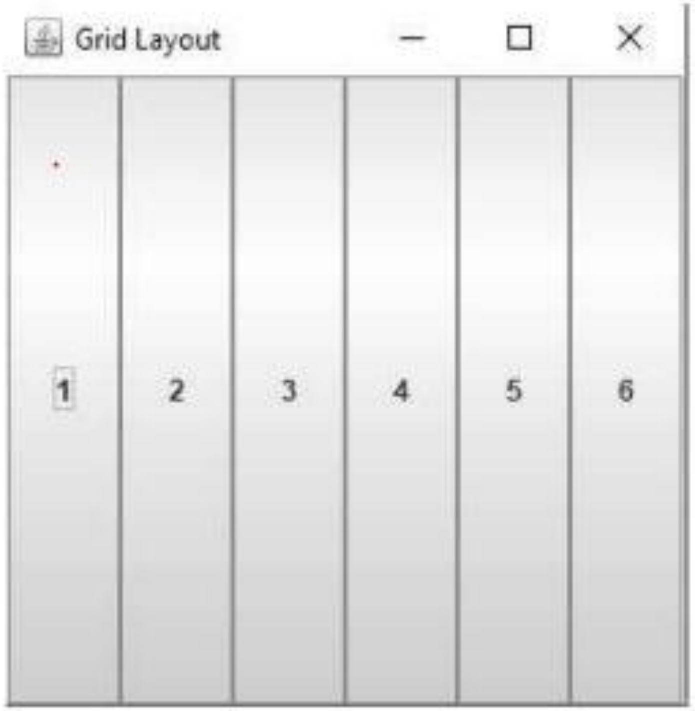
c. GridLayout(int rows, int columns, int hgap, int vgap): creates a grid layout with the given rows and columns alongwith given horizontal and vertical gaps.

Grid Layout

•

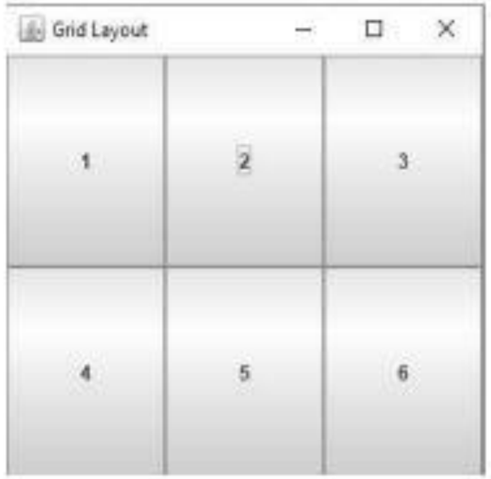


```
JFrame frame = new JFrame();
frame.setTitle("Grid Layout");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(300,300);

frame.setLayout(new GridLayout());

JButton btn1 = new JButton("1");
frame.add(btn1);
//Another way to add
frame.add(new JButton("2"));
frame.add(new JButton("3"));
frame.add(new JButton("4"));
frame.add(new JButton("5"));
frame.add(new JButton("6"));

frame.setVisible(true);
```

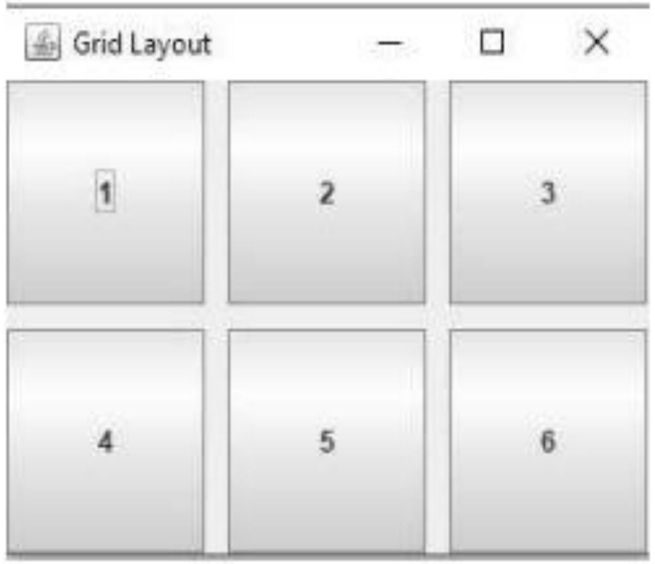


Grid Layout

```java
frame.setLayout(new GridLayout(2,3));
```

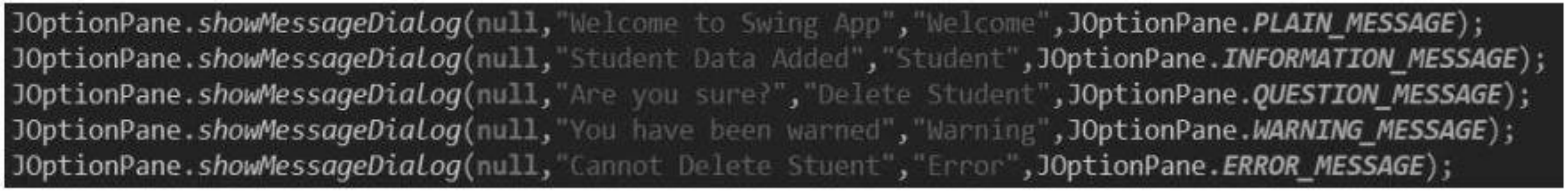


```java
frame.setLayout(new GridLayout(2,3,10,10));
```

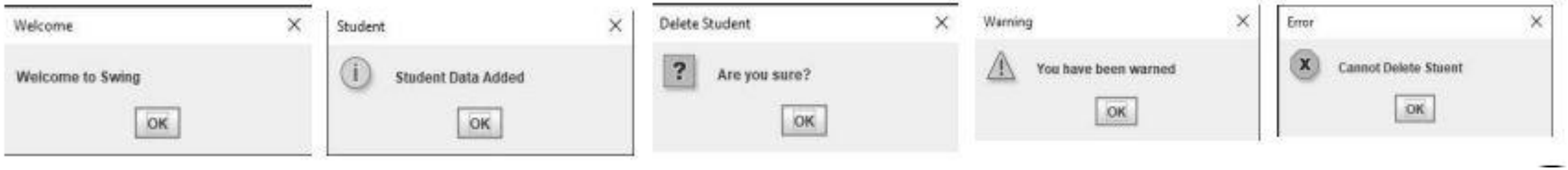


JOption Pane

- ❖ The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box.
- ❖ These dialog boxes are used to display information or get input from the user.



```
JOptionPane.showMessageDialog(null,"Welcome to Swing App","Welcome",JOptionPane.PLAIN_MESSAGE);
JOptionPane.showMessageDialog(null,"Student Data Added","Student",JOptionPane.INFORMATION_MESSAGE);
JOptionPane.showMessageDialog(null,"Are you sure?","Delete Student",JOptionPane.QUESTION_MESSAGE);
JOptionPane.showMessageDialog(null,"You have been warned","Warning",JOptionPane.WARNING_MESSAGE);
JOptionPane.showMessageDialog(null,"Cannot Delete Stuent","Error",JOptionPane.ERROR_MESSAGE);
```

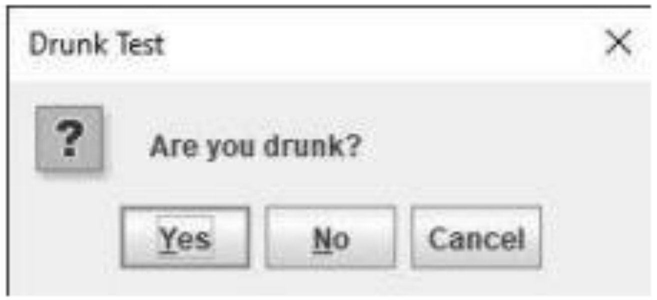


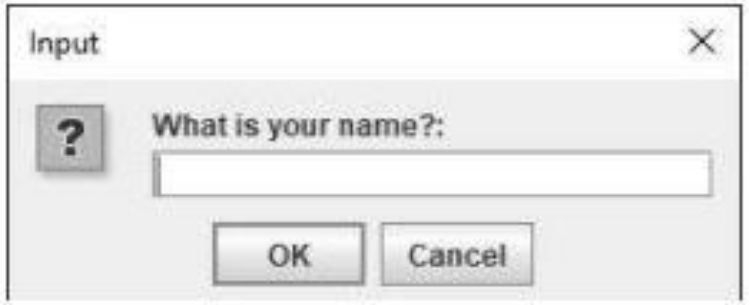
```
Welcome          Welcome to Swing        OK
Student          Student Data Added      OK
Delete Student   Are you sure?           OK
Warning          You have been warned    OK
Error            Cannot Delete Stuent    OK
```

JOptionPane

```
int res = JOptionPane.showConfirmDialog(null, "Are you drunk?","Drunk Test",JOptionPane.YES_NO_CANCEL_OPTION);
System.out.println(res);
//0 on yes, 1 on no, 2 on cancel, -1 on close
```

```
String name= JOptionPane.showInputDialog("What is your name?:");
System.out.println(name);
```

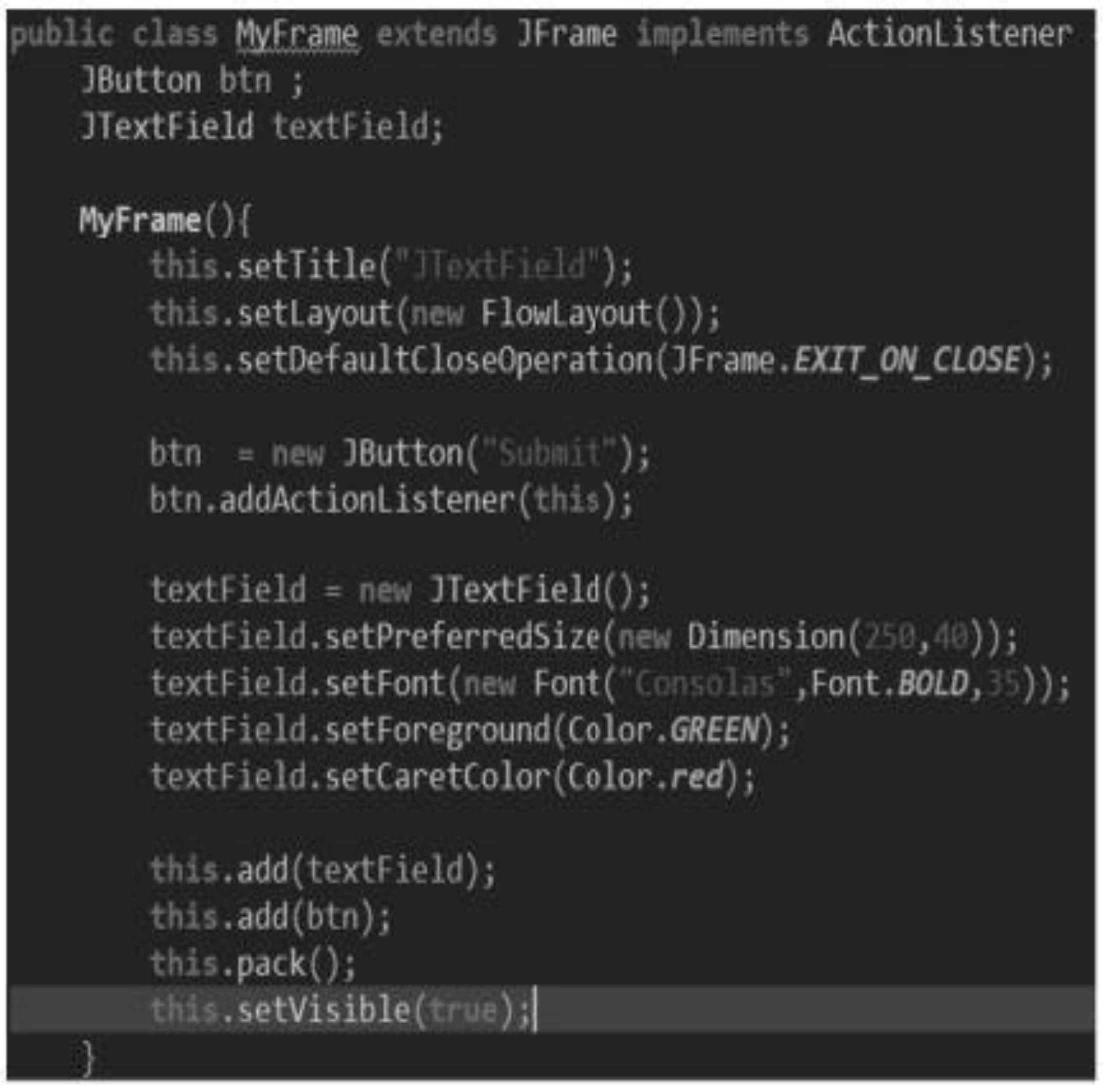




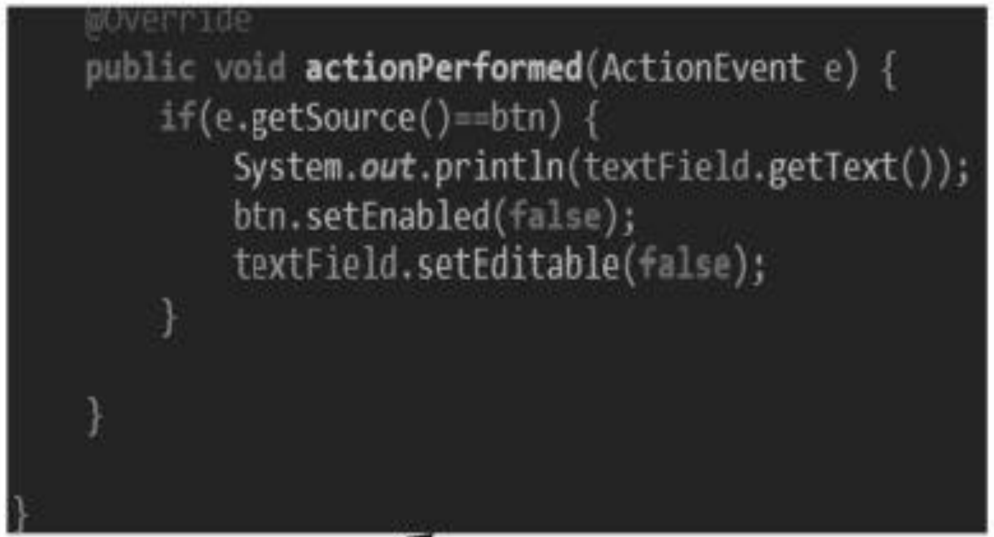
Java JTextField

- ❖ The object of a JTextField class is a text component that allows the editing of a single line text.
- ❖ It inherits JTextComponent class.
- ❖ The constructor of the class are :
  a. **JTextField()** : constructor that creates a new TextField
  b. **JTextField(int columns)** : constructor that creates a new empty TextField with specified number of columns.
  c. **JTextField(String text)** : constructor that creates a new empty text field initialized with the given string.
  d. **JTextField(String text, int columns)** : constructor that creates a new empty textField with the given string and a specified number of columns .
  e. **JTextField(Document doc, String text, int columns)** : constructor that creates a textfield that uses the given text storage model and the given number of columns.

Java JTextField

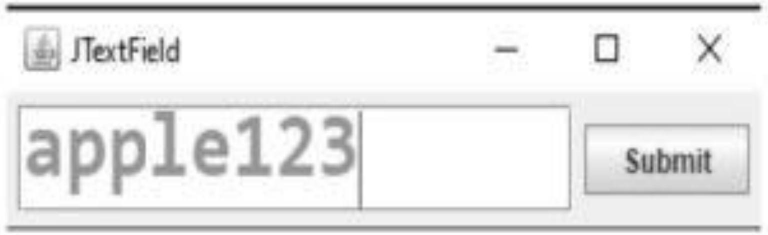


```
public class MyFrame extends JFrame implements ActionListener
    JButton btn ;
    JTextField textField;

    MyFrame(){
        this.setTitle("JTextField");
        this.setLayout(new FlowLayout());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        btn  = new JButton("Submit");
        btn.addActionListener(this);

        textField = new JTextField();
        textField.setPreferredSize(new Dimension(250,40));
        textField.setFont(new Font("Consolas",Font.BOLD,35));
        textField.setForeground(Color.GREEN);
        textField.setCaretColor(Color.red);

        this.add(textField);
        this.add(btn);
        this.pack();
        this.setVisible(true);
    }
```



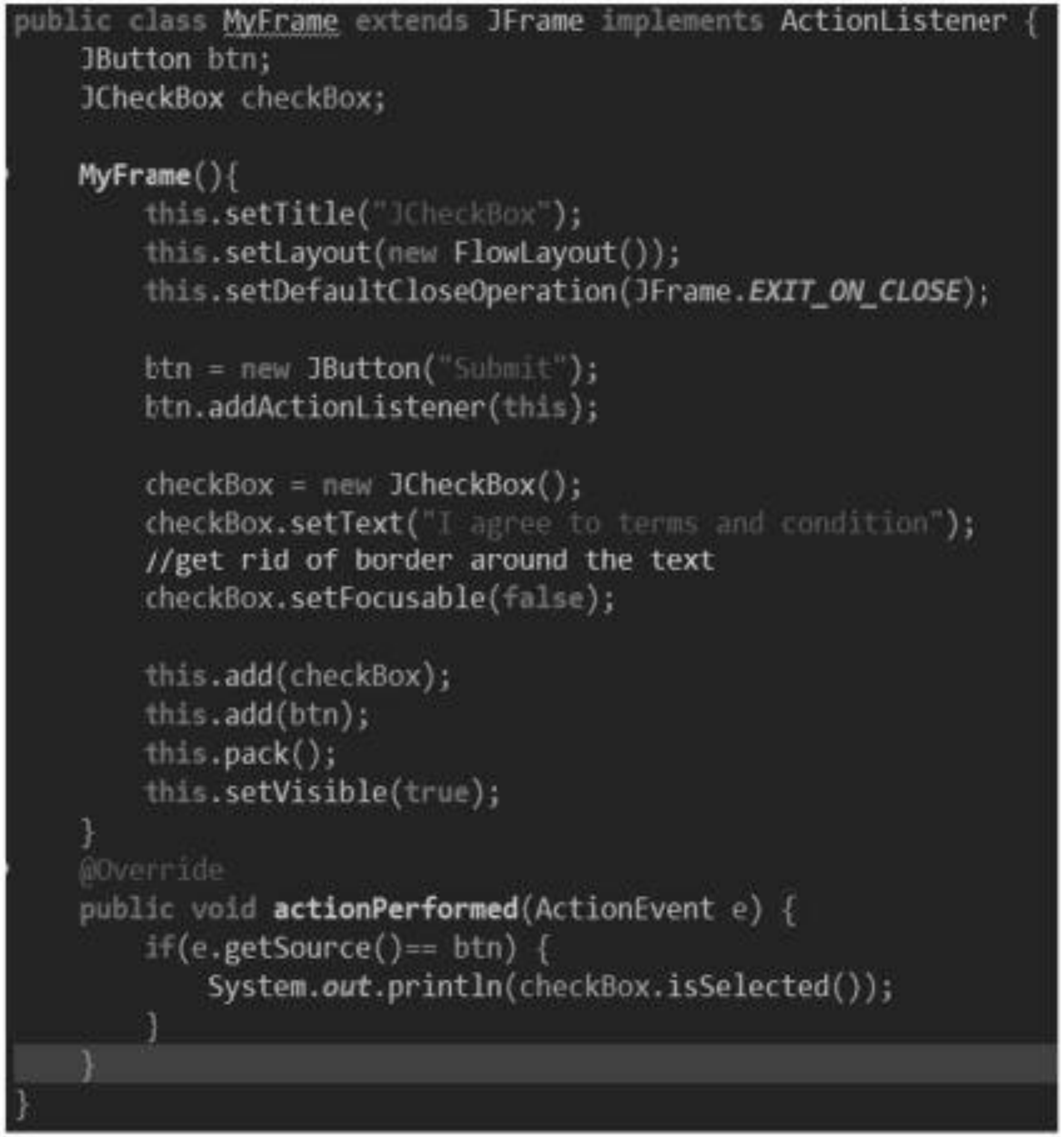
```
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btn) {
            System.out.println(textField.getText());
            btn.setEnabled(false);
            textField.setEditable(false);
        }

    }

}
```

We are disabling button and text field after button click



JCheckBox

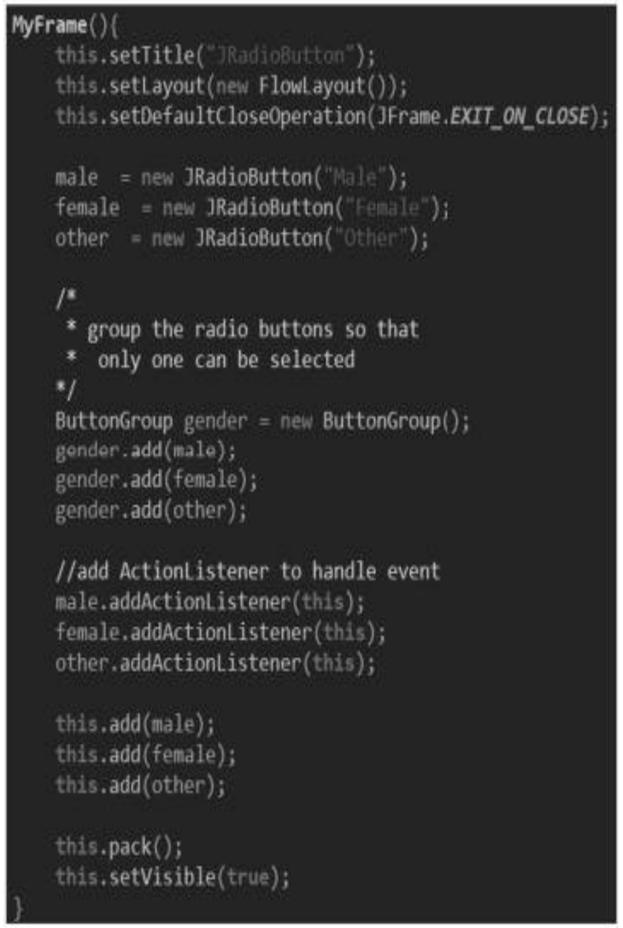
- ❖ The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false).
- ❖ Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on



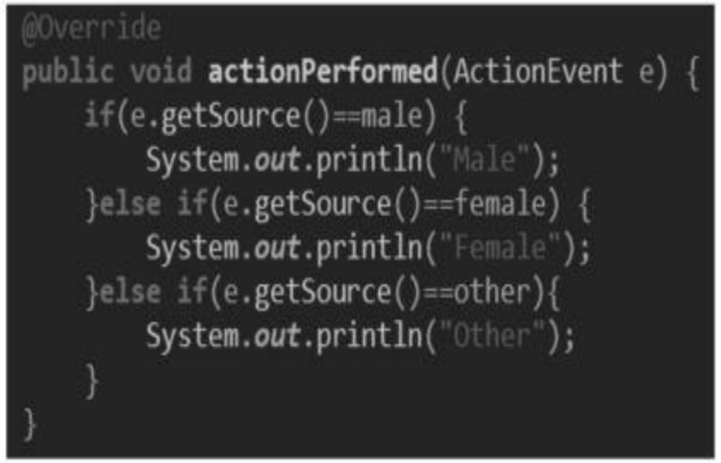
```
public class MyFrame extends JFrame implements ActionListener {
    JButton btn;
    JCheckBox checkBox;

    MyFrame(){
        this.setTitle("JCheckBox");
        this.setLayout(new FlowLayout());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        btn = new JButton("Submit");
        btn.addActionListener(this);

        checkBox = new JCheckBox();
        checkBox.setText("I agree to terms and condition");
        //get rid of border around the text
        checkBox.setFocusable(false);

        this.add(checkBox);
        this.add(btn);
        this.pack();
        this.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()== btn) {
            System.out.println(checkBox.isSelected());
        }
    }
}
```

JRadioButton

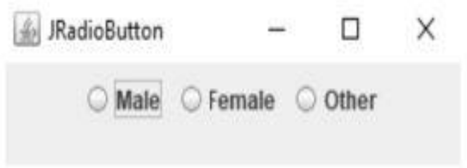
- ❖ The JRadioButton class is used to create a radio button.
- ❖ It is used to choose one option from multiple options.
- ❖ **It should be added in ButtonGroup to select one radio button only**



```
MyFrame(){
    this.setTitle("JRadioButton");
    this.setLayout(new FlowLayout());
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    male   = new JRadioButton("Male");
    female  = new JRadioButton("Female");
    other   = new JRadioButton("Other");

    /*
     * group the radio buttons so that
     *  only one can be selected
     */
    ButtonGroup gender = new ButtonGroup();
    gender.add(male);
    gender.add(female);
    gender.add(other);

    //add ActionListener to handle event
    male.addActionListener(this);
    female.addActionListener(this);
    other.addActionListener(this);

    this.add(male);
    this.add(female);
    this.add(other);

    this.pack();
    this.setVisible(true);
}
```



```
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==male) {
        System.out.println("Male");
    }else if(e.getSource()==female) {
        System.out.println("Female");
    }else if(e.getSource()==other){
        System.out.println("Other");
    }
}
```

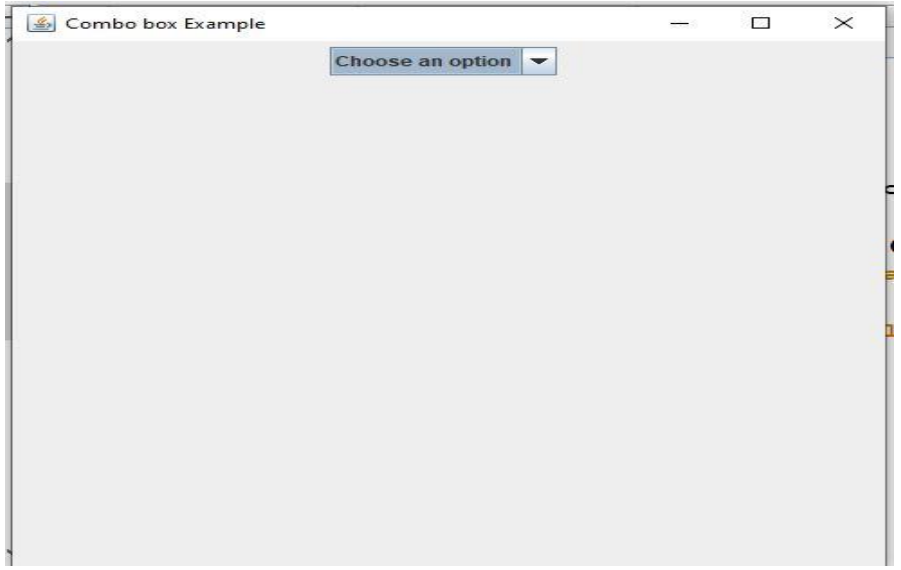


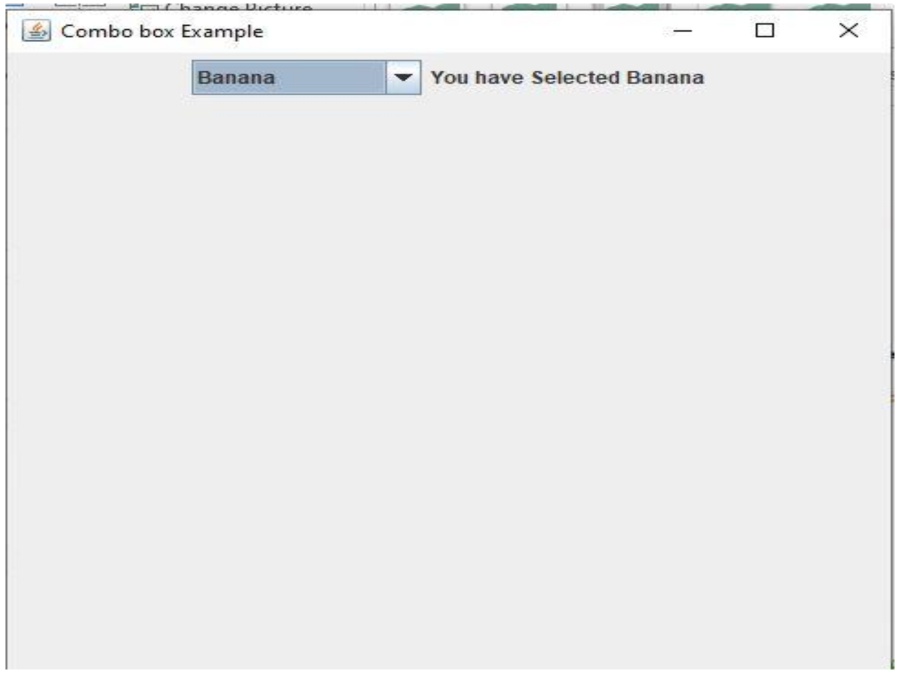
JComboBox

- ❖ JComboBox shows a popup menu that shows a list and the user can select a option from that specified list .
- ❖ JComboBox can be editable or read- only depending on the choice of the programmer .
- ❖ Constructor of the JComboBox are:
  - JComboBox() : creates a new empty JComboBox .
  - JComboBox(ComboJFrBoxModel M) : creates a new JComboBox with items from specified ComboBoxModel
  - JComboBox(E [ ] i) : creates a new JComboBox with items from specified array.
  - JComboBox(Vector items) : creates a new JComboBox with items from the specified vector

JComboBox-Example

```java
public class ComboBoxExample extends JFrame implements ActionListener{
    JComboBox combo;
    JLabel lbl = new JLabel();
    public ComboBoxExample(){
        this.setTitle("Combo box Example");
        this.setSize(500, 500);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLayout(new FlowLayout());
        String[] fruitList = {"Choose an option","Apple","Banana","Mango"};
        combo = new JComboBox(fruitList);
        combo.addActionListener(this);
        this.add(combo);
        this.add(lbl);
        this.setVisible(true);
    }
   @Override
   public void actionPerformed(ActionEvent e) {
       if(e.getSource()==combo){
           if(combo.getSelectedItem().toString().equalsIgnoreCase("Choose an option")){
               lbl.setText("Choose a fruit");
           }else{
           lbl.setText("You have Selected "+ combo.getSelectedItem());
          }
       }
   }
}
```



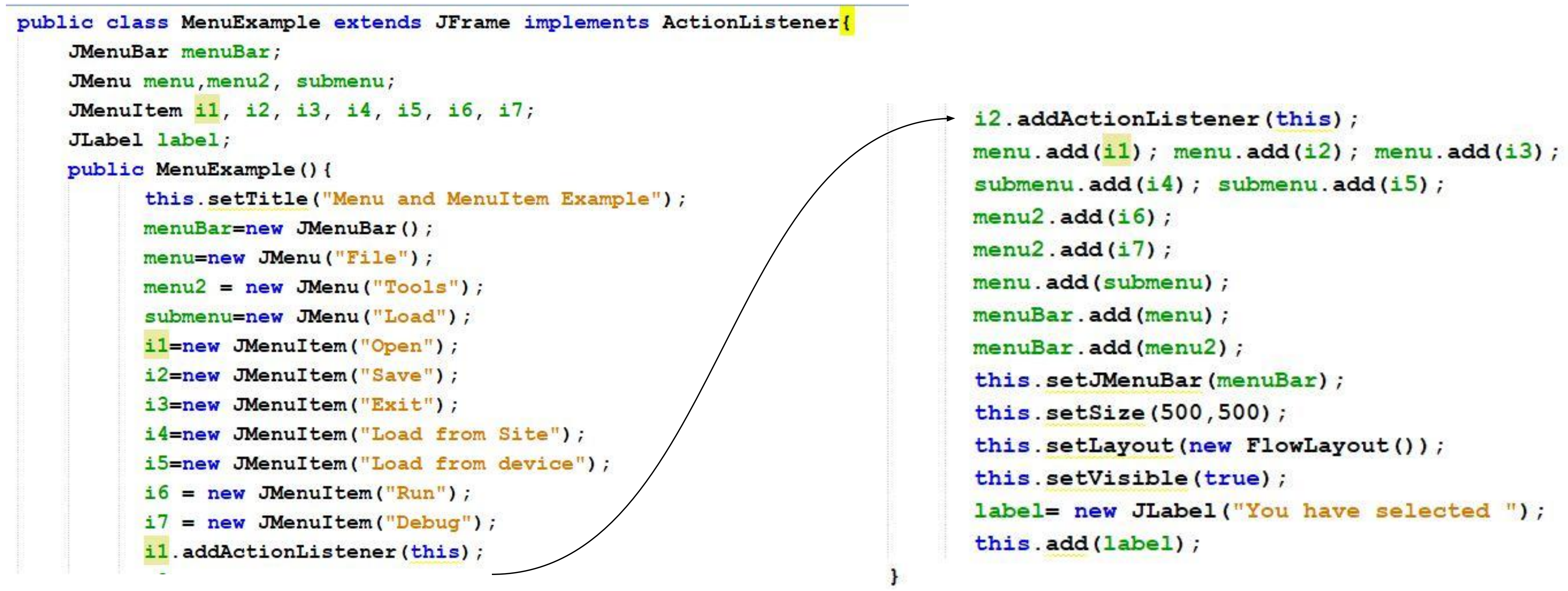


You have Selected Banana

Java JMenuBar, JMenu and JMenuItem

- ❖ The JMenuBar class is used to display menubar on the window or frame. It may have several menus.
- ❖ The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.
- ❖ The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

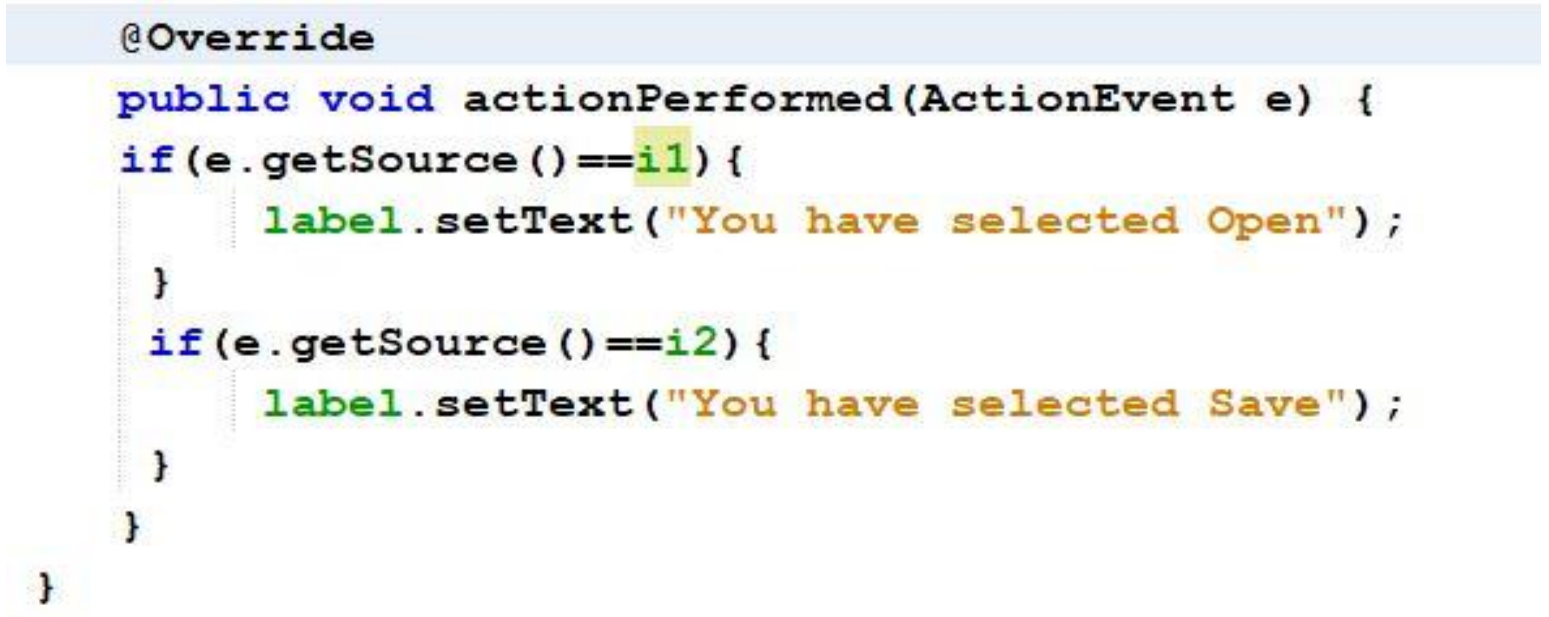
Java JMenuBar, JMenu and JMenuItem-Example



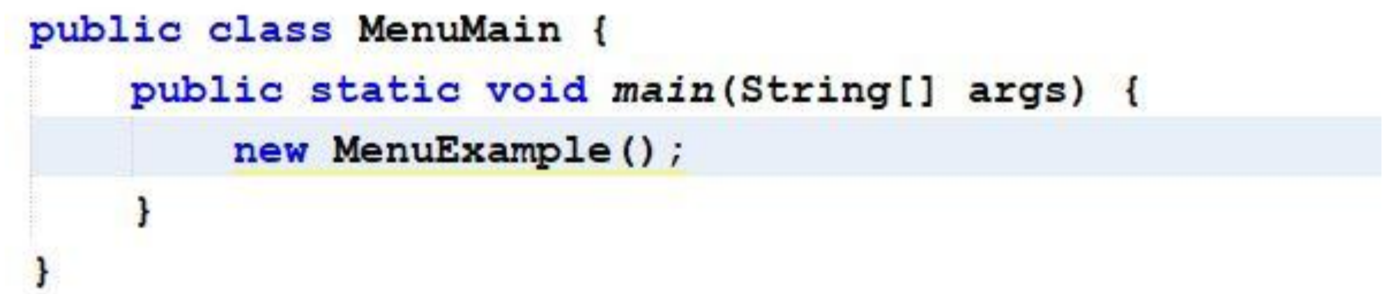
```
public class MenuExample extends JFrame implements ActionListener{
    JMenuBar menuBar;
    JMenu menu,menu2, submenu;
    JMenuItem i1, i2, i3, i4, i5, i6, i7;
    JLabel label;
    public MenuExample(){
          this.setTitle("Menu and MenuItem Example");
          menuBar=new JMenuBar();
          menu=new JMenu("File");
          menu2 = new JMenu("Tools");
          submenu=new JMenu("Load");
          i1=new JMenuItem("Open");
          i2=new JMenuItem("Save");
          i3=new JMenuItem("Exit");
          i4=new JMenuItem("Load from Site");
          i5=new JMenuItem("Load from device");
          i6 = new JMenuItem("Run");
          i7 = new JMenuItem("Debug");
          i1.addActionListener(this);
```

```
          i2.addActionListener(this);
          menu.add(i1); menu.add(i2); menu.add(i3);
          submenu.add(i4); submenu.add(i5);
          menu2.add(i6);
          menu2.add(i7);
          menu.add(submenu);
          menuBar.add(menu);
          menuBar.add(menu2);
          this.setJMenuBar(menuBar);
          this.setSize(500,500);
          this.setLayout(new FlowLayout());
          this.setVisible(true);
          label= new JLabel("You have selected ");
          this.add(label);
    }
```

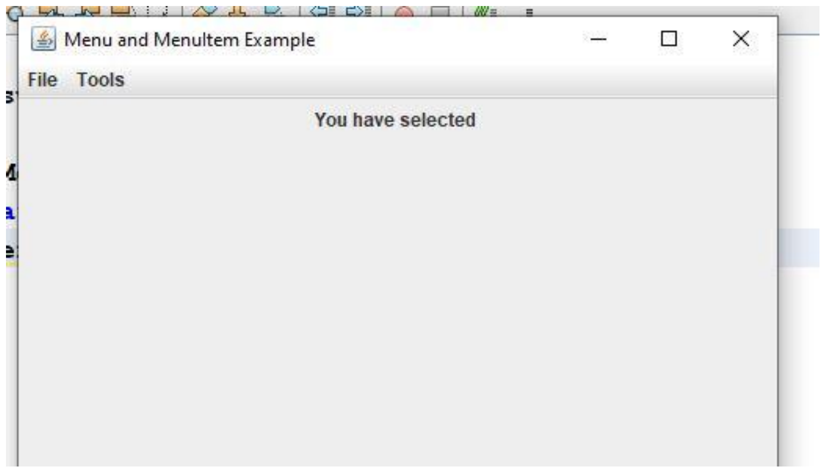
Java JMenuBar, JMenu and JMenuItem-Example



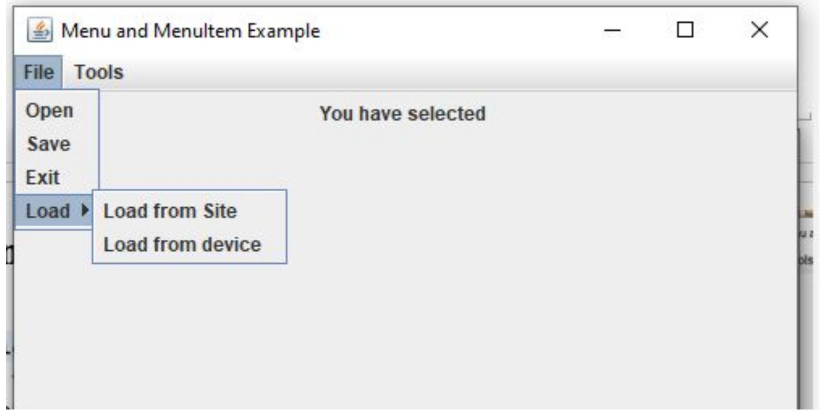
```
    @Override
    public void actionPerformed(ActionEvent e)  {
    if(e.getSource()==i1){
            label.setText("You have selected Open");
     }
     if(e.getSource()==i2){
            label.setText("You have selected Save");
     }
    }
}
```



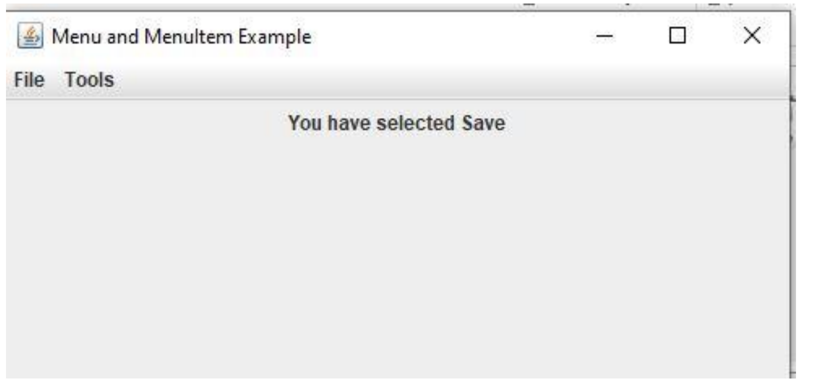
```
 public class MenuMain {
     public static void main(String[] args)  {
         new MenuExample();
     }
 }
```



```
You have selected
```



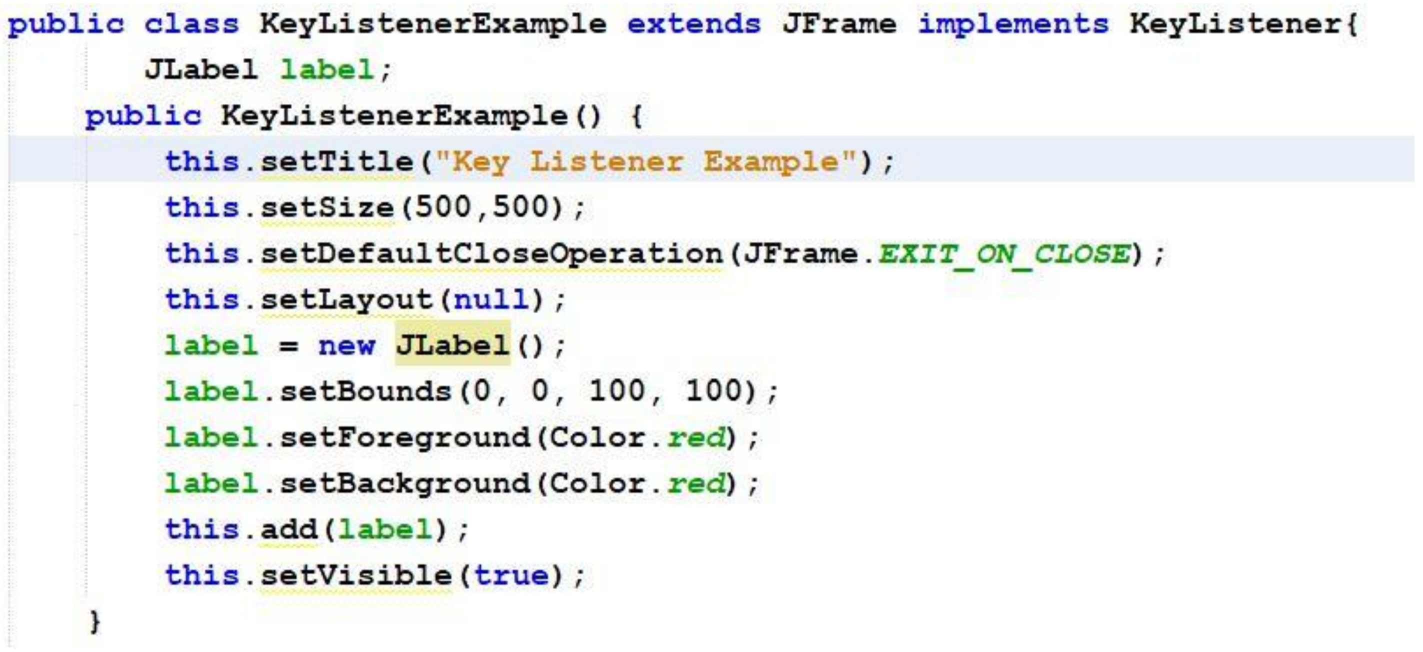
```
You have selected
```



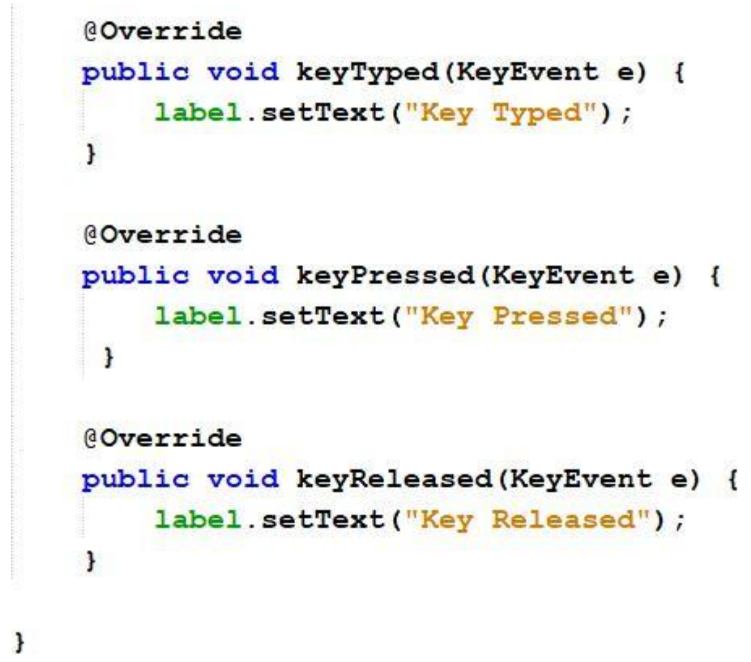
```
You have selected Save
```

Key Event Handling

- The **Java KeyListener is notified whenever you change the state of key.**
- It is notified against KeyEvent.
- The KeyListener interface is found in java.awt.event package, and it has three methods.



```
public class KeyListenerExample extends JFrame implements KeyListener{
        JLabel label;
    public KeyListenerExample() {
        this.setTitle("Key Listener Example");
        this.setSize(500,500);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLayout(null);
        label = new JLabel();
        label.setBounds(0, 0, 100, 100);
        label.setForeground(Color.red);
        label.setBackground(Color.red);
        this.add(label);
        this.setVisible(true);
    }
```



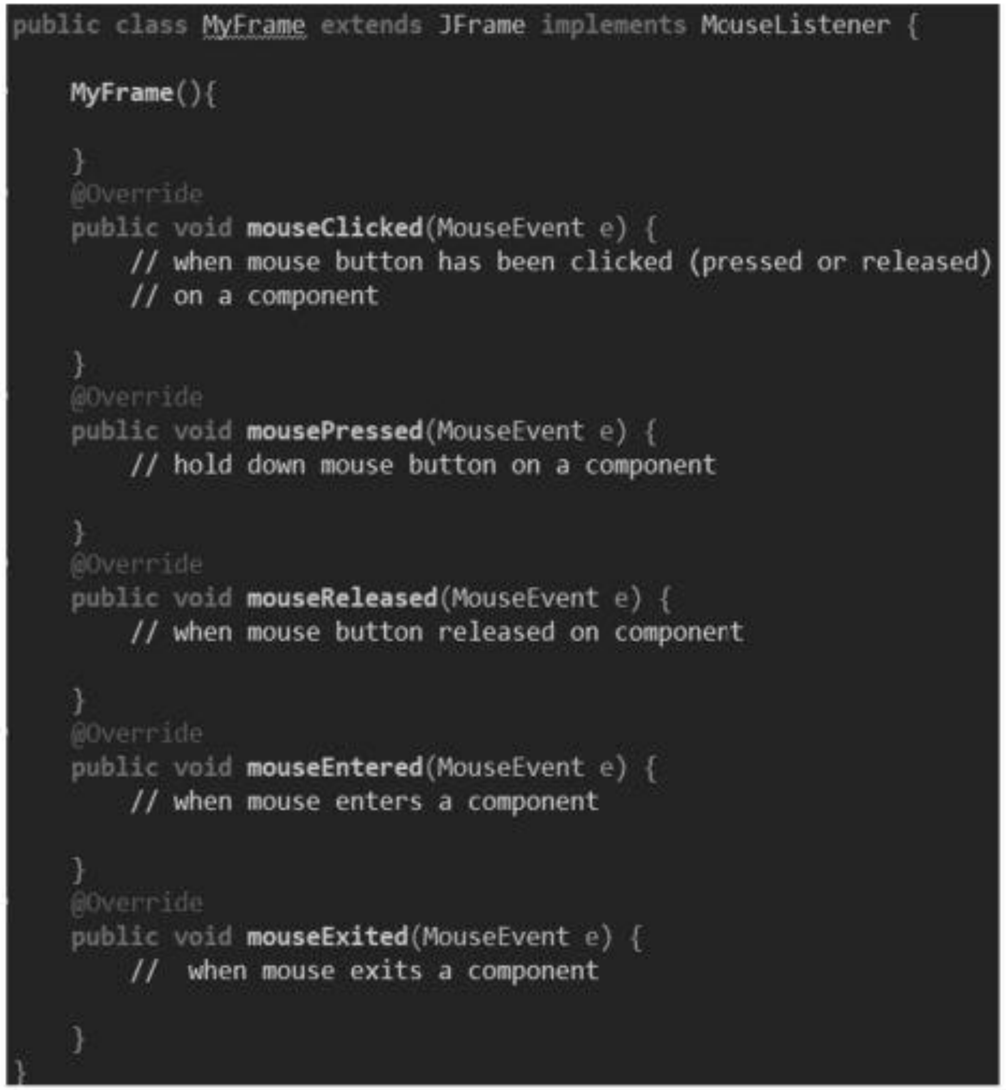
```
    @Override
    public void keyTyped(KeyEvent e) {
        label.setText("Key Typed");
    }

    @Override
    public void keyPressed(KeyEvent e) {
        label.setText("Key Pressed");
     }

    @Override
    public void keyReleased(KeyEvent e) {
        label.setText("Key Released");
    }

}
```

Mouse Event Handling

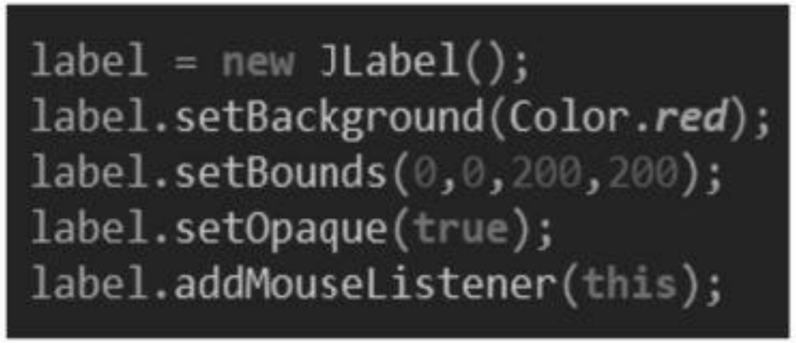
- ❖ The class which processes the MouseEvent should implement MouseListener Interface.
- ❖ The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent.
- ❖ The MouseListener interface is found in java.awt.event package. It has five methods.
- ❖ Methods:

a. public abstract void mouseClicked(MouseEvent e);
b. public abstract void mouseEntered(MouseEvent e);
c. public abstract void mouseExited(MouseEvent e);
d. public abstract void mousePressed(MouseEvent e);
e. public abstract void mouseReleased(MouseEvent e);

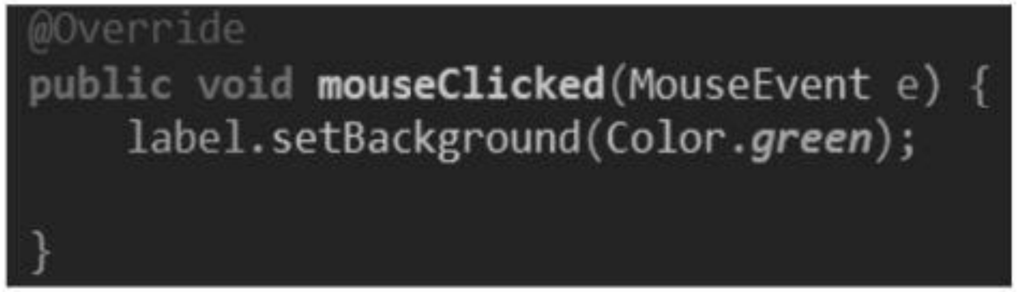
Mouse Event Handling



```
public class MyFrame extends JFrame implements MouseListener {

    MyFrame(){

    }
    @Override
    public void mouseClicked(MouseEvent e) {
        // when mouse button has been clicked (pressed or released)
        // on a component

    }
    @Override
    public void mousePressed(MouseEvent e) {
        // hold down mouse button on a component

    }
    @Override
    public void mouseReleased(MouseEvent e) {
        // when mouse button released on component

    }
    @Override
    public void mouseEntered(MouseEvent e) {
        // when mouse enters a component

    }
    @Override
    public void mouseExited(MouseEvent e) {
        //  when mouse exits a component

    }
}
```



```
label = new JLabel();
label.setBackground(Color.red);
label.setBounds(0,0,200,200);
label.setOpaque(true);
label.addMouseListener(this);
```

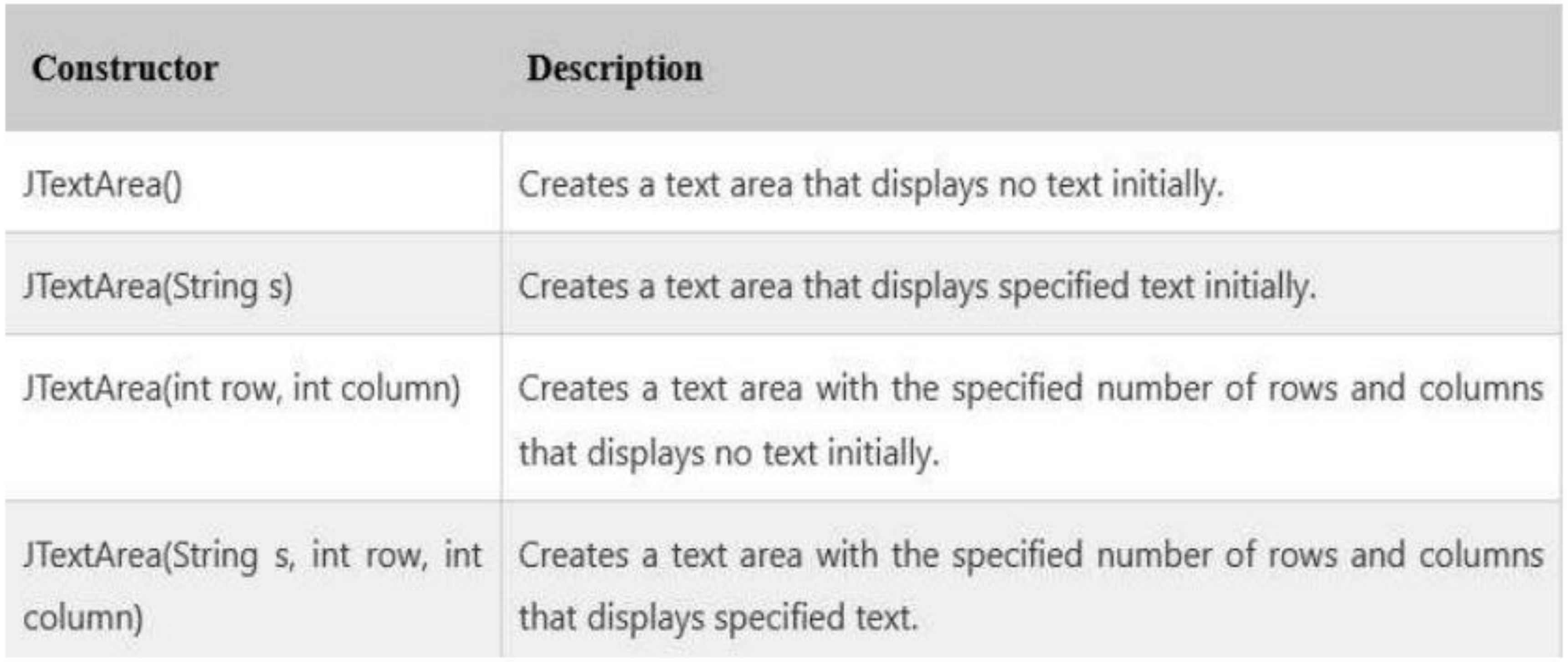


```
@Override
public void mouseClicked(MouseEvent e) {
    label.setBackground(Color.green);

}
```

JTextArea

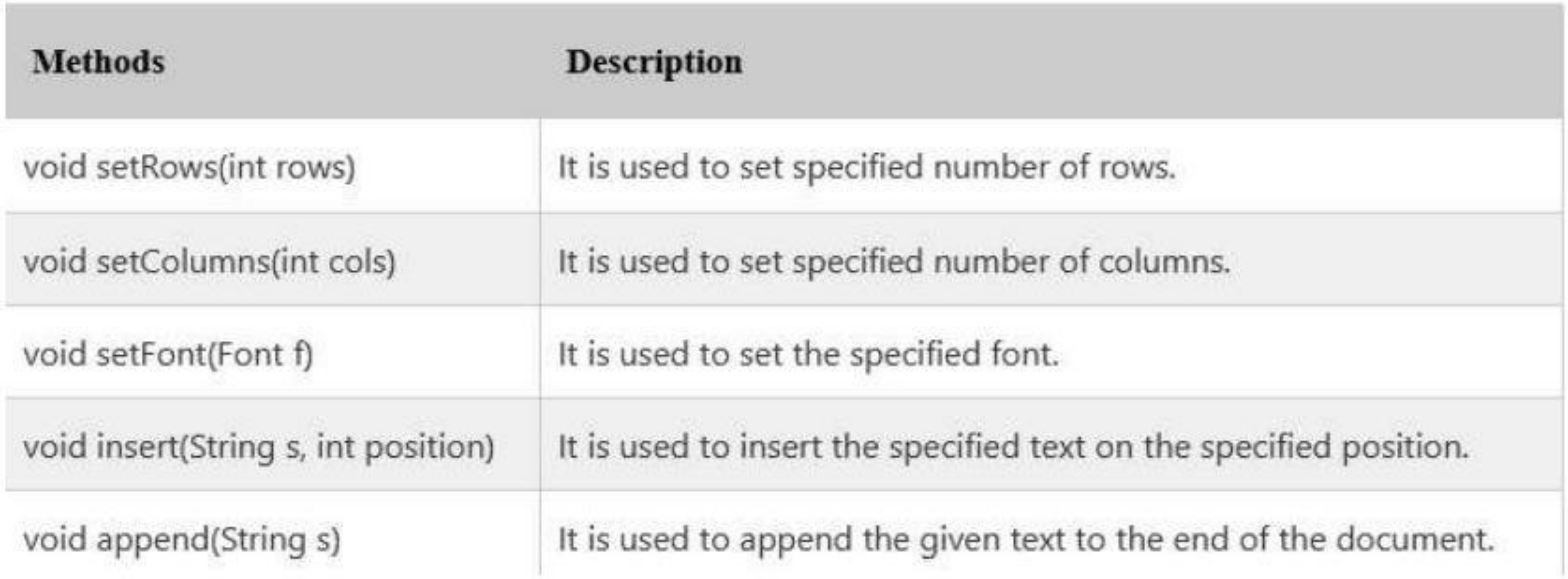
❖ The object of a JTextArea class is a multi line region that displays text.

❖ **It allows the editing of multiple line text**.



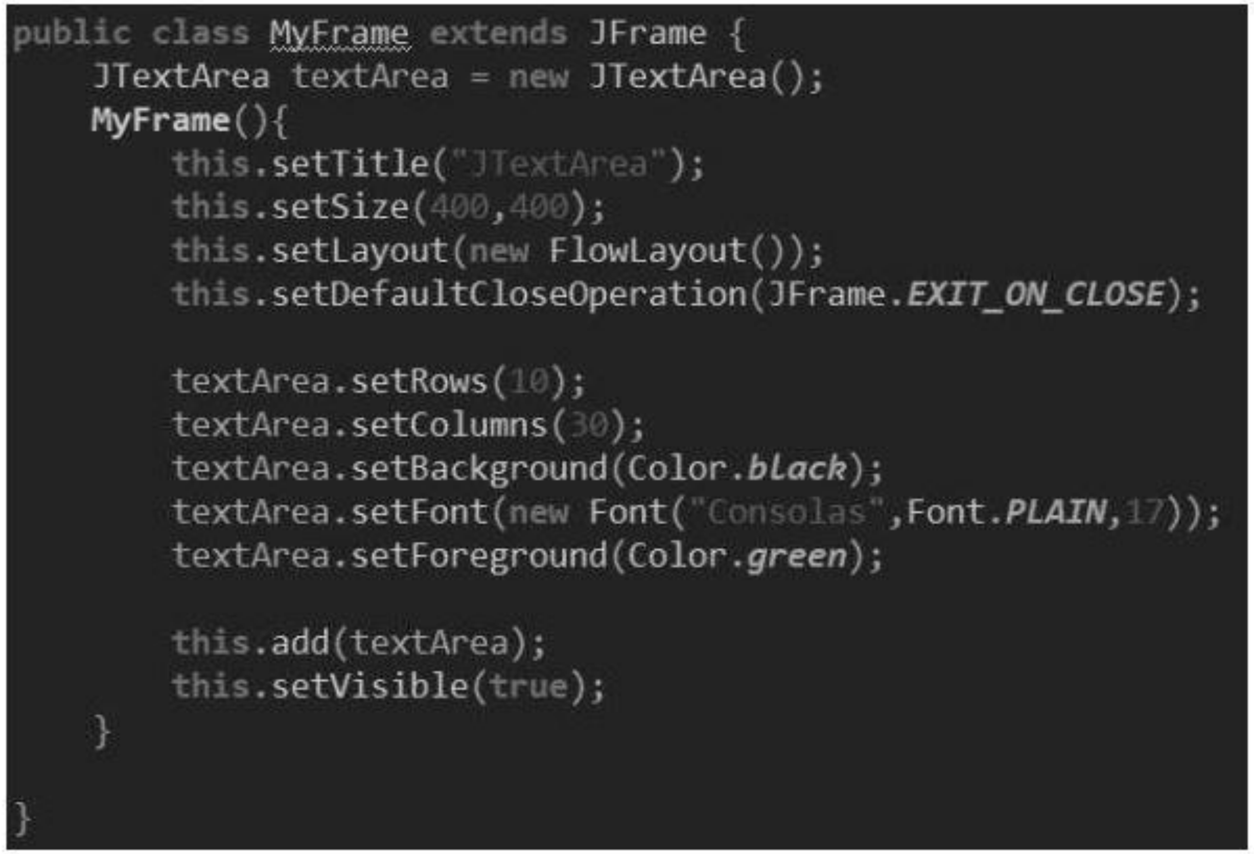
Constructor	Description
JTextArea()	Creates a text area that displays no text initially.
JTextArea(String s)	Creates a text area that displays specified text initially.
JTextArea(int row, int column)	Creates a text area with the specified number of rows and columns that displays no text initially.
JTextArea(String s, int row, int column)	Creates a text area with the specified number of rows and columns that displays specified text.

JTextArea

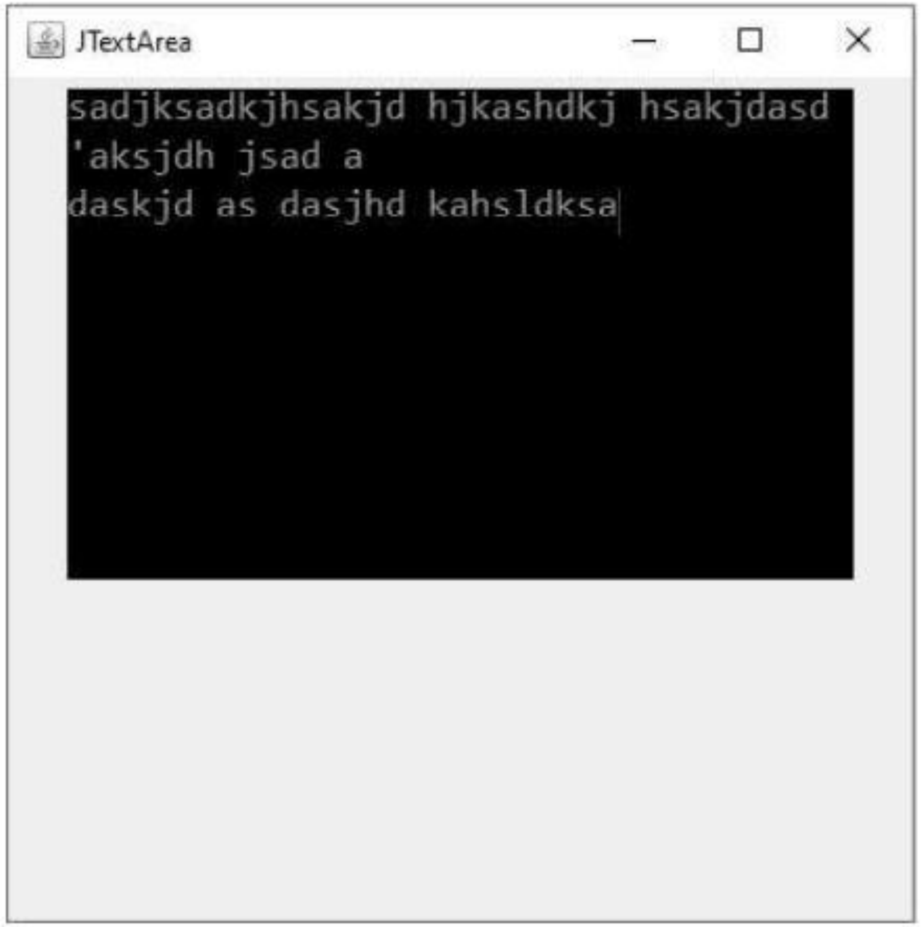


Methods	Description
void setRows(int rows)	It is used to set specified number of rows.
void setColumns(int cols)	It is used to set specified number of columns.
void setFont(Font f)	It is used to set the specified font.
void insert(String s, int position)	It is used to insert the specified text on the specified position.
void append(String s)	It is used to append the given text to the end of the document.

JTextArea-Example



```
public class MyFrame extends JFrame {
    JTextArea textArea = new JTextArea();
    MyFrame(){
        this.setTitle("JTextArea");
        this.setSize(400,400);
        this.setLayout(new FlowLayout());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        textArea.setRows(10);
        textArea.setColumns(30);
        textArea.setBackground(Color.black);
        textArea.setFont(new Font("Consolas",Font.PLAIN,17));
        textArea.setForeground(Color.green);

        this.add(textArea);
        this.setVisible(true);
    }

}
```

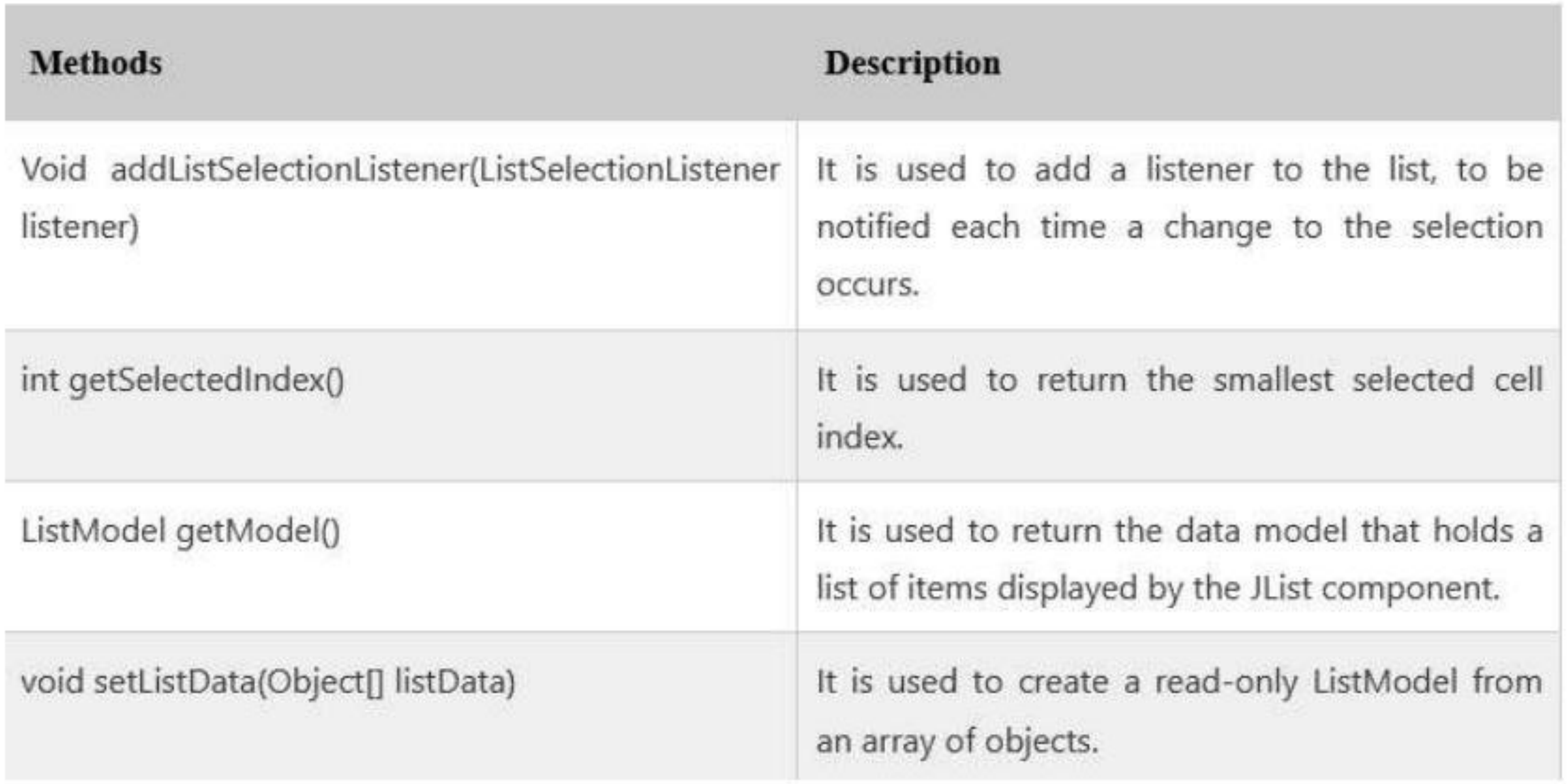


```
sadjksadkjhsakjd hjkashdkj hsakjdasd
'aksjdh jsad a
daskjd as dasjhd kahsldksa
```

JList

- ❖ JList is part of Java Swing package . JList is a component that displays a set of Objects and allows the user to select one or more items .
- ❖ JList inherits JComponent class. JList is a easy way to display an array of Vectors .
- ❖ Constructor for JList are :
    - **JList()**: creates an empty blank list
    - **JList(E [ ] lst)** : creates an new list with the elements of the array.
    - **JList(ListModel d)**: creates a new list with the specified List Model
    - **JList(Vector lst)** : creates a new list with the elements of the vector

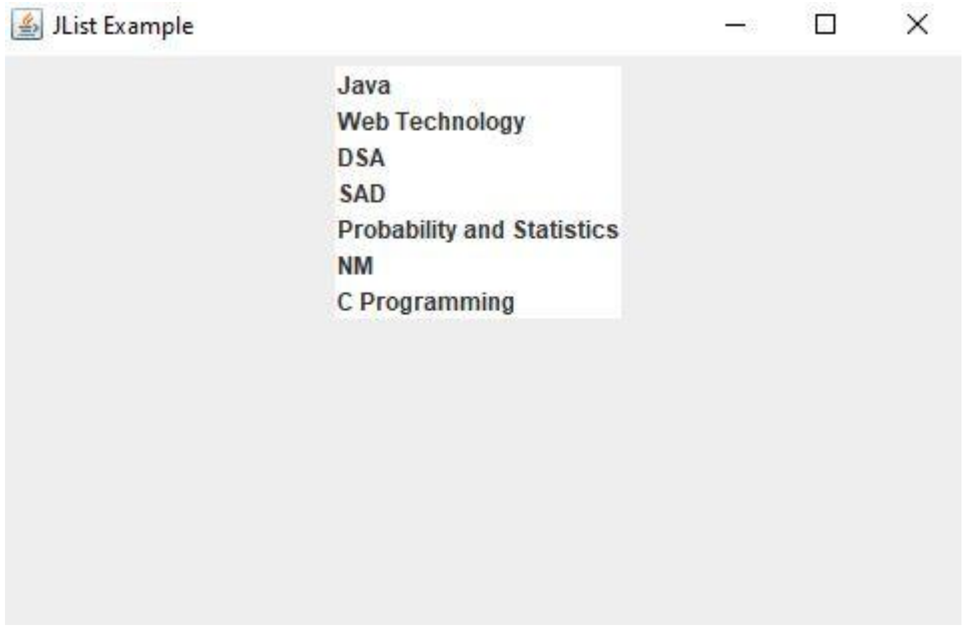
Jlist-Commonly Used Methods



Methods	Description
Void addListSelectionListener(ListSelectionListener listener)	It is used to add a listener to the list, to be notified each time a change to the selection occurs.
int getSelectedIndex()	It is used to return the smallest selected cell index.
ListModel getModel()	It is used to return the data model that holds a list of items displayed by the JList component.
void setListData(Object[] listData)	It is used to create a read-only ListModel from an array of objects.

JList-Example

```java
public class JListExample extends JFrame {
    JList<String> subjectList;
    public JListExample(){
        this.setTitle("JList Example");
        this.setSize(500, 500);
        this.setLayout(new FlowLayout());

        String[] mySubjects ={"Java", "Web Technology",
            "DSA","SAD","Probability and Statistics","NM","C Programming"};
        subjectList = new JList<>(mySubjects);
        this.add(subjectList);
        this.setVisible(true);
    }
}
```

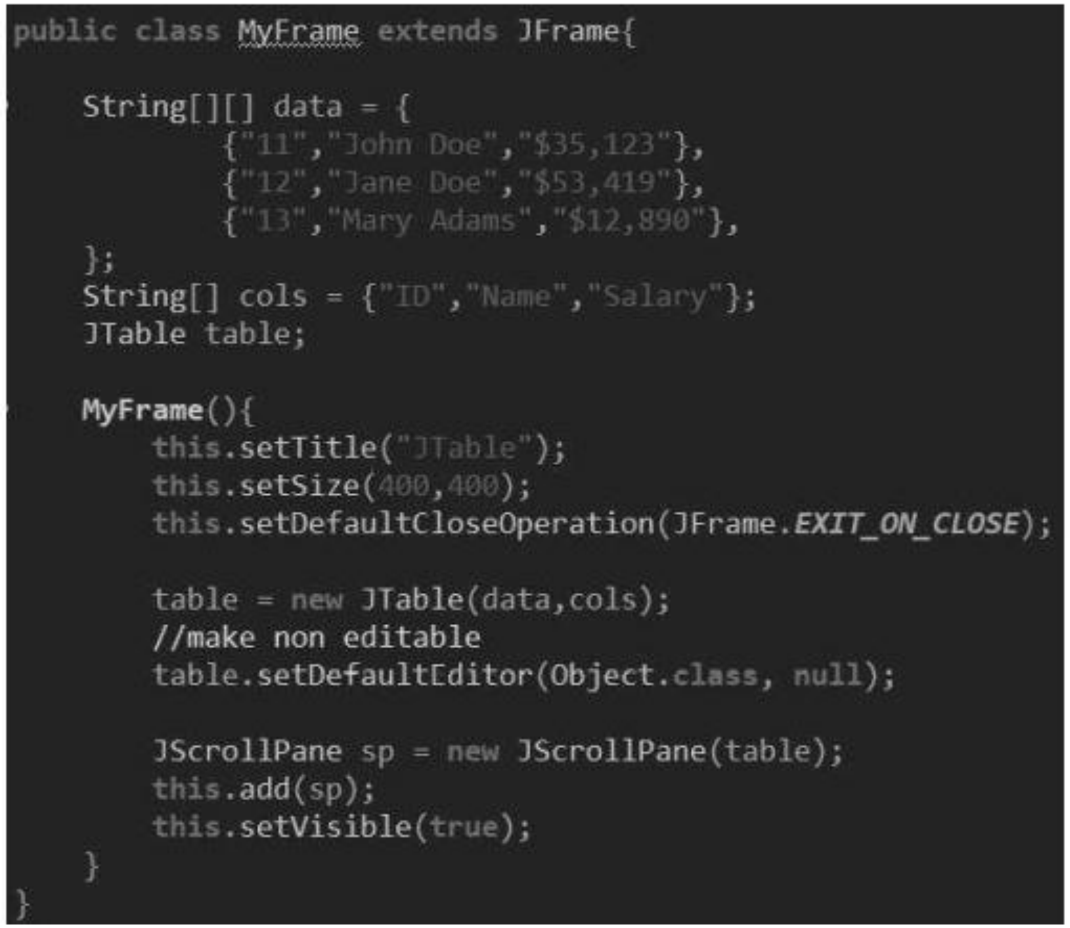


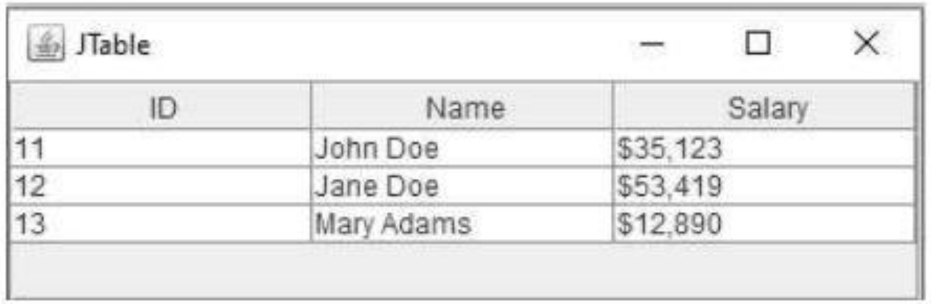
```
Java
Web Technology
DSA
SAD
Probability and Statistics
NM
C Programming
```

JTable

- The JTable class is a part of Java Swing Package and is generally used to display or edit two-dimensional data that is having both rows and columns.
- It is similar to a spreadsheet.
- Constructors in JTable:
  - **JTable():** A table is created with empty cells.
  - **JTable(int rows, int cols):** Creates a table of size rows * cols.
  - **JTable(Object[][] data, Object []Column):** A table is created with the specified name where []Column defines the column names.

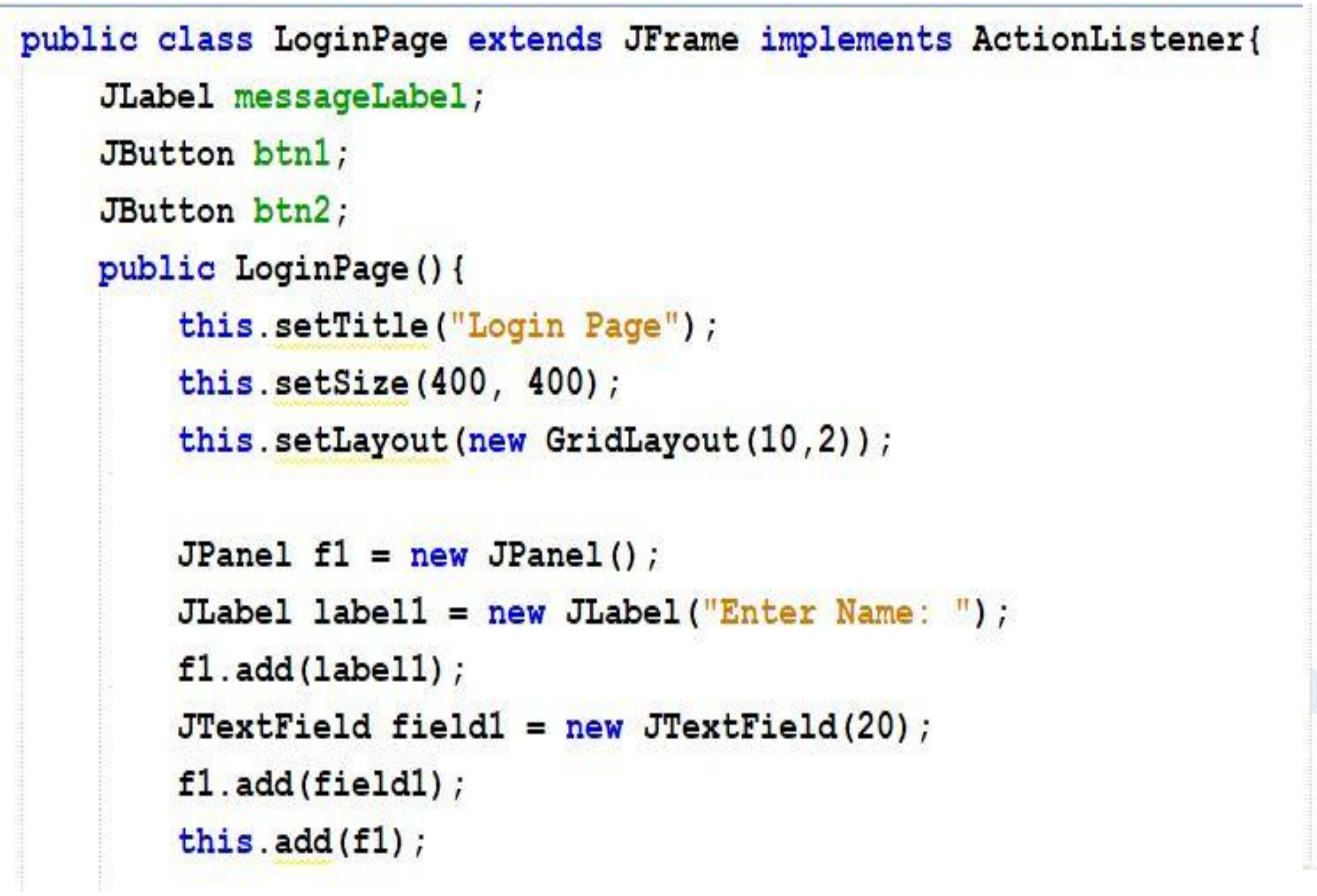
JTable-Example


```
public class MyFrame extends JFrame{

    String[][] data = {
            {"11","John Doe","$35,123"},
            {"12","Jane Doe","$53,419"},
            {"13","Mary Adams","$12,890"},
    };
    String[] cols = {"ID","Name","Salary"};
    JTable table;

    MyFrame(){
        this.setTitle("JTable");
        this.setSize(400,400);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        table = new JTable(data,cols);
        //make non editable
        table.setDefaultEditor(Object.class, null);

        JScrollPane sp = new JScrollPane(table);
        this.add(sp);
        this.setVisible(true);
    }
}
```

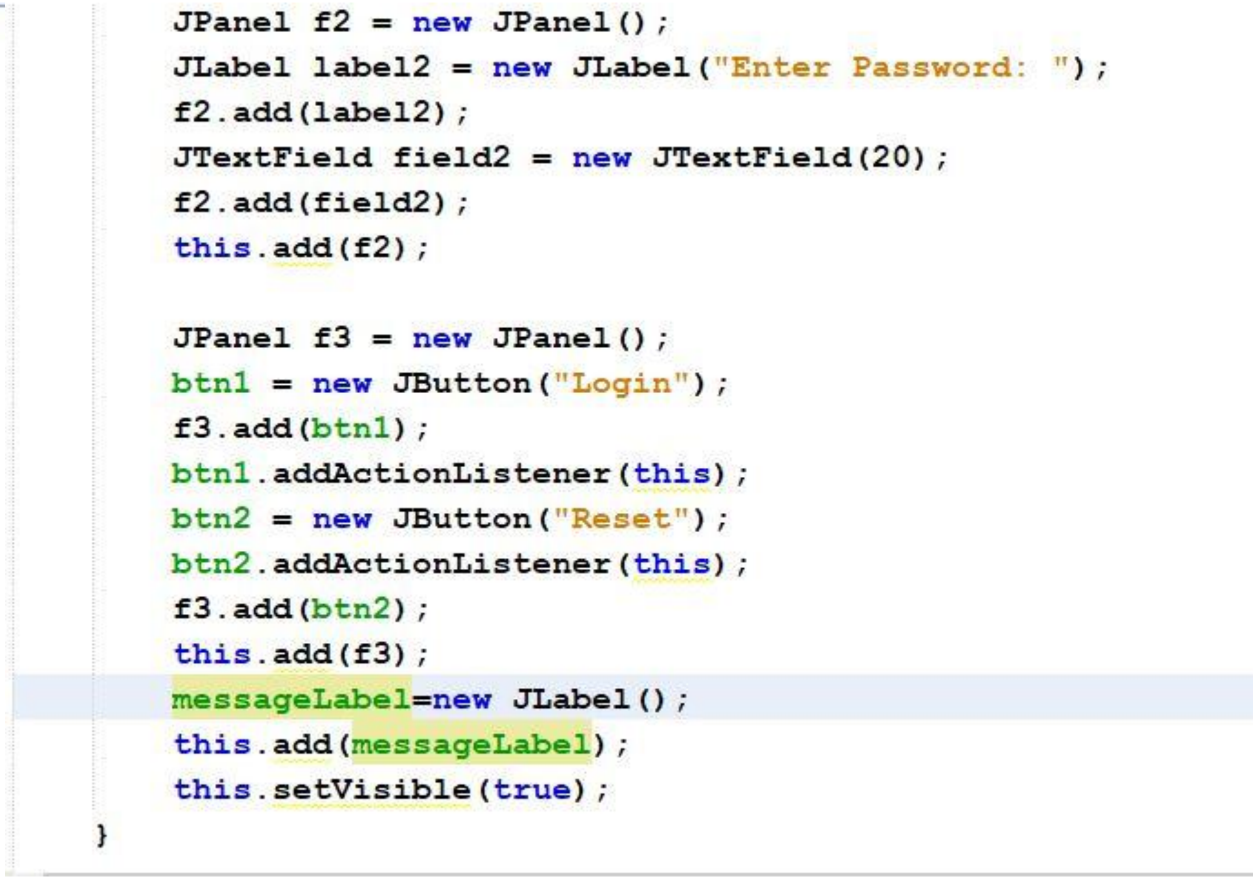


ID	Name	Salary
11	John Doe	$35,123
12	Jane Doe	$53,419
13	Mary Adams	$12,890

Login Form Example



```java
public class LoginPage extends JFrame implements ActionListener{
    JLabel messageLabel;
    JButton btn1;
    JButton btn2;
    public LoginPage(){
        this.setTitle("Login Page");
        this.setSize(400, 400);
        this.setLayout(new GridLayout(10,2));

        JPanel f1 = new JPanel();
        JLabel label1 = new JLabel("Enter Name: ");
        f1.add(label1);
        JTextField field1 = new JTextField(20);
        f1.add(field1);
        this.add(f1);
```

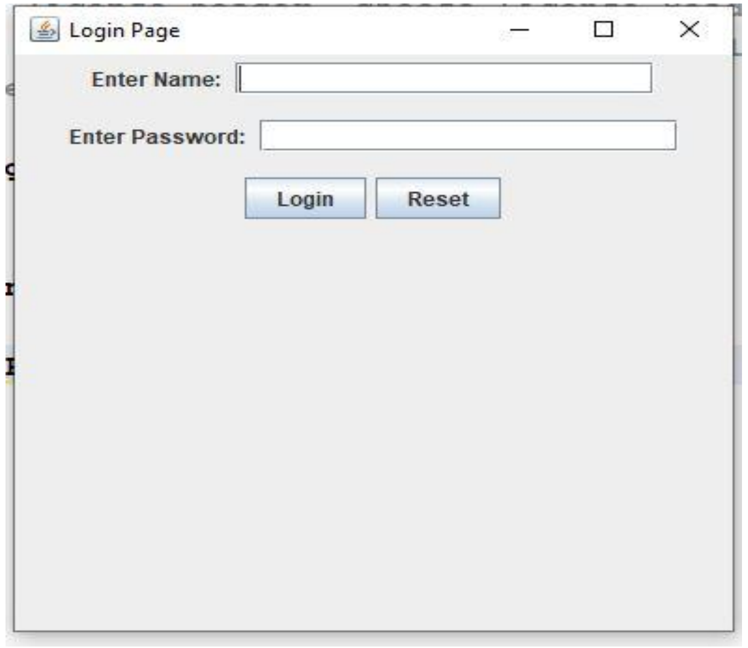


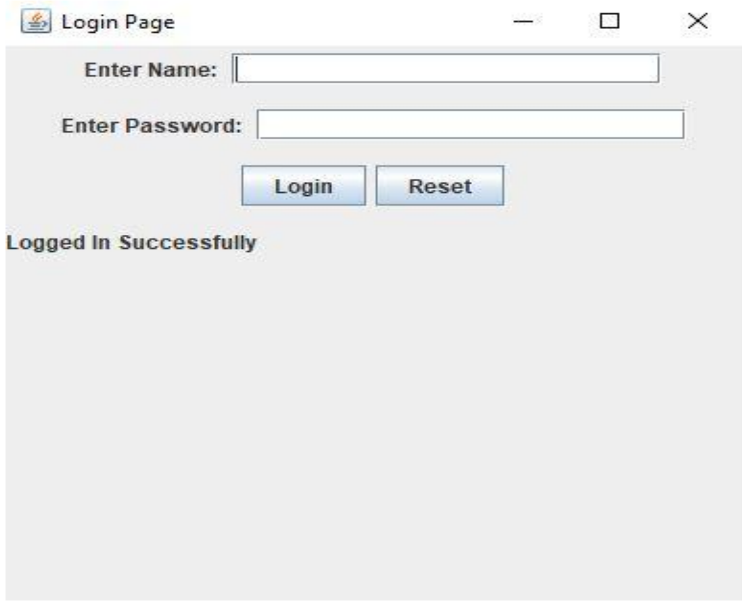
```java
        JPanel f2 = new JPanel();
        JLabel label2 = new JLabel("Enter Password: ");
        f2.add(label2);
        JTextField field2 = new JTextField(20);
        f2.add(field2);
        this.add(f2);

        JPanel f3 = new JPanel();
        btn1 = new JButton("Login");
        f3.add(btn1);
        btn1.addActionListener(this);
        btn2 = new JButton("Reset");
        btn2.addActionListener(this);
        f3.add(btn2);
        this.add(f3);
        messageLabel=new JLabel();
        this.add(messageLabel);
        this.setVisible(true);
    }
```

Login Form Example

```
    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btn1){
            messageLabel.setText("Logged In Successfully");
        }
        if(e.getSource()==btn2){
            messageLabel.setText(" ");
        }
    }

}
```

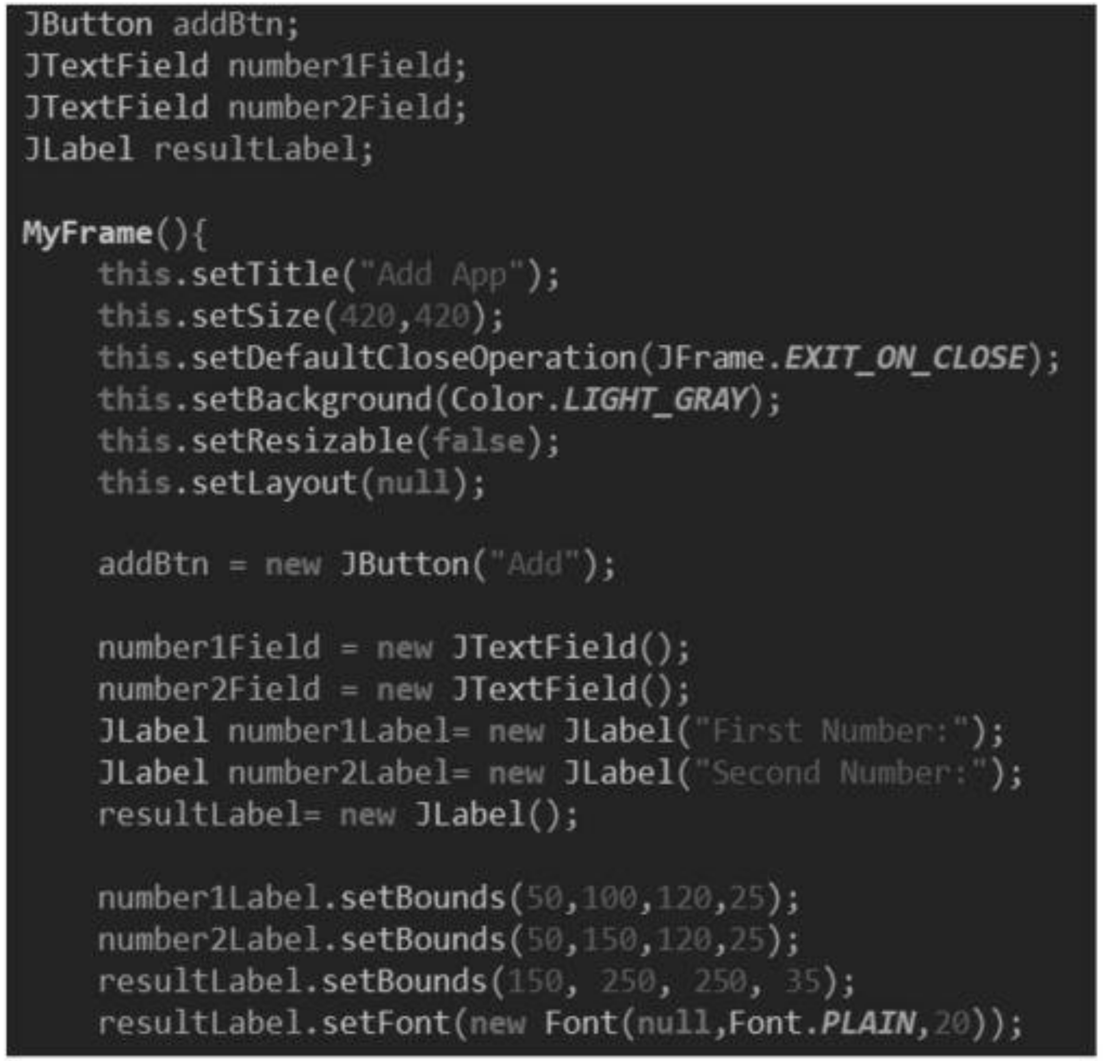
```
public class LoginPageMain {
    public static void main(String[] args) {
        new LoginPage();
    }
}
```



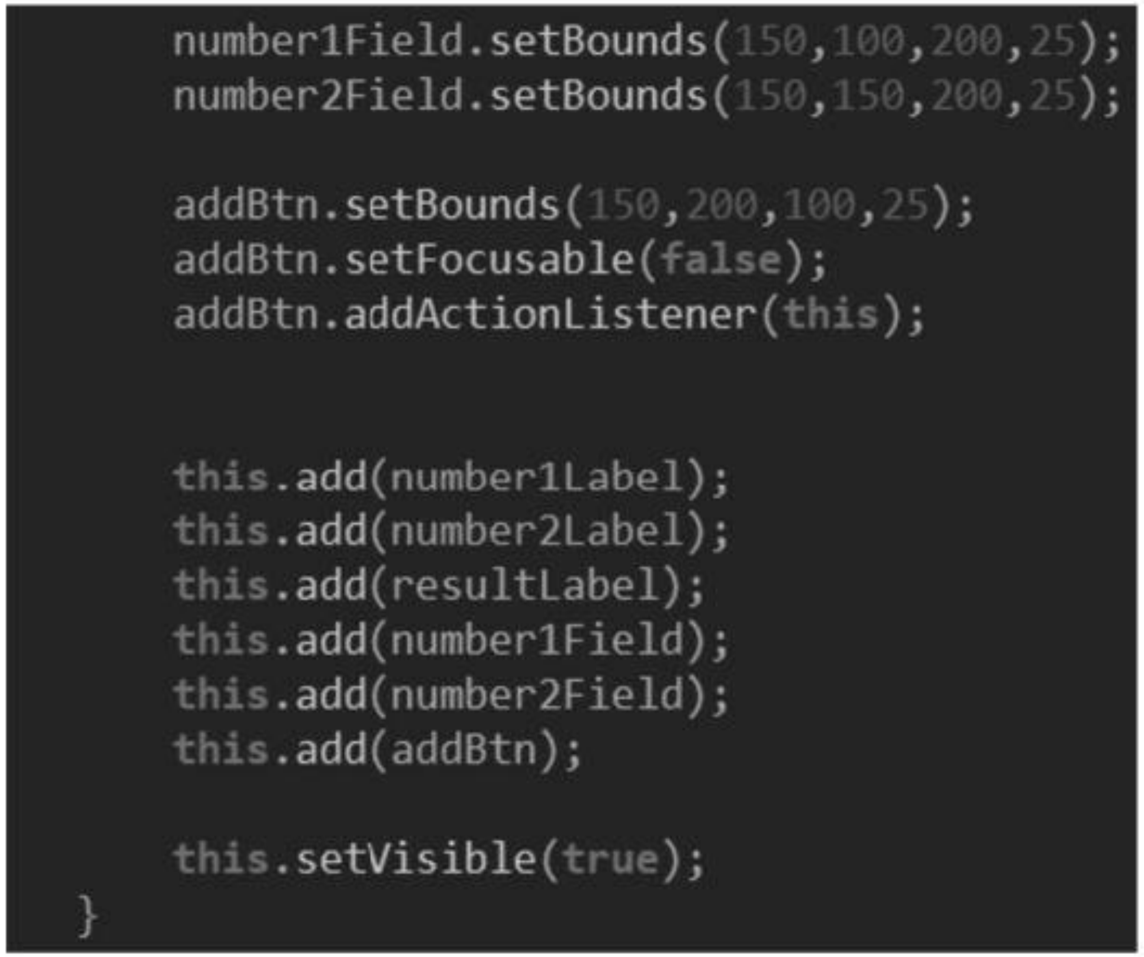


Logged In Successfully

Addition App

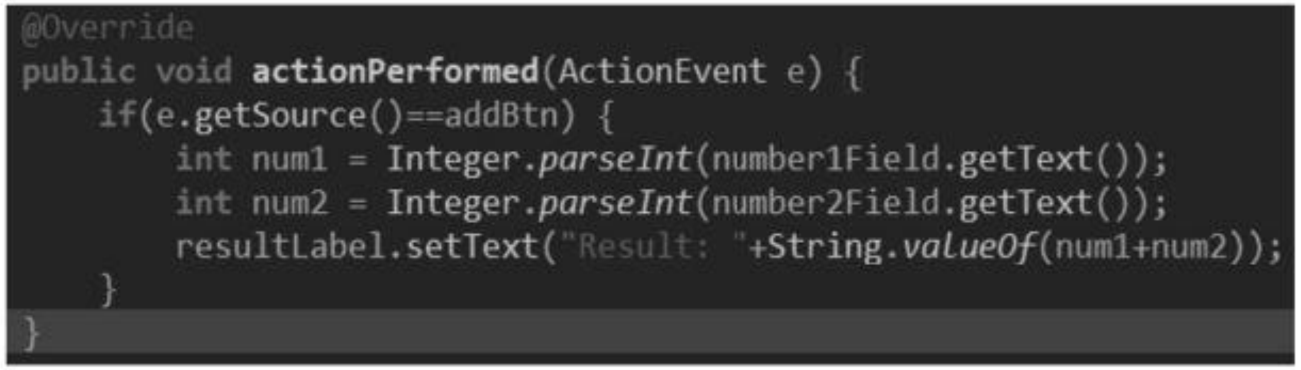


```
JButton addBtn;
JTextField number1Field;
JTextField number2Field;
JLabel resultLabel;

MyFrame(){
    this.setTitle("Add App");
    this.setSize(420,420);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setBackground(Color.LIGHT_GRAY);
    this.setResizable(false);
    this.setLayout(null);

    addBtn = new JButton("Add");

    number1Field = new JTextField();
    number2Field = new JTextField();
    JLabel number1Label= new JLabel("First Number:");
    JLabel number2Label= new JLabel("Second Number:");
    resultLabel= new JLabel();

    number1Label.setBounds(50,100,120,25);
    number2Label.setBounds(50,150,120,25);
    resultLabel.setBounds(150, 250, 250, 35);
    resultLabel.setFont(new Font(null,Font.PLAIN,20));
```

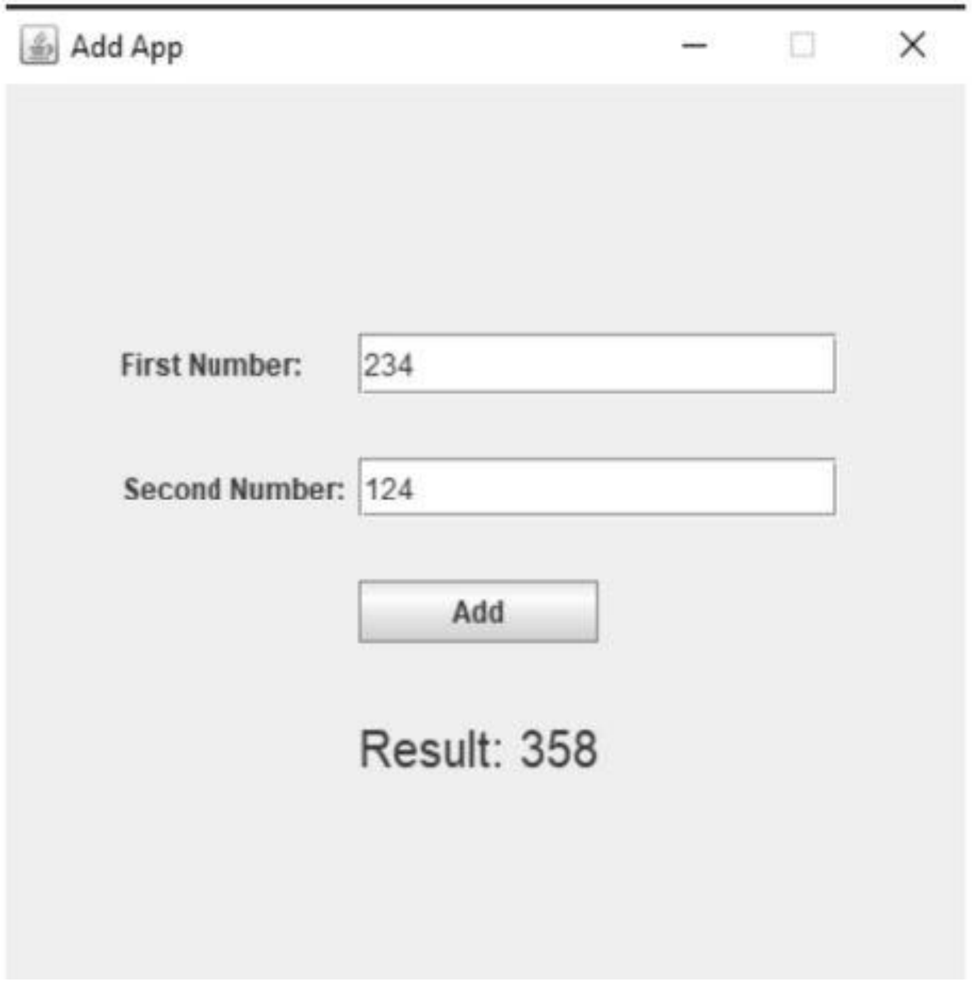


```
    number1Field.setBounds(150,100,200,25);
    number2Field.setBounds(150,150,200,25);

    addBtn.setBounds(150,200,100,25);
    addBtn.setFocusable(false);
    addBtn.addActionListener(this);


    this.add(number1Label);
    this.add(number2Label);
    this.add(resultLabel);
    this.add(number1Field);
    this.add(number2Field);
    this.add(addBtn);

    this.setVisible(true);
}
```

Addition App-Continue



```
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==addBtn) {
        int num1 = Integer.parseInt(number1Field.getText());
        int num2 = Integer.parseInt(number2Field.getText());
        resultLabel.setText("Result: "+String.valueOf(num1+num2));
    }
}
```



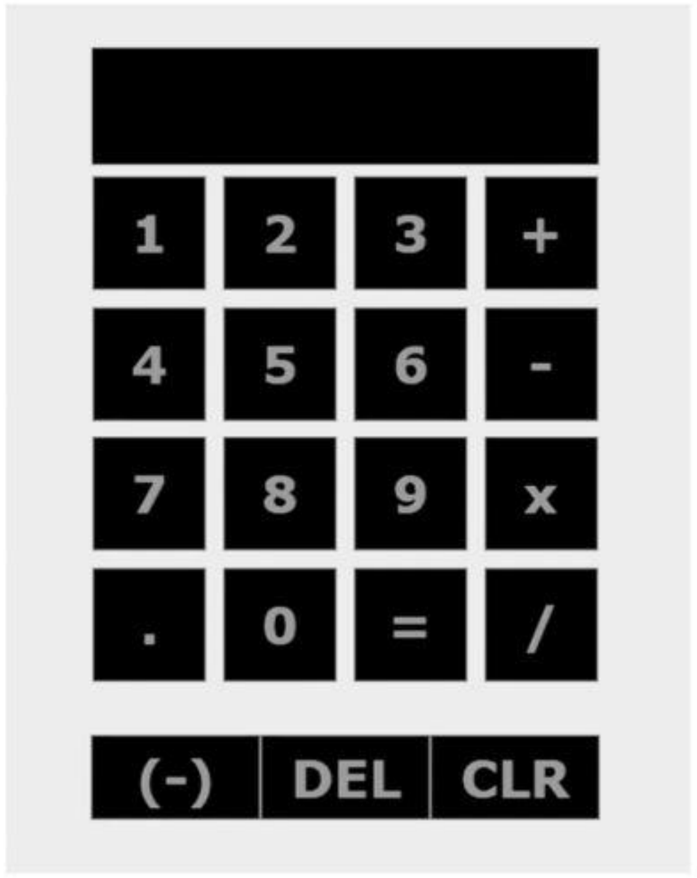
Add App

First Number: 234

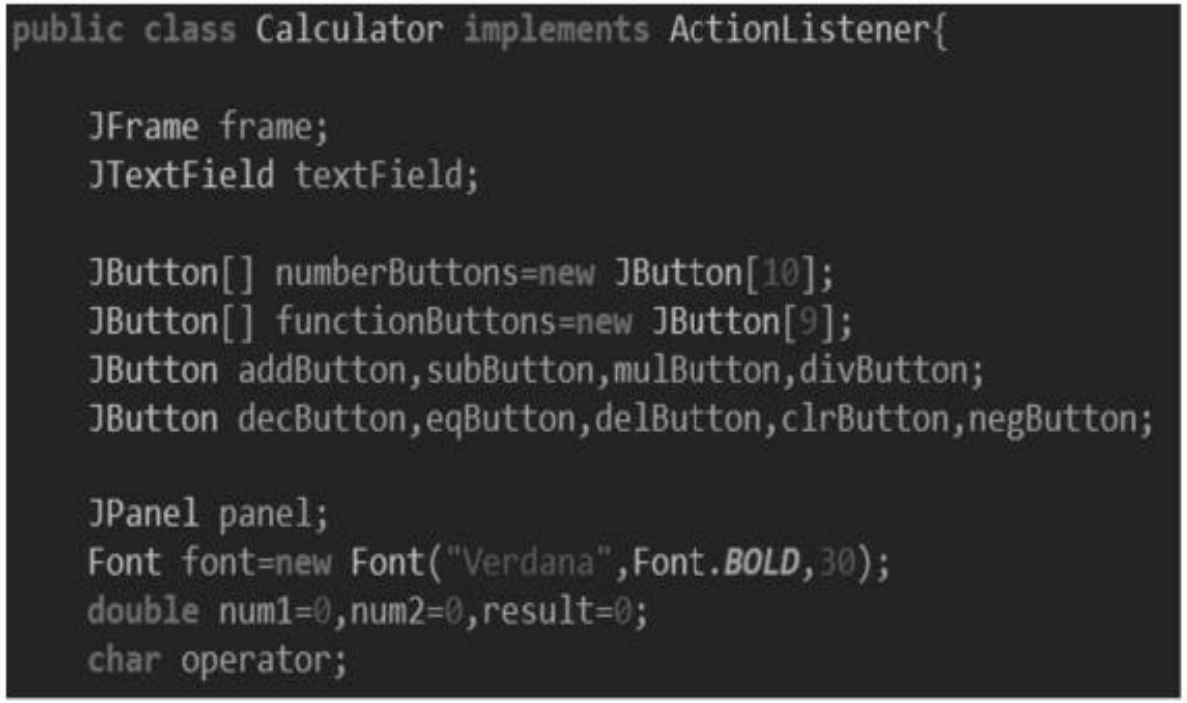
Second Number: 124

Result: 358

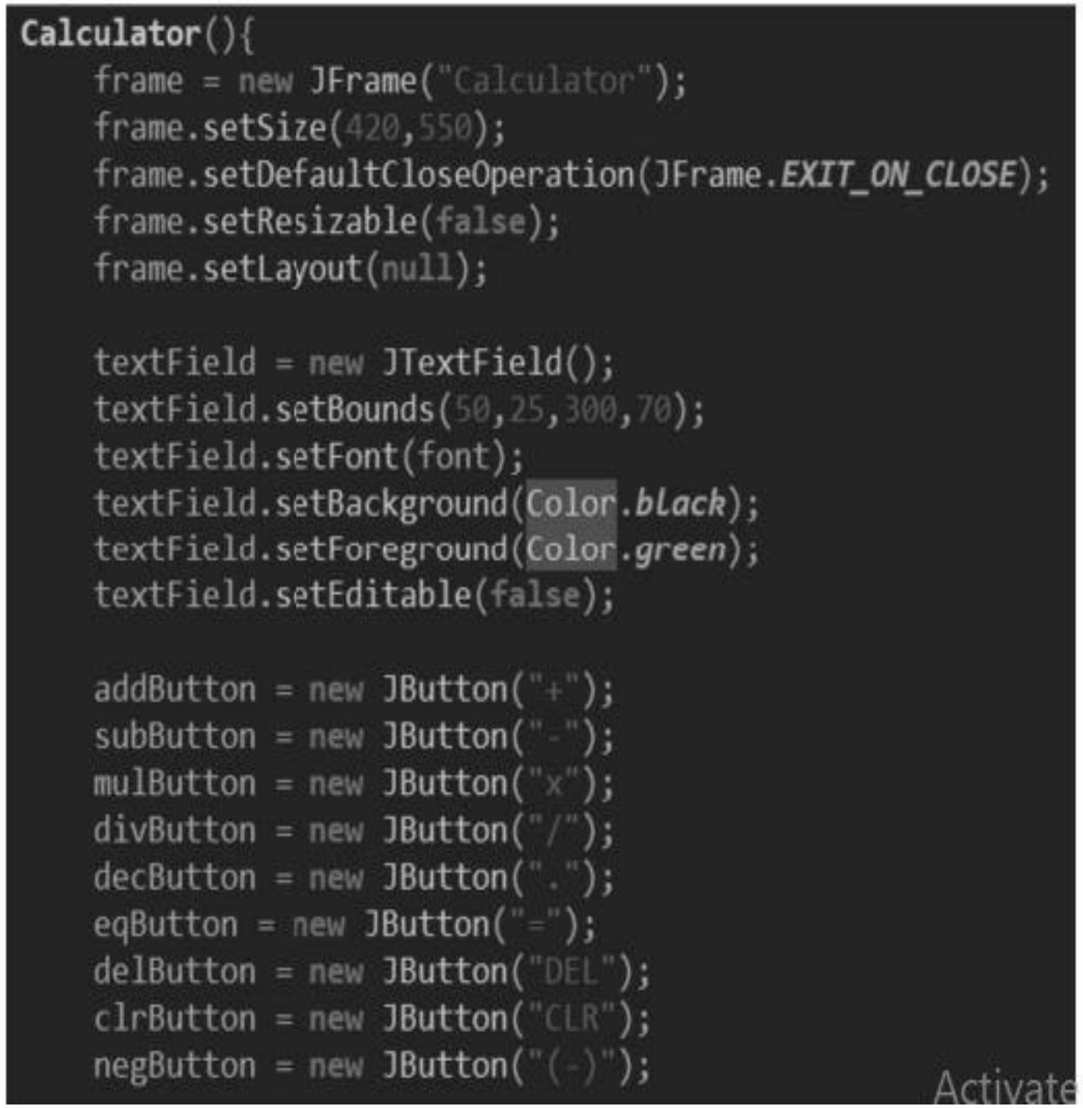
Calculator App Example



Calculator App Example

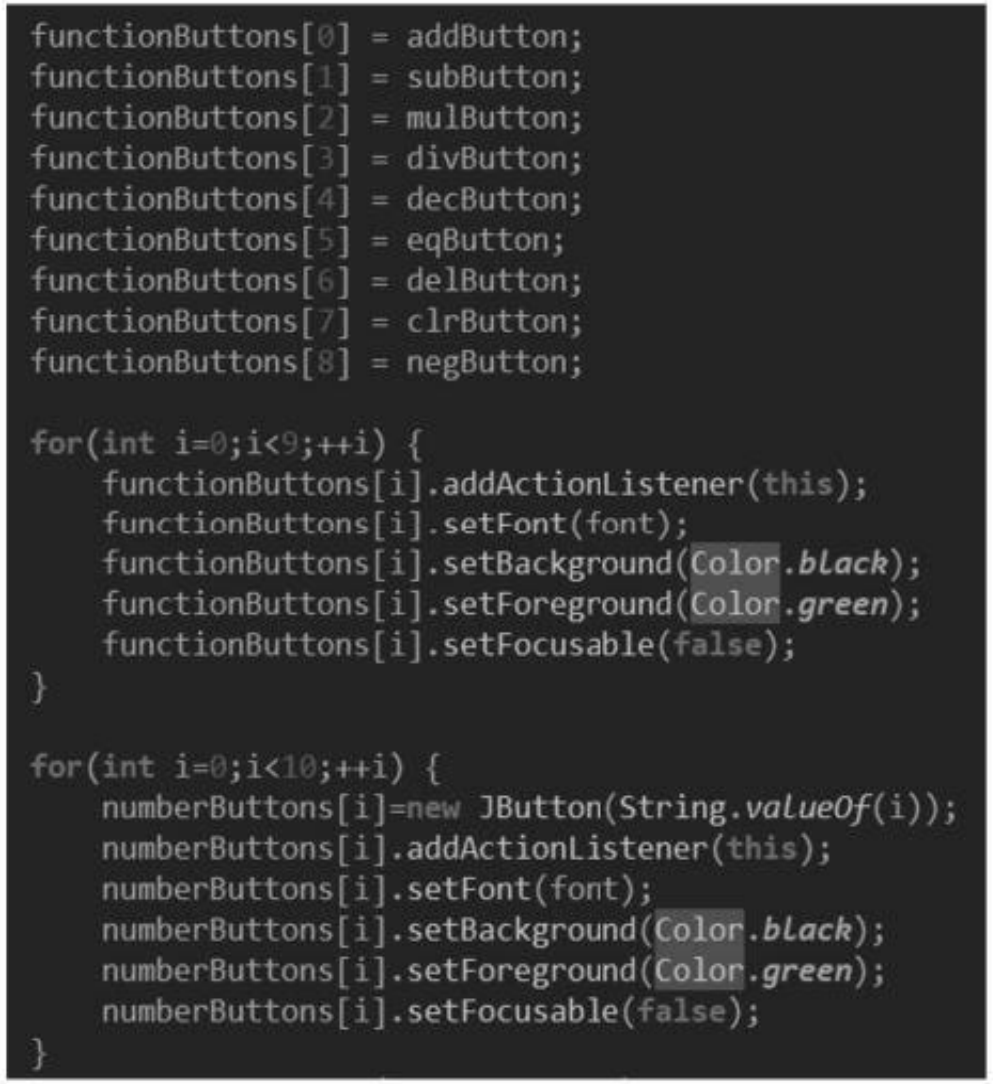


```
public class Calculator implements ActionListener{

    JFrame frame;
    JTextField textField;

    JButton[] numberButtons=new JButton[10];
    JButton[] functionButtons=new JButton[9];
    JButton addButton,subButton,mulButton,divButton;
    JButton decButton,eqButton,delButton,clrButton,negButton;

    JPanel panel;
    Font font=new Font("Verdana",Font.BOLD,30);
    double num1=0,num2=0,result=0;
    char operator;
```

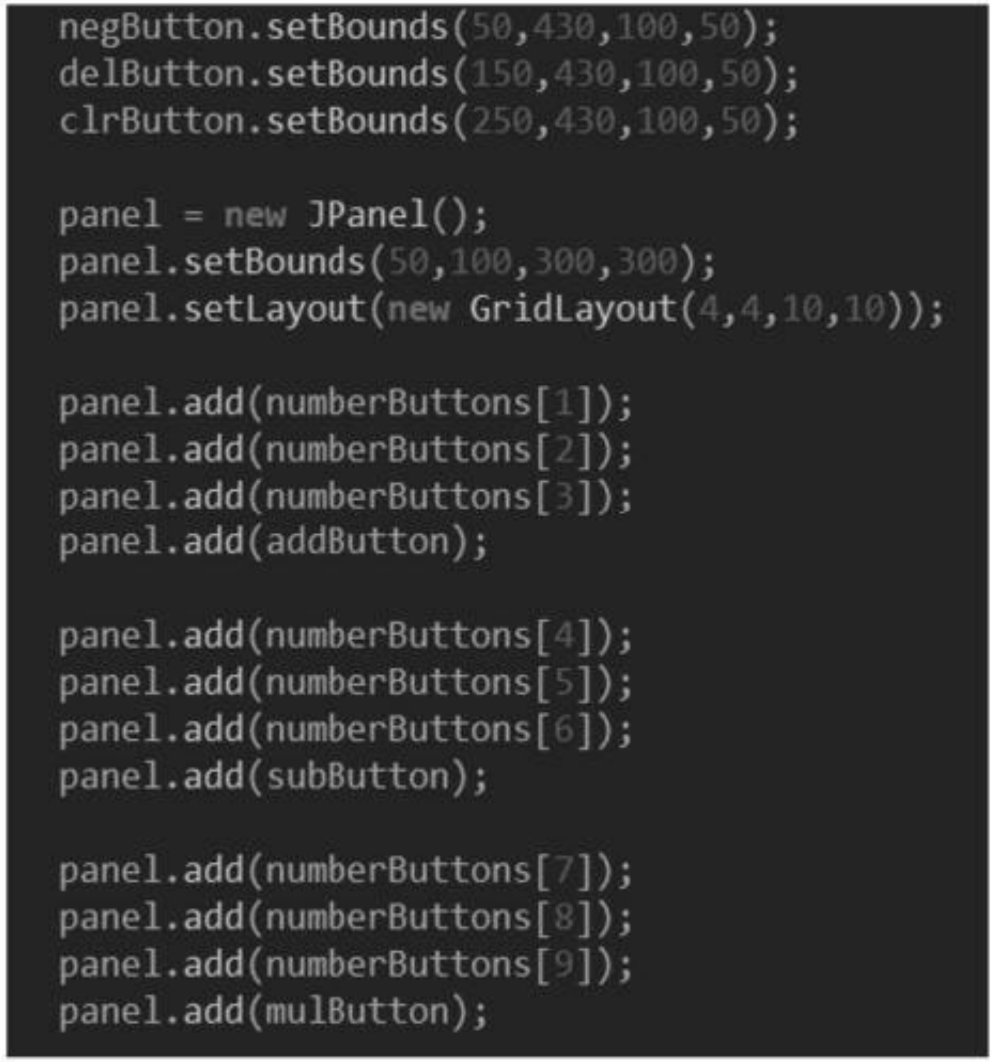


```
Calculator(){
    frame = new JFrame("Calculator");
    frame.setSize(420,550);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setResizable(false);
    frame.setLayout(null);

    textField = new JTextField();
    textField.setBounds(50,25,300,70);
    textField.setFont(font);
    textField.setBackground(Color.black);
    textField.setForeground(Color.green);
    textField.setEditable(false);

    addButton = new JButton("+");
    subButton = new JButton("-");
    mulButton = new JButton("x");
    divButton = new JButton("/");
    decButton = new JButton(".");
    eqButton = new JButton("=");
    delButton = new JButton("DEL");
    clrButton = new JButton("CLR");
    negButton = new JButton("(-)");
```

Calculator App Example-Continue

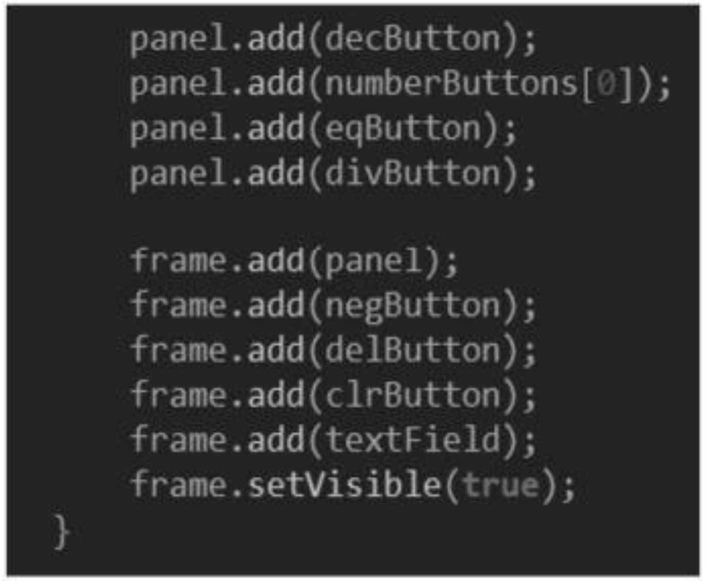


```
functionButtons[0] = addButton;
functionButtons[1] = subButton;
functionButtons[2] = mulButton;
functionButtons[3] = divButton;
functionButtons[4] = decButton;
functionButtons[5] = eqButton;
functionButtons[6] = delButton;
functionButtons[7] = clrButton;
functionButtons[8] = negButton;

for(int i=0;i<9;++i) {
    functionButtons[i].addActionListener(this);
    functionButtons[i].setFont(font);
    functionButtons[i].setBackground(Color.black);
    functionButtons[i].setForeground(Color.green);
    functionButtons[i].setFocusable(false);
}

for(int i=0;i<10;++i) {
    numberButtons[i]=new JButton(String.valueOf(i));
    numberButtons[i].addActionListener(this);
    numberButtons[i].setFont(font);
    numberButtons[i].setBackground(Color.black);
    numberButtons[i].setForeground(Color.green);
    numberButtons[i].setFocusable(false);
}
```

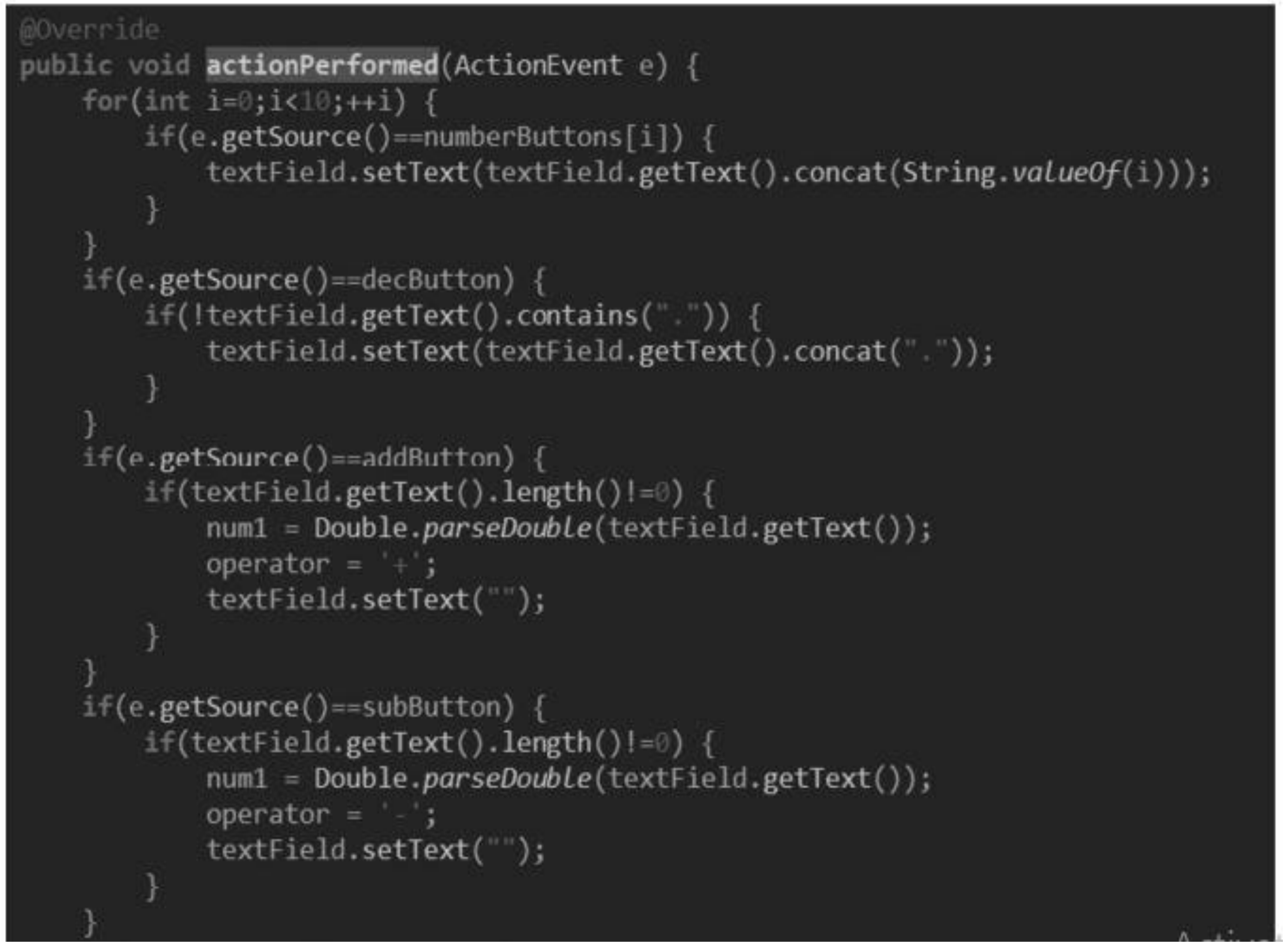


```
negButton.setBounds(50,430,100,50);
delButton.setBounds(150,430,100,50);
clrButton.setBounds(250,430,100,50);

panel = new JPanel();
panel.setBounds(50,100,300,300);
panel.setLayout(new GridLayout(4,4,10,10));

panel.add(numberButtons[1]);
panel.add(numberButtons[2]);
panel.add(numberButtons[3]);
panel.add(addButton);

panel.add(numberButtons[4]);
panel.add(numberButtons[5]);
panel.add(numberButtons[6]);
panel.add(subButton);

panel.add(numberButtons[7]);
panel.add(numberButtons[8]);
panel.add(numberButtons[9]);
panel.add(mulButton);
```

Calculator App Example-Continue

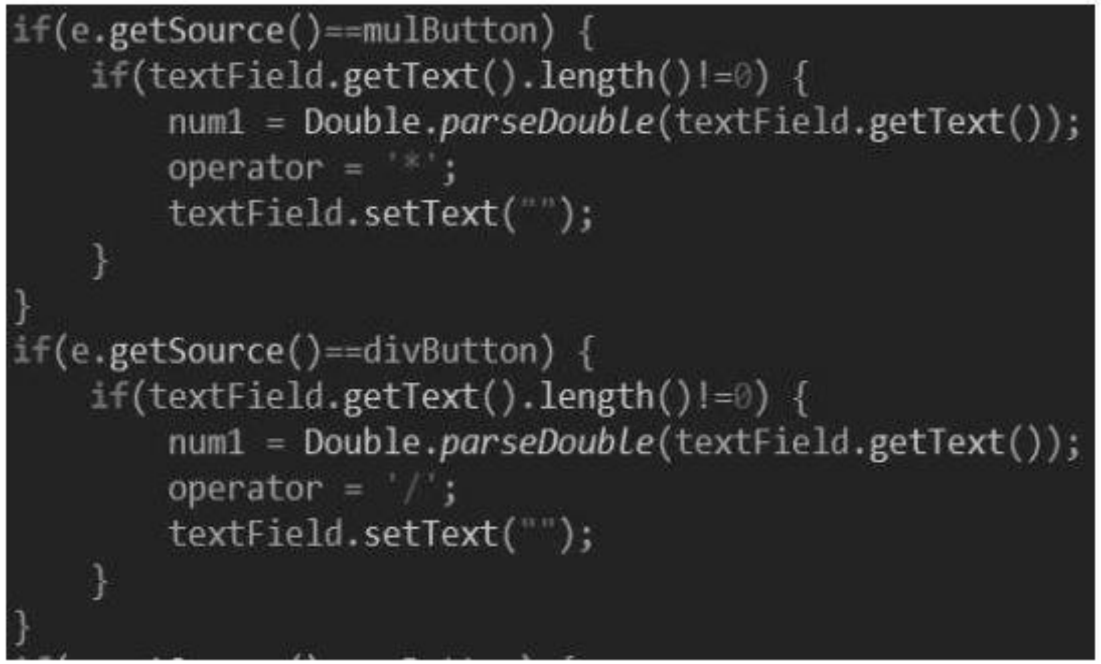


```
        panel.add(decButton);
        panel.add(numberButtons[0]);
        panel.add(eqButton);
        panel.add(divButton);

        frame.add(panel);
        frame.add(negButton);
        frame.add(delButton);
        frame.add(clrButton);
        frame.add(textField);
        frame.setVisible(true);
    }
```

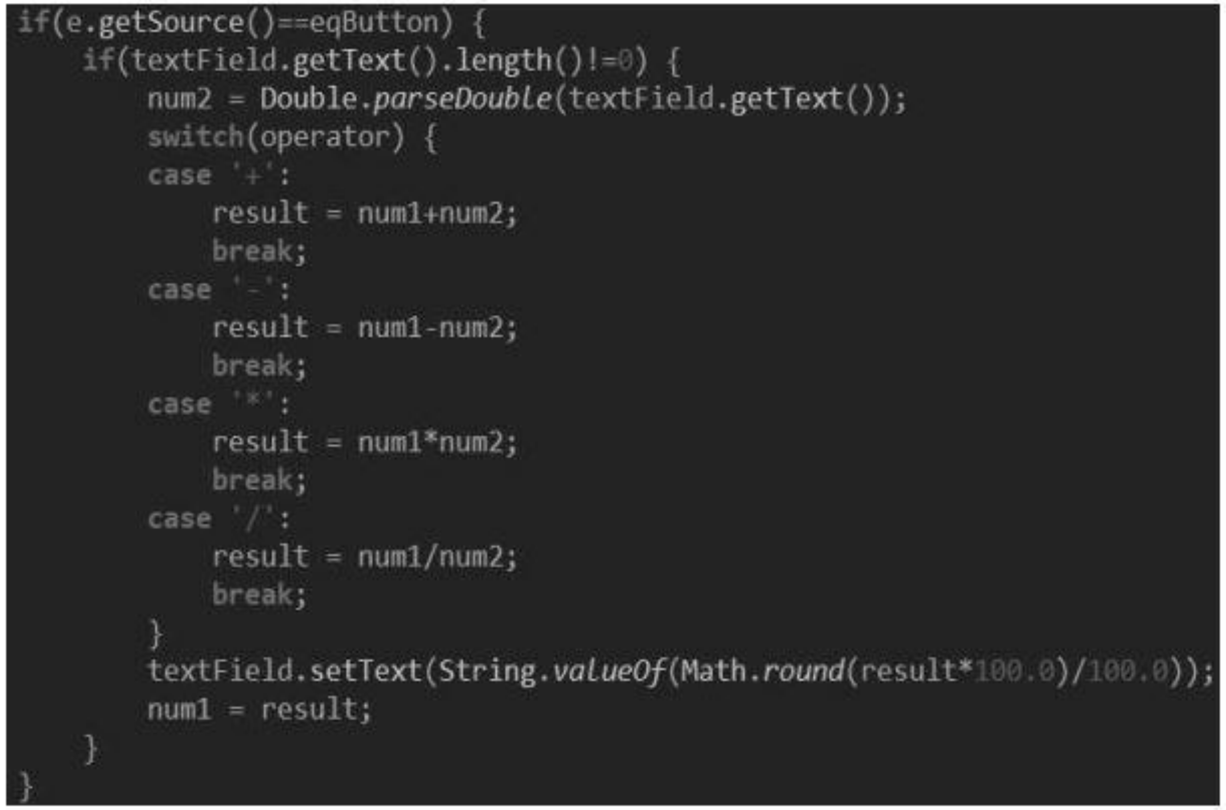


```
@Override
public void actionPerformed(ActionEvent e) {
    for(int i=0;i<10;++i) {
        if(e.getSource()==numberButtons[i]) {
            textField.setText(textField.getText().concat(String.valueOf(i)));
        }
    }
    if(e.getSource()==decButton) {
        if(!textField.getText().contains(".")) {
            textField.setText(textField.getText().concat("."));
        }
    }
    if(e.getSource()==addButton) {
        if(textField.getText().length()!=0) {
            num1 = Double.parseDouble(textField.getText());
            operator = '+';
            textField.setText("");
        }
    }
    if(e.getSource()==subButton) {
        if(textField.getText().length()!=0) {
            num1 = Double.parseDouble(textField.getText());
            operator = '-';
            textField.setText("");
        }
    }
```

Calculator App Example-Continue



```java
if(e.getSource()==mulButton) {
    if(textField.getText().length()!=0) {
        num1 = Double.parseDouble(textField.getText());
        operator = '*';
        textField.setText("");
    }
}
if(e.getSource()==divButton) {
    if(textField.getText().length()!=0) {
        num1 = Double.parseDouble(textField.getText());
        operator = '/';
        textField.setText("");
    }
}
```



```java
if(e.getSource()==eqButton) {
    if(textField.getText().length()!=0) {
        num2 = Double.parseDouble(textField.getText());
        switch(operator) {
        case '+':
            result = num1+num2;
            break;
        case '-':
            result = num1-num2;
            break;
        case '*':
            result = num1*num2;
            break;
        case '/':
            result = num1/num2;
            break;
        }
        textField.setText(String.valueOf(Math.round(result*100.0)/100.0));
        num1 = result;
    }
}
```

Java Swing MDI

- ❖ MDI stands for Multiple Document Interface.
- ❖ In an MDI application, one main window is opened, and multiple child windows are open within the main window.
- ❖ We can organize multiple internal frames in many ways. For example, we can maximize and minimize them; we can view them side by side in a tiled fashion, or we can view them in a cascaded form.
- ❖ The following are four classes we will be working with in an MDI application:
  - JInternalFrame
  - JDesktopPane
  - DesktopManager
  - JFrame

JInternalFrame

- ❖ JInternalFrame is a part of Java Swing .
- ❖ JInternalFrame is a container that provides many features of a frame which includes displaying title, opening, closing, resizing, support for menu bar, etc.
- ❖ Constructors for JInternalFrame
  - **JInternalFrame()** : creates a new non- closable, non- resizable, non- iconifiable, non- maximizable JInternalFrame with no title
  - **JInternalFrame(String t)** :creates a new non- closable, non- resizable, non- iconifiable, non- maximizable JInternalFrame with a title specified
  - **JInternalFrame(String t, boolean resizable)** :creates a new non- closable, non- iconifiable, non- maximizable JInternalFrame with a title and resizability specified
  - **JInternalFrame(String t, boolean resizable, boolean closable)** : creates a new non- iconifiable, non- maximizable JInternalFrame with a title, closability and resizability specified
  - **JInternalFrame(String t, boolean resizable, boolean closable, boolean maximizable)** :creates a new non- iconifiable JInternalFrame with a title, closability, maximizability and resizability specified
  - **JInternalFrame(String t, boolean resizable, boolean closable, boolean maximizable, boolean iconifiable)** : creates a new JInternalFrame with a title, closability, maximizability, iconifiability and resizability specified

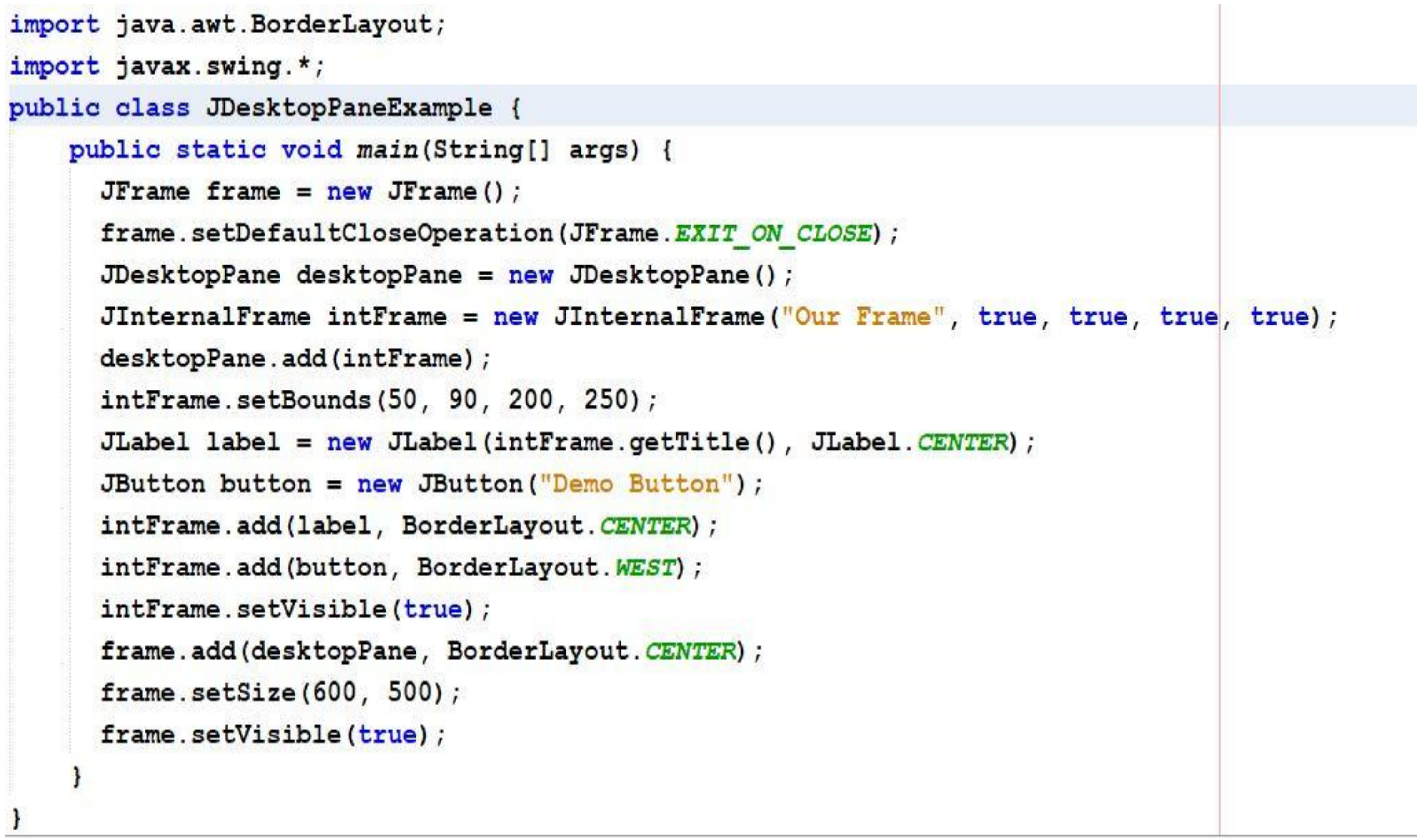
JDesktopPane

- ❖ The JDesktopPane class, can be used to create "multi-document" applications.
- ❖ A multi-document application can have many windows included in it. We do it by making the contentPane in the main window as an instance of the JDesktopPane class or a subclass.
- ❖ Internal windows add instances of JInternalFrame to the JdesktopPane instance. The internal windows are the instances of JInternalFrame or its subclasses.

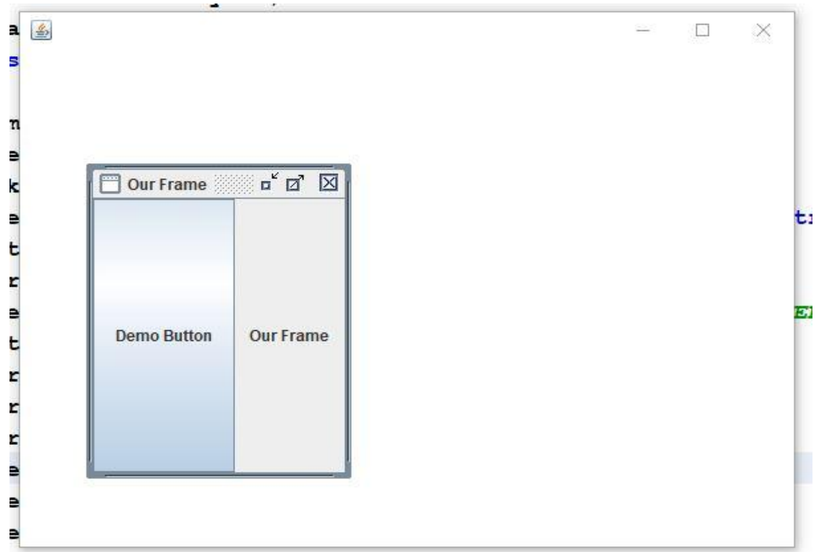
JDesktopPane

- ❖ The JDesktopPane class, can be used to create "multi-document" applications.
- ❖ A multi-document application can have many windows included in it. We do it by making the contentPane in the main window as an instance of the JDesktopPane class or a subclass.
- ❖ Internal windows add instances of JInternalFrame to the JdesktopPane instance. The internal windows are the instances of JInternalFrame or its subclasses.

JDesktopPane-Example



```
import java.awt.BorderLayout;
import javax.swing.*;
public class JDesktopPaneExample {
    public static void main(String[] args) {
      JFrame frame = new JFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JDesktopPane desktopPane = new JDesktopPane();
      JInternalFrame intFrame = new JInternalFrame("Our Frame", true, true, true, true);
      desktopPane.add(intFrame);
      intFrame.setBounds(50, 90, 200, 250);
      JLabel label = new JLabel(intFrame.getTitle(), JLabel.CENTER);
      JButton button = new JButton("Demo Button");
      intFrame.add(label, BorderLayout.CENTER);
      intFrame.add(button, BorderLayout.WEST);
      intFrame.setVisible(true);
      frame.add(desktopPane, BorderLayout.CENTER);
      frame.setSize(600, 500);
      frame.setVisible(true);
    }
}
```



Assignment

1. JBuilder and JBuilder for building java applications
2. Adapter Classes