COM2104 Advanced Programming

WK8 LECTURE: GUI

Objectives

- Know how to create a Java window and how to add components to it.
- Know how to organize components in one window by layouts.
- Know how to handle events in one window.

Components

Basic components for one window

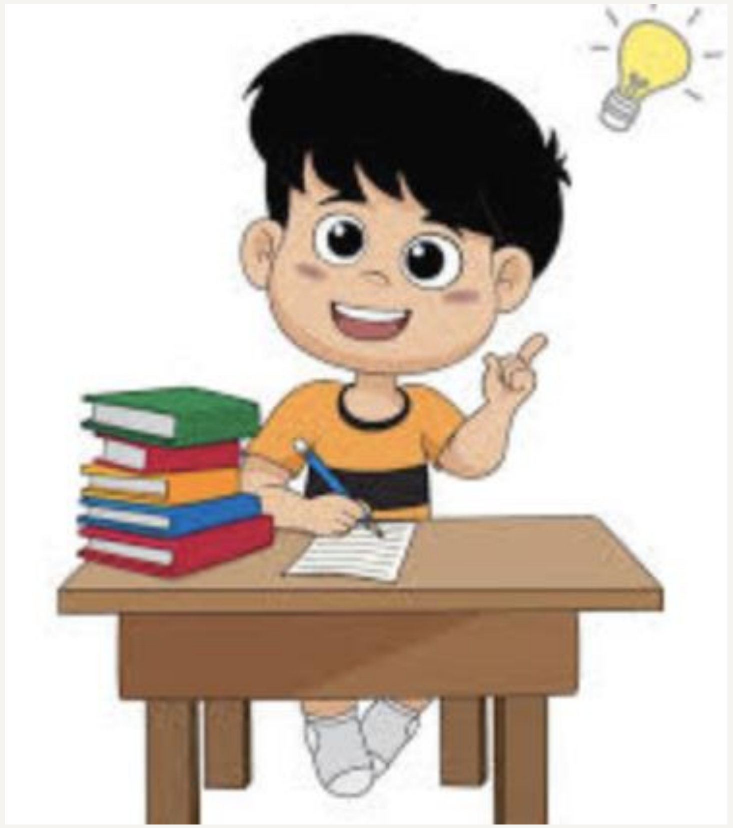
- The Component class is the superclass of all components. The components in a component class are as follows :
  - **Container**
  - **Button**
  - **Label**
  - **Checkbox**
  - **Choice**
  - **List**

Container

- **Frame:** The Frame is a container used while creating an AWT application. It can have components like title bar, menu bars, borders and also buttons, scroll bar, etc.
- **Panel:** The Panel is a Container that doesn’t include a title bar, menu, or border. It is a container that holds components like buttons, textfield, etc. Creating an instance is the way to create a Panel Container and can add components.

Important methods of Component Class

- public void add(Component c): This method inserts a component into a Container by taking a component as a parameter.
- public void setSize(int width, int height): This method sets the size of the component by taking height and width as parameters.
- public void setLayout(LayoutManager lm): This method sets the layout to set the components in a particular manner by taking LayoutManager as a parameter.
- public void setVisible(boolean status): This method sets the visibility of a component to be visible or not. If it sets to true then the component will be visible in the output else if it sets to false or not defined component won't be visible in the output.

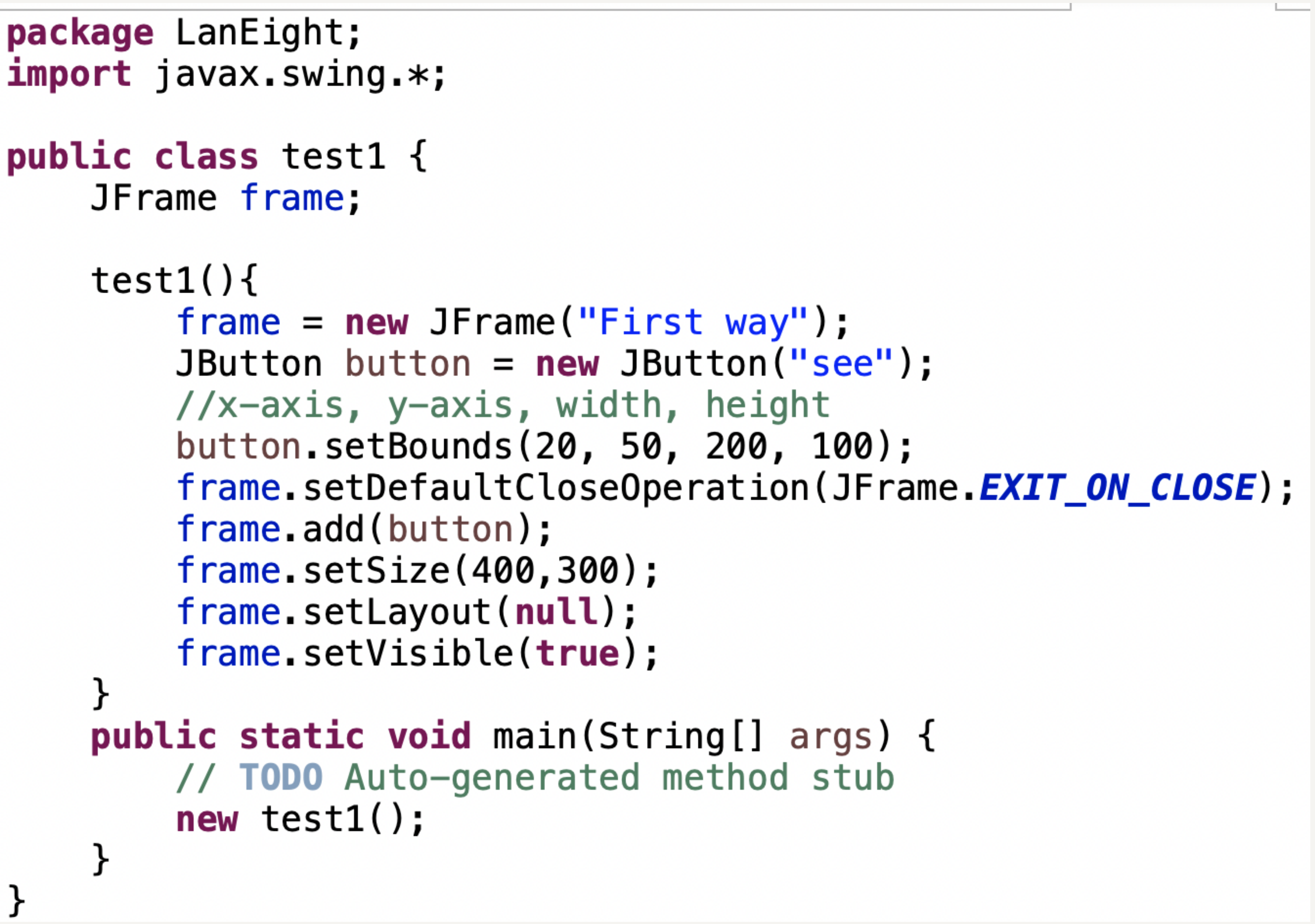


We need to create a container first so that we could add components to it.

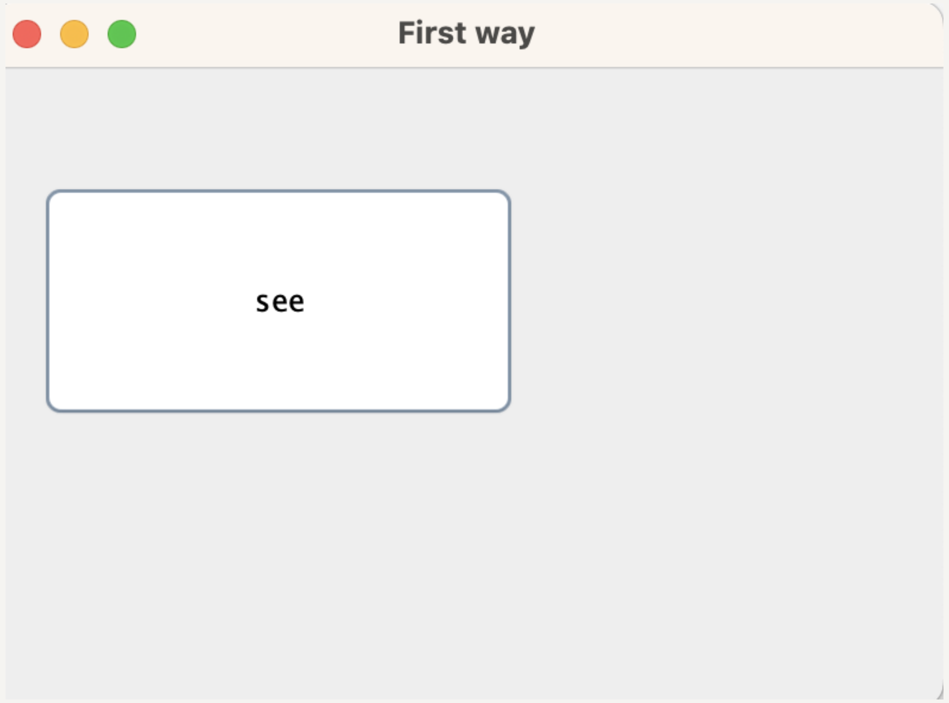
Creating Frames using Swing in Java

- Ways to create a frame:
  - By creating the object of Frame class (association)
  - By extending Frame class (inheritance)
  - Create a frame using Swing inside main()

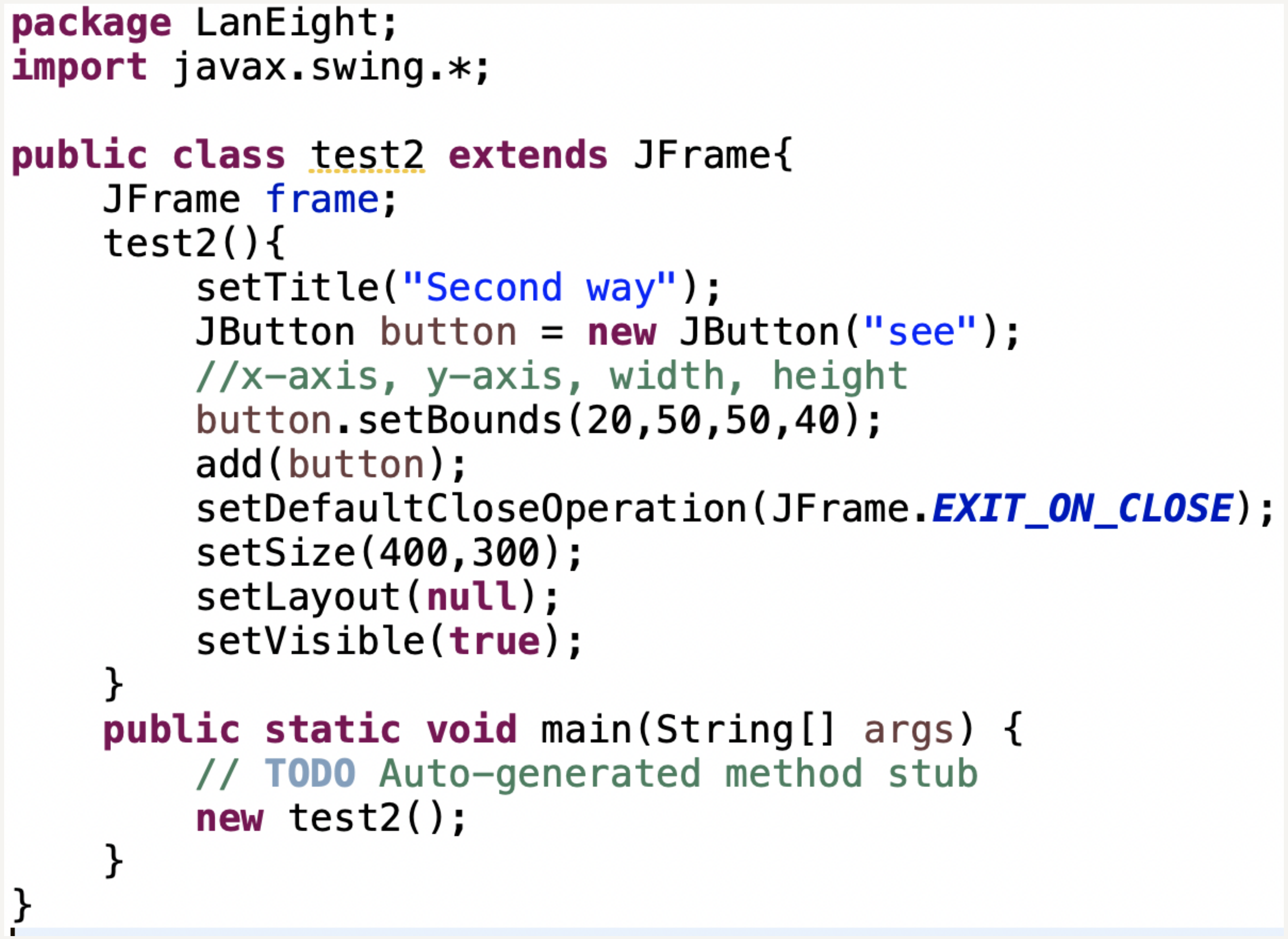
By creating the object of Frame class (association)



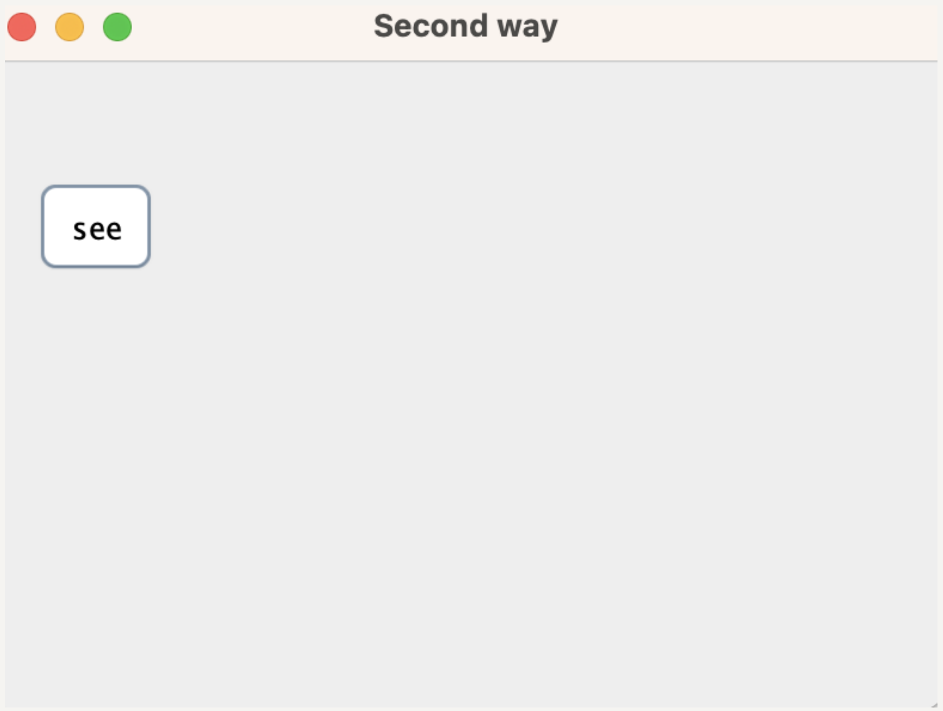
```
package LanEight;
import javax.swing.*;

public class test1 {
    JFrame frame;

    test1(){
        frame = new JFrame("First way");
        JButton button = new JButton("see");
        //x-axis, y-axis, width, height
        button.setBounds(20, 50, 200, 100);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(button);
        frame.setSize(400,300);
        frame.setLayout(null);
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new test1();
    }
}
```



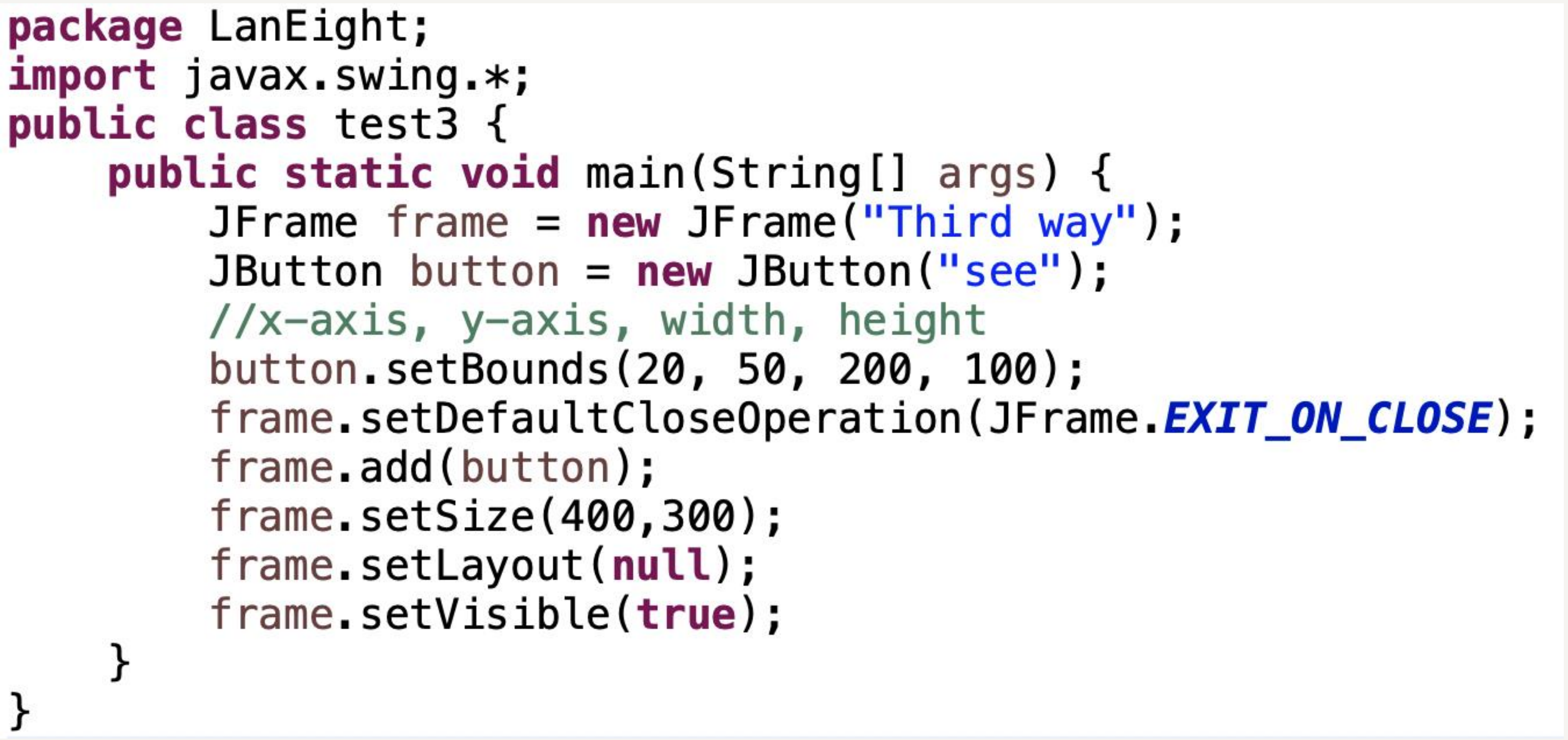
By extending Frame class (inheritance)



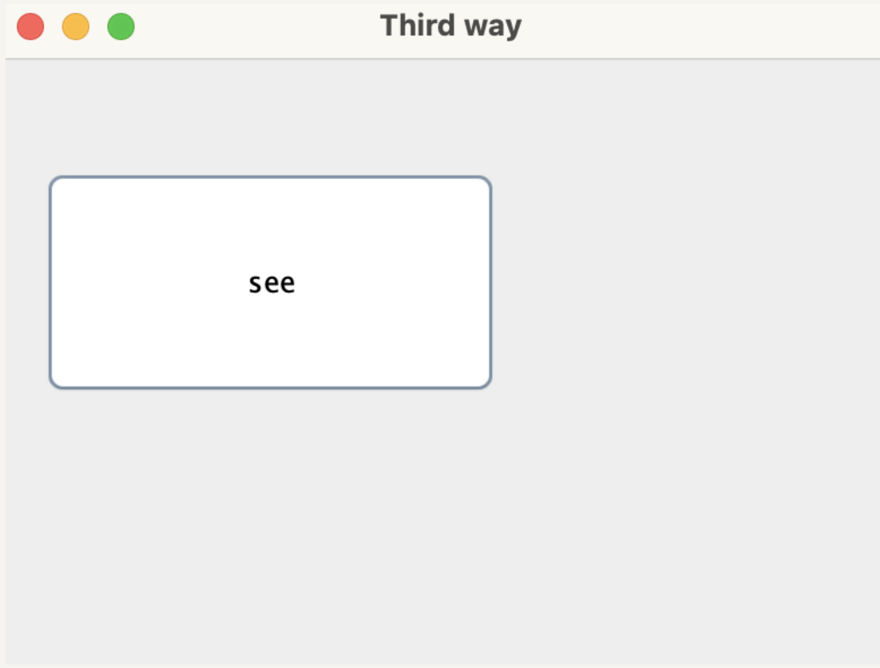
```
package LanEight;
import javax.swing.*;

public class test2 extends JFrame{
    JFrame frame;
    test2(){
        setTitle("Second way");
        JButton button = new JButton("see");
        //x-axis, y-axis, width, height
        button.setBounds(20,50,50,40);
        add(button);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400,300);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new test2();
    }
}
```



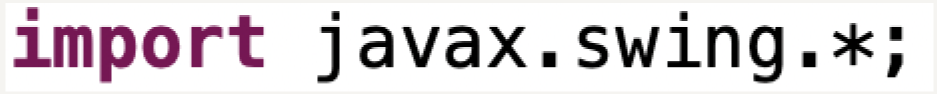
Create a frame using Swing inside main()



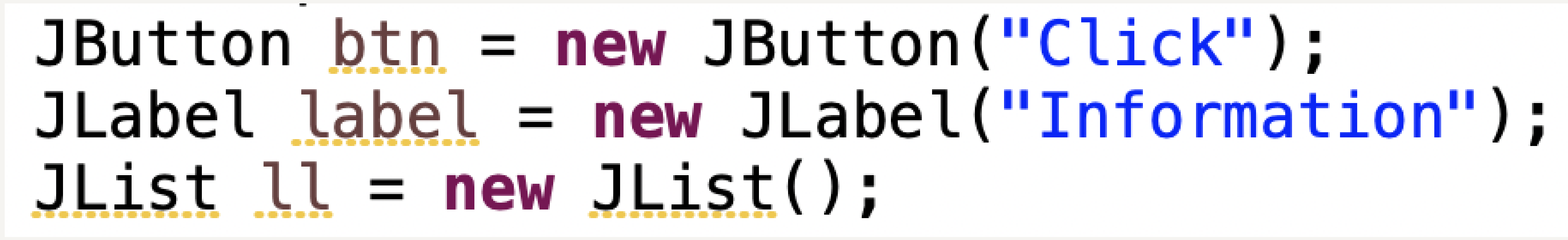
```
package LanEight;
import javax.swing.*;
public class test3 {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Third way");
        JButton button = new JButton("see");
        //x-axis, y-axis, width, height
        button.setBounds(20, 50, 200, 100);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(button);
        frame.setSize(400,300);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```



For P14-P25, you will learn how to create a window and add basic components using AWT. For button, label, and list, you can use Swing to implement them, except for Choice and Checkbox, which are AWT-only. Simply add a J to the name of these components, such as JButton, JLabel and JList.



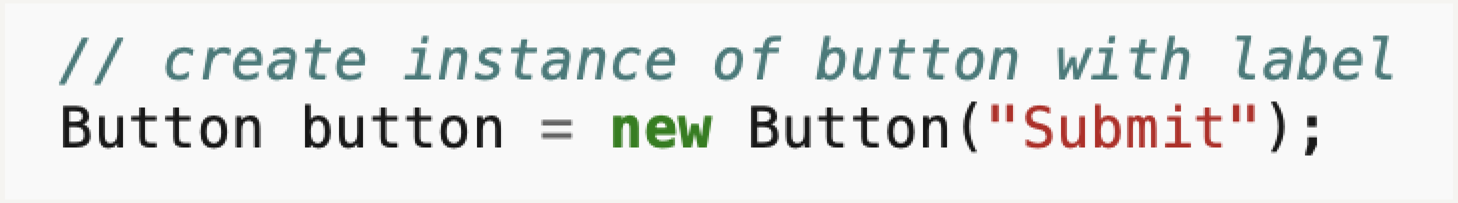
```java
import javax.swing.*;
```



```java
JButton btn = new JButton("Click");
JLabel label = new JLabel("Information");
JList ll = new JList();
```

Button

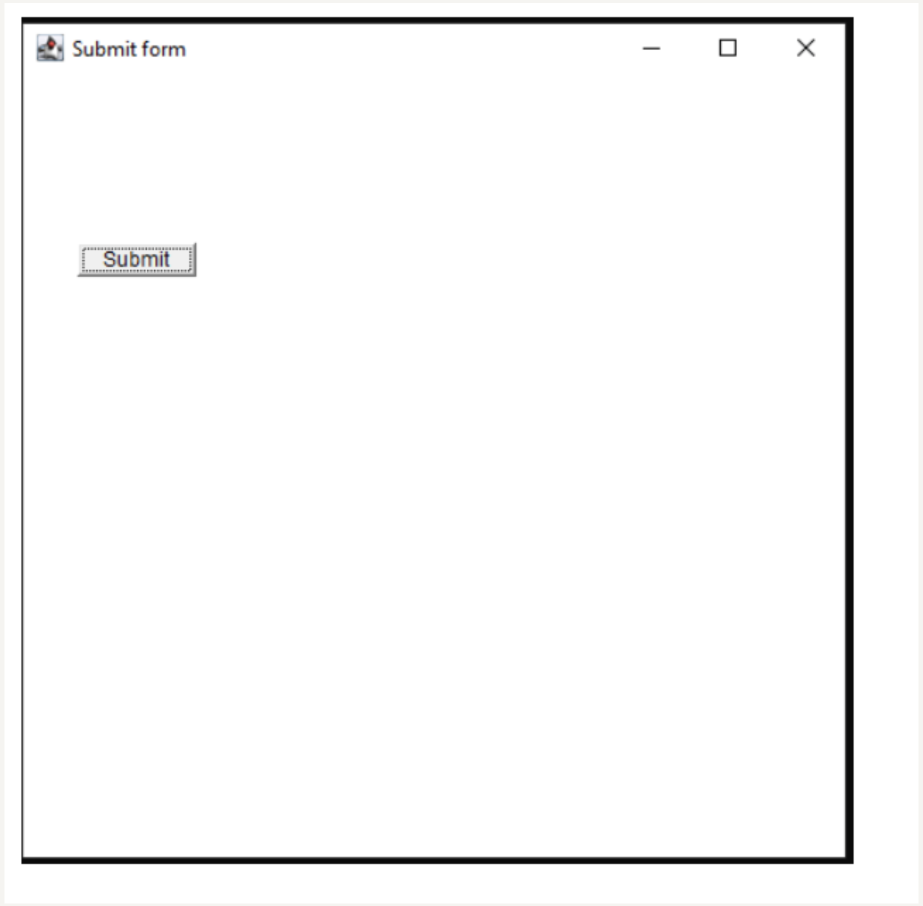
- A button is a labeled component when clicked performs an event. It is created by the Button class. When clicked it performs some action by sending an instance of ActionEvent by AWT.



```java
// create instance of button with label
Button button = new Button("Submit");
```

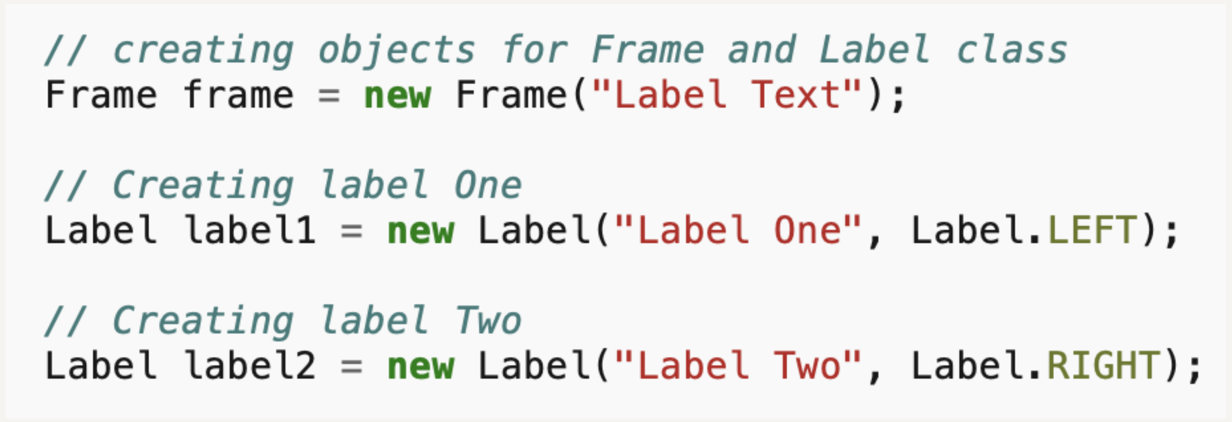
Example

```java
import java.awt.*;

// Driver Class
class SubmitButton extends Frame {
      // main function
    public static void main(String[] args)
    {
        // create instance of frame with label
        Frame frame = new Frame("Submit form");

        // create instance of button with label
        Button button = new Button("Submit");

        // set the position for the button in frame
        button.setBounds(40, 130, 70, 20);

        // adding button to the frame
        frame.add(button);

        // setting size for the frame
        frame.setSize(500, 500);

        // setting layout for the frame
        frame.setLayout(null);

        // visibility of frame to display the output\
         // without this output will be blank
        frame.setVisible(true);
    }
}
```



Label

- It is used to show text in the Container. It will displays text in the form of **READ-ONLY**, which cannot be changed by the user directly. We need to create an instance of Label Class to create a Label.

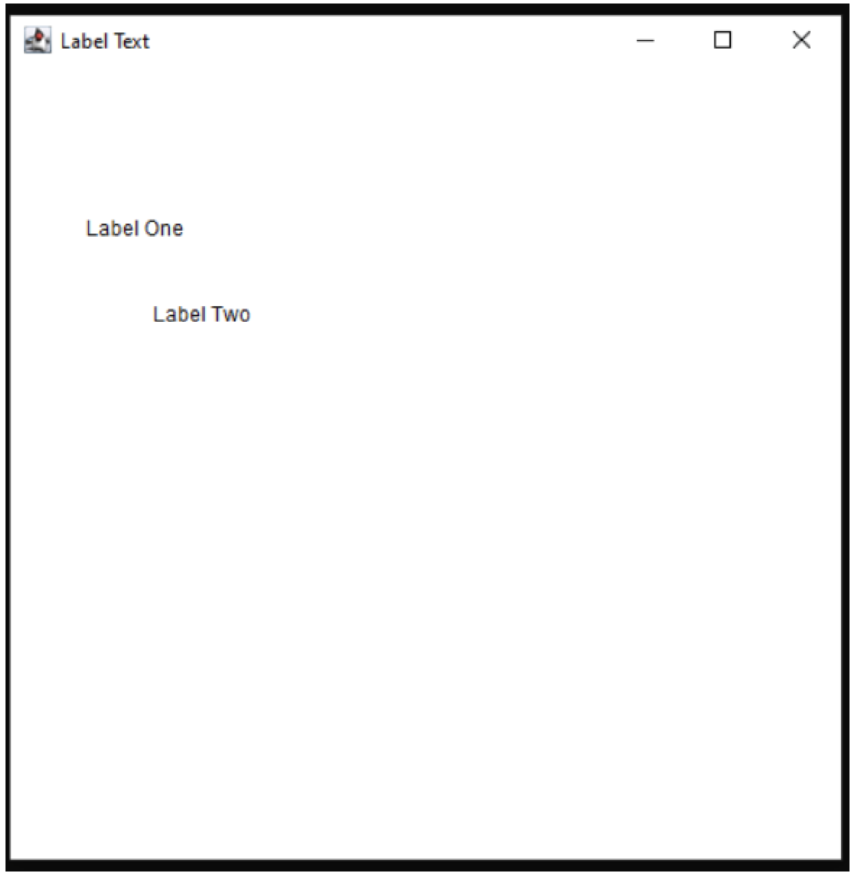


```
// creating objects for Frame and Label class
Frame frame = new Frame("Label Text");

// Creating label One
Label label1 = new Label("Label One", Label.LEFT);

// Creating label Two
Label label2 = new Label("Label Two", Label.RIGHT);
```

- **LEFT**: specifies that text should be aligned to left.
- **RIGHT**: specifies that text should be aligned to right.
- **CENTER** specifies that text should be aligned to the center.

Example

```java
import java.awt.*;

public class ShowLabelText extends Frame {
    public static void main(String args[])
    {

        // creating objects for Frame and Label class
        Frame frame = new Frame("Label Text");

        // Creating label One
        Label label1 = new Label("Label One", Label.LEFT);

        // Creating label Two
        Label label2 = new Label("Label Two", Label.RIGHT);

        // set the location of label in px
        label1.setBounds(50, 100, 100, 50);
        label2.setBounds(50, 150, 100, 50);

        // adding labels to the frame
        frame.add(label1);
        frame.add(label2);

        // setting size, layout
        // and visibility of frame
        frame.setSize(500, 500);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```



The usage of Frame

- If we want to add components to one window, we need to create one frame first.

```
// creating objects for Frame and Label class
Frame frame = new Frame("Label Text");
```

Window name

```
 // adding labels to the frame
 frame.add(label1);
 frame.add(label2);

 // setting size, layout
 // and visibility of frame
 frame.setSize(500, 500);
 frame.setLayout(null);
 frame.setVisible(true);
```

Size of the window

The window has no laout

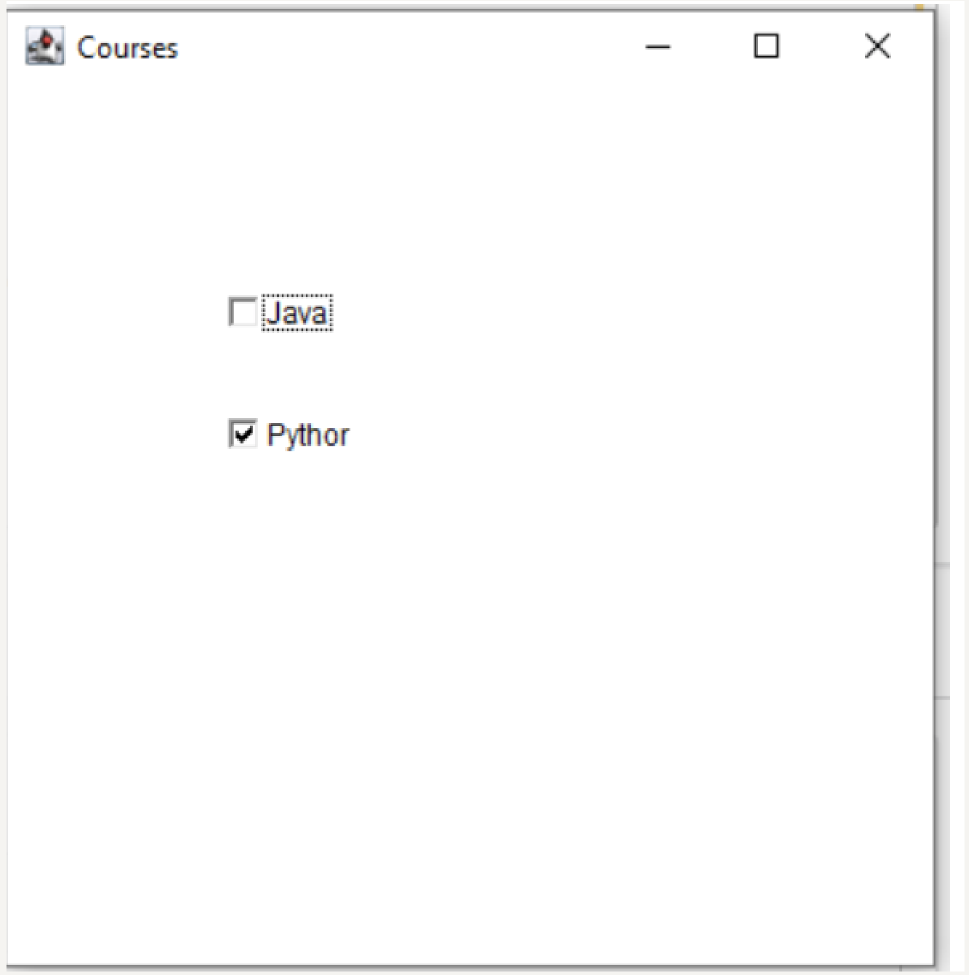
The window is visible.

Checkbox

- It is used to create a Checkbox in the Container. It can get a value either true or false by checking and unchecking the checkbox.
    - checked – returns true
    - unchecked – returns false

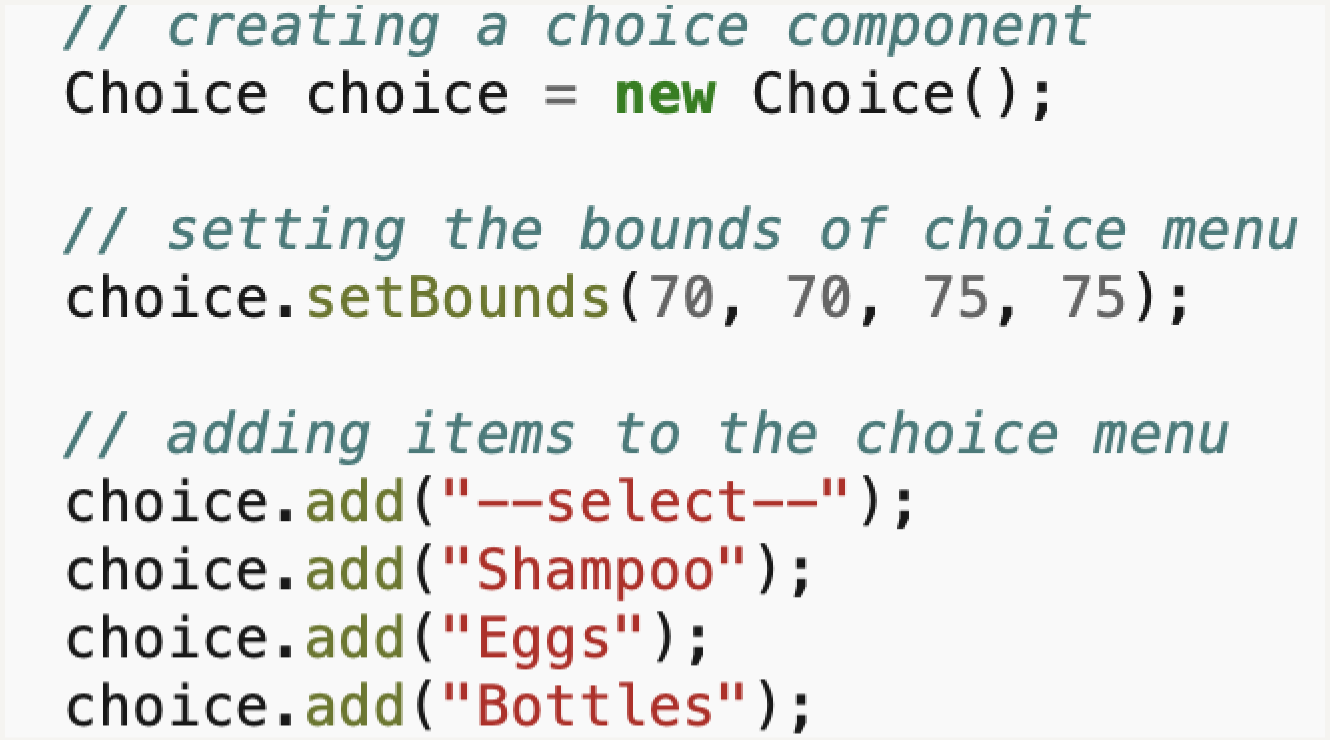
Example

```
// importing AWT class
import java.awt.*;

public class CourseCheck {
    // main method
    public static void main(String args[])
    {
        // creating the frame with the label
        Frame frame = new Frame("Courses");

        // creating checkbox java
        Checkbox java = new Checkbox("Java");

        // setting location of checkbox in frame
        java.setBounds(100, 100, 50, 50);

        // creating checkbox python with status as true
        Checkbox python = new Checkbox("Python", true);

        // setting location of checkbox in frame
        python.setBounds(100, 150, 50, 50);

        // adding checkboxes to frame
        frame.add(java);
        frame.add(python);

        // setting size, layout and
        // visibility of frame
        frame.setSize(400, 400);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```



Choice

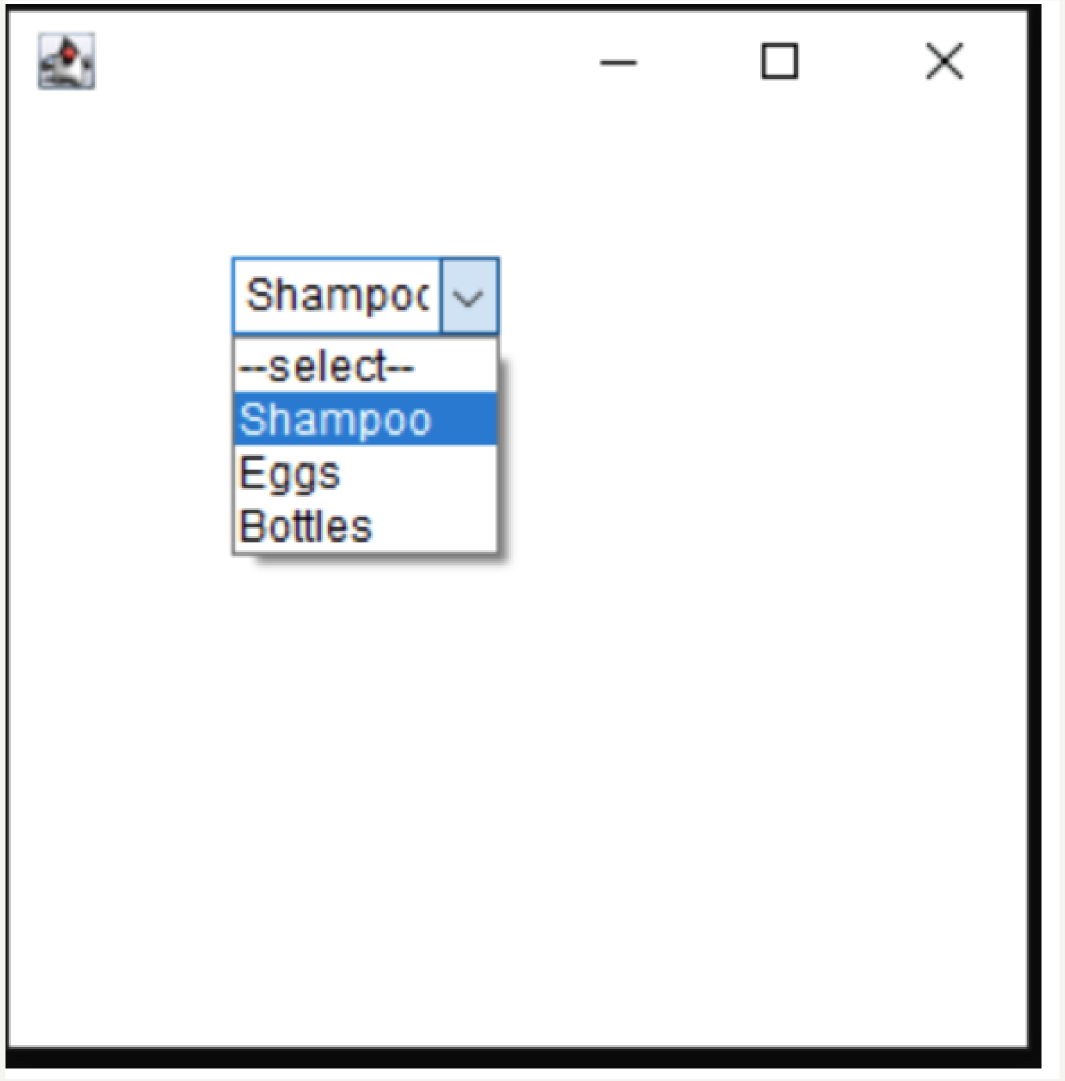
- It is used to show the popup menu to select any item from the menu items. The selected choice will be shown at the top of the menu bar. We need to create an instance of Choice Class to create a Choice.



```
// creating a choice component
Choice choice = new Choice();

// setting the bounds of choice menu
choice.setBounds(70, 70, 75, 75);

// adding items to the choice menu
choice.add("--select--");
choice.add("Shampoo");
choice.add("Eggs");
choice.add("Bottles");
```

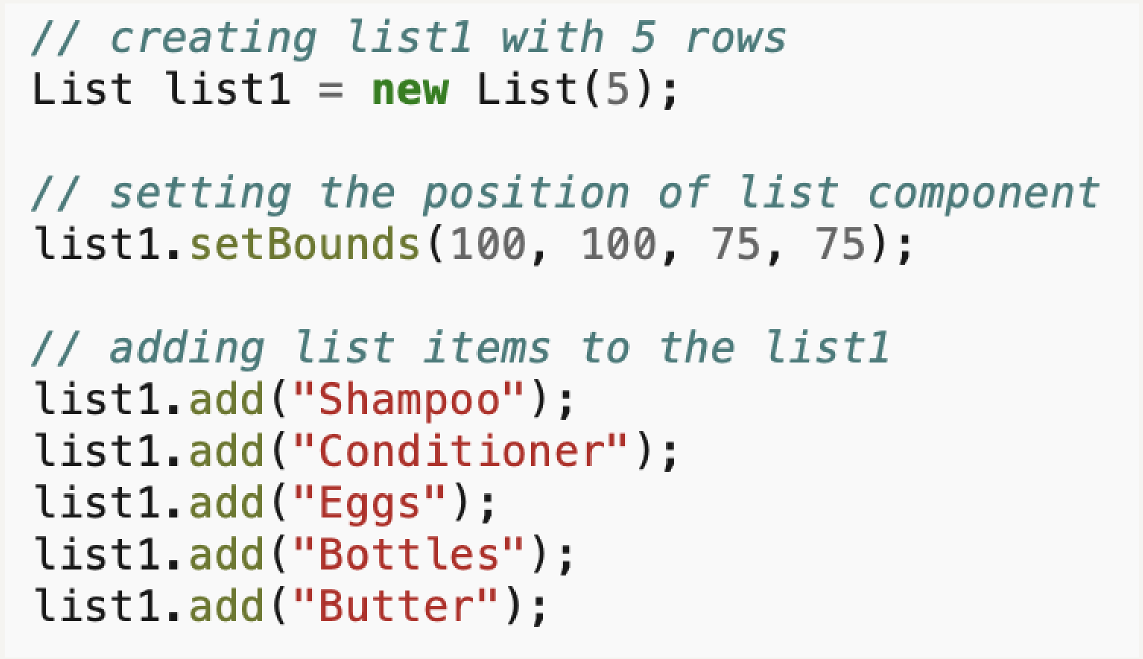
Example

```
import java.awt.*;
public class SelectItems {

    // main method
    public static void main(String args[])
    {
        // creating a Frame
        Frame frame = new Frame();

        // creating a choice component
        Choice choice = new Choice();

        // setting the bounds of choice menu
        choice.setBounds(70, 70, 75, 75);

        // adding items to the choice menu
        choice.add("--select--");
        choice.add("Shampoo");
        choice.add("Eggs");
        choice.add("Bottles");

        // adding choice menu to frame
        frame.add(choice);

        // setting size, layout
        // and visibility of frame
        frame.setSize(300, 300);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```



List

- The List Object creates a list of items in which we can choose one or multiple items at a time. We need to create an instance of List Class to create a List.



```
// creating list1 with 5 rows
List list1 = new List(5);

// setting the position of list component
list1.setBounds(100, 100, 75, 75);

// adding list items to the list1
list1.add("Shampoo");
list1.add("Conditioner");
list1.add("Eggs");
list1.add("Bottles");
list1.add("Butter");
```

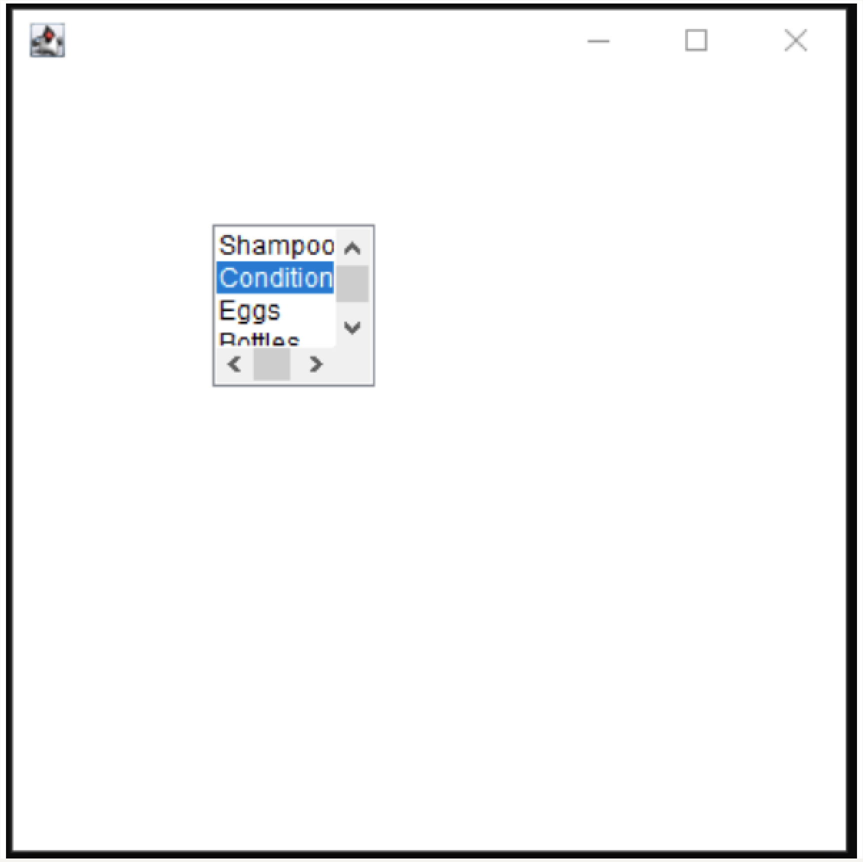
Example

```
import java.awt.*;
public class SelectList {

    // main method
    public static void main(String args[])
    {
        // creating frame1
        Frame frame1 = new Frame();

        // creating list1 with 5 rows
        List list1 = new List(5);

        // setting the position of list component
        list1.setBounds(100, 100, 75, 75);

        // adding list items to the list1
        list1.add("Shampoo");
        list1.add("Conditioner");
        list1.add("Eggs");
        list1.add("Bottles");
        list1.add("Butter");

        // adding the list1 to frame1
        frame1.add(list1);

        // setting size, layout
        // and visibility of frame1
        frame1.setSize(400, 400);
        frame1.setLayout(null);
        frame1.setVisible(true);

        // creating frame2
        Frame frame2 = new Frame();
```

```
        // creating list2 with 5 rows
        // and multi select items as true
        List list2 = new List(5, true);

        // setting the position of list component
        list2.setBounds(100, 100, 75, 75);

        // adding list items to the list2
        list2.add("Shampoo");
        list2.add("Conditioner");
        list2.add("Eggs");
        list2.add("Bottles");
        list2.add("Butter");

        // adding the list2 to frame2
        frame2.add(list2);

        // setting size, layout
        // and visibility of frame2
        frame2.setSize(400, 400);
        frame2.setLayout(null);
        frame2.setVisible(true);
    }
}
```

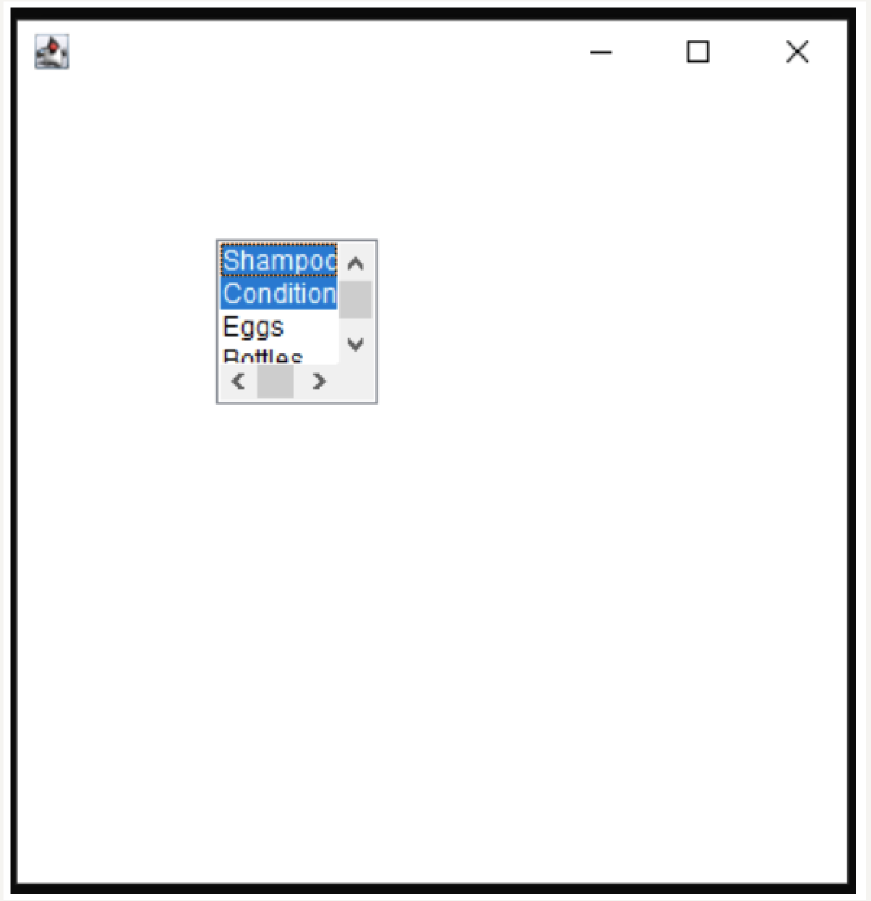
$\boxed{\texttt{true}}\rightarrow$ You could select multiple items at once

Output 1



In this output List, we can select any one item at a time.

Output 2



In this output List, we can select multiple items at a time.

Layout

Panel/JPanel

- Panel/JPanel is a container that can store a group of components.
- The main task of Panel/JPanel is to organize components, various layouts can be set in Panel/JPanel which provide better organization of components
- However, it does not have a title bar.

We could use JPanel to organize components of one window,

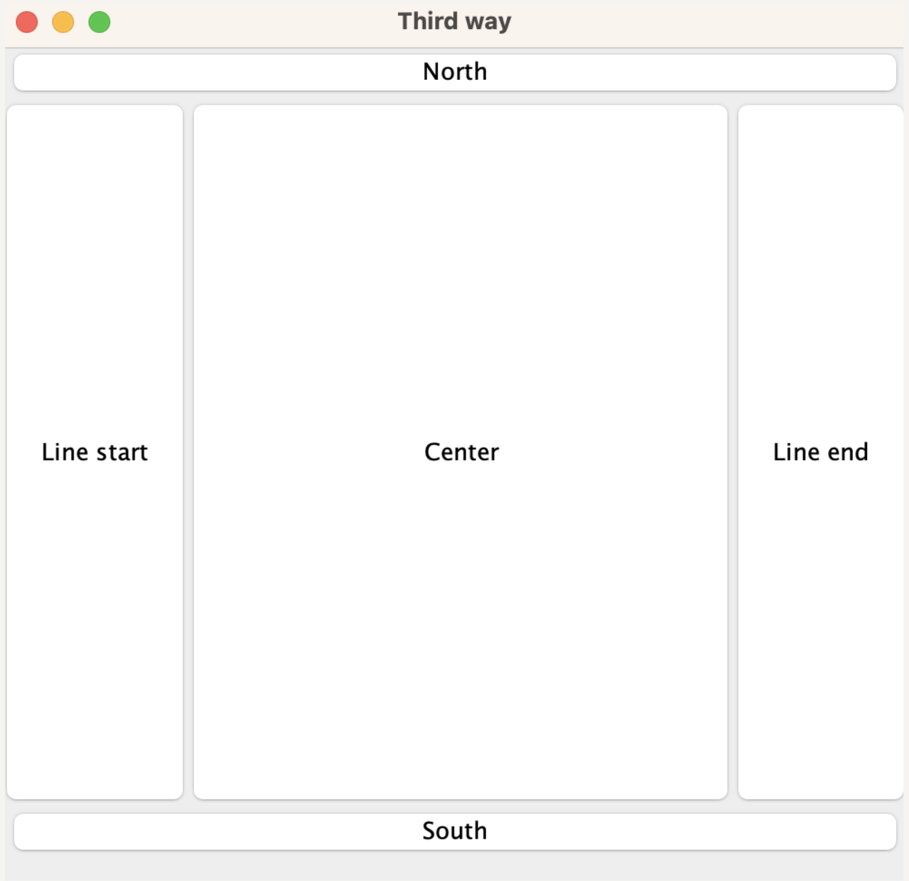
Common Layouts for organizing components

BorderLayout: The borderlayout arranges the components to fit in the five regions: east, west, north, south, and center.
FlowLayout: The FlowLayout is the default layout. It layout the components in a directional flow.
GridLayout: The GridLayout manages the components in the form of a rectangular grid.

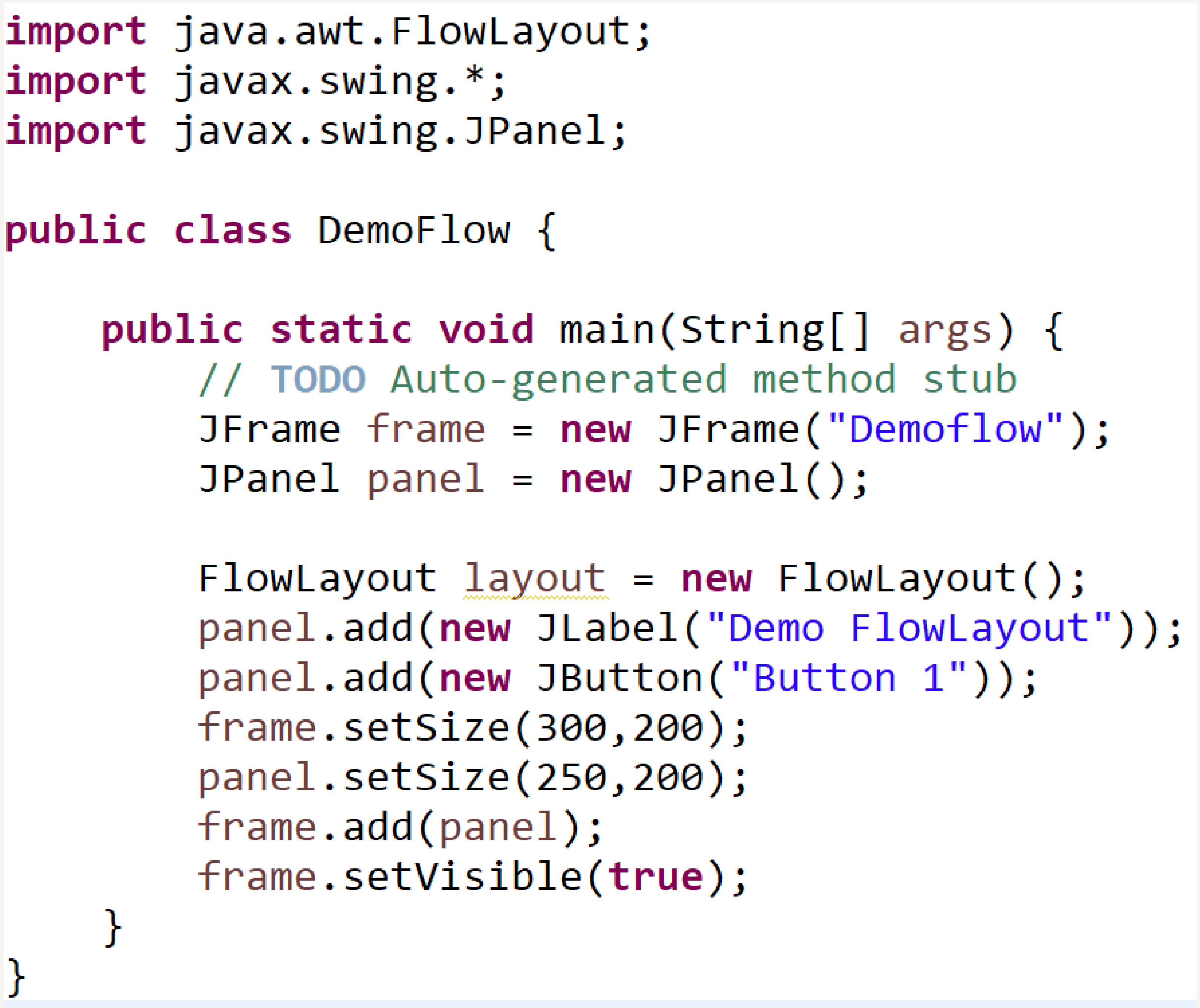
https://www.tutorialspoint.com/swing/swing_layouts.htm

BorderLayout Example

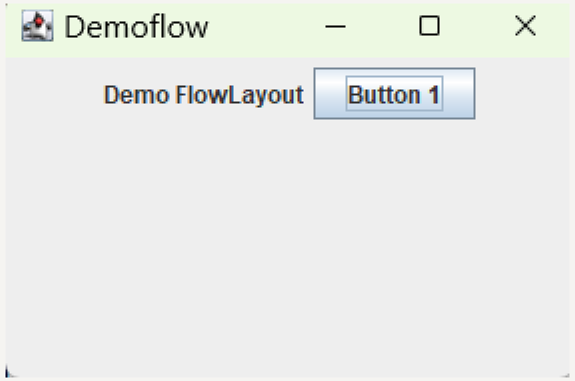
```
package LanEight;
import java.awt.BorderLayout;
import javax.swing.*;

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

        JPanel panel = new JPanel();
        panel.setLayout(layout);
        panel.add(new JButton("Center"),BorderLayout.CENTER);
        panel.add(new JButton("Line start"),BorderLayout.LINE_START);
        panel.add(new JButton("Line end"),BorderLayout.LINE_END);
        //LINE_START == WEST
        panel.add(new JButton("East"),BorderLayout.EAST);
        //LINE_END == EAST
        panel.add(new JButton("West"),BorderLayout.WEST);
        panel.add(new JButton("North"),BorderLayout.NORTH);
        panel.add(new JButton("South"),BorderLayout.SOUTH);

        frame.setSize(500,500);
        panel.setSize(500,450);
        frame.add(panel);

        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```



FlowLayout Example

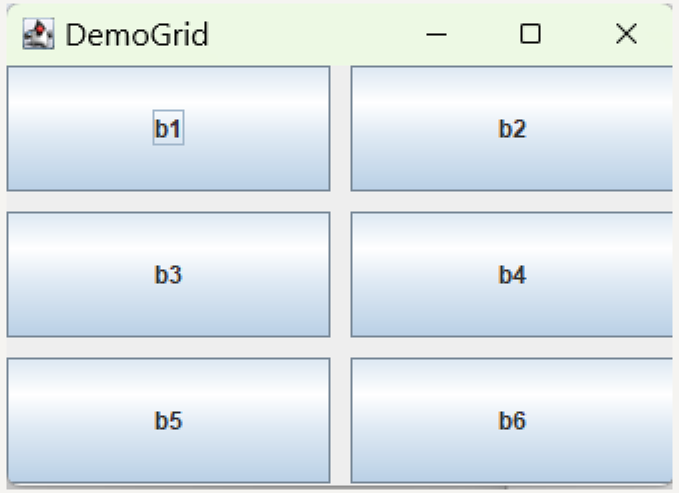


```
import java.awt.FlowLayout;
import javax.swing.*;
import javax.swing.JPanel;

public class DemoFlow {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        JFrame frame = new JFrame("Demoflow");
        JPanel panel = new JPanel();

        FlowLayout layout = new FlowLayout();
        panel.add(new JLabel("Demo FlowLayout"));
        panel.add(new JButton("Button 1"));
        frame.setSize(300,200);
        panel.setSize(250,200);
        frame.add(panel);
        frame.setVisible(true);
    }
}
```



```java
import java.awt.GridLayout;
import javax.swing.*;
public class DemoGrid {
    public static void main(String[] args) {
    JFrame frame = new JFrame("DemoGrid");
    JPanel panel = new JPanel();
    //3 rows and 2 columns
    GridLayout layout = new GridLayout(3,2);
    //height gap 10
    layout.setHgap(10);
    //vertical gap 10
    layout.setVgap(10);
    panel.setLayout(layout);

    panel.add(new JButton("b1"));
    panel.add(new JButton("b2"));
    panel.add(new JButton("b3"));
    panel.add(new JButton("b4"));
    panel.add(new JButton("b5"));
    panel.add(new JButton("b6"));

    frame.setSize(400,300);
    frame.setSize(350,250);
    frame.add(panel);
    frame.setVisible(true);
    }
}
```

GridLayout Example



Tips about using layouts

- By default, the window will have a FlowLayout layout. If you find it difficult to add different components in different rows of a window, you can put the components that should appear in the first row in one panel, then the components that should appear in the second row in another panel, and so on.
- This way you can add components in different rows. For each panel, you can also set the layout to organize the different components.

Event Listener

ActionListener

- In Java AWT, the ActionListener interface is a part of the **'java.awt.event'** package.
- When you click on a **button, menu item, or a check box**, the Java ActionListener is **called**.
- ActionListener is notified in reference to an **ActionEvent**. It only has one method, **actionPerformed().**
- The principle of an ActionListener is to record and respond to user interactions with GUI components.

Private class to create Action Listener

- Inside the class, you could create another class. This class is called inner class. To realize the action listener, we should use the inner class. The syntax is shown below:

```
Access modifier class name implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    ...
}
```

ActionListener Example

2

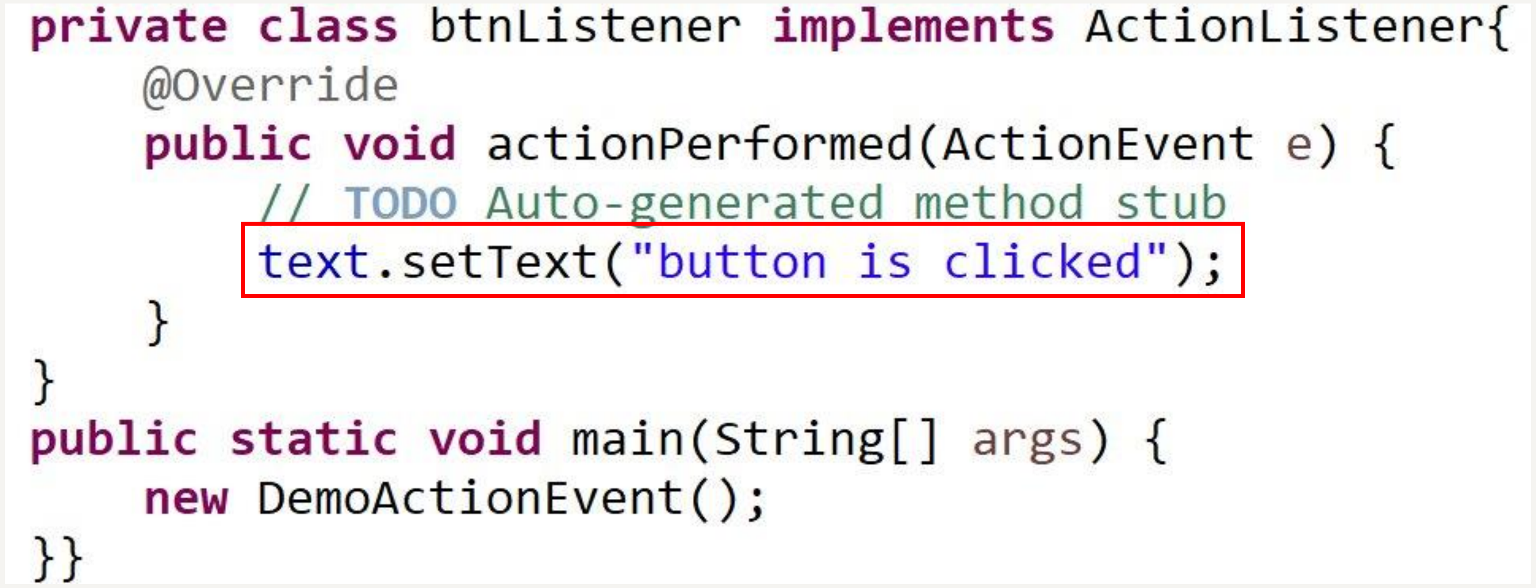
1

```
import javax.swing.*;

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

```
public class DemoActionEvent extends JFrame{
    private JButton btn;
    private JPanel panel;
    private JTextField text;

    public DemoActionEvent() {
        setLayout(new FlowLayout());
        text = new JTextField(20);
        btn = new JButton("Click me");
        panel=new JPanel();

        buildPanel();
        add(panel);
        setTitle("DemoActionListener");
        setSize(400,300);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public void buildPanel() {
        btn.setBounds(40, 50, 100, 20);
        text.setBounds(40, 20, 150, 20);
        panel.add(text);
        panel.add(btn);
        btn.addActionListener(new btnListener());

    }
```

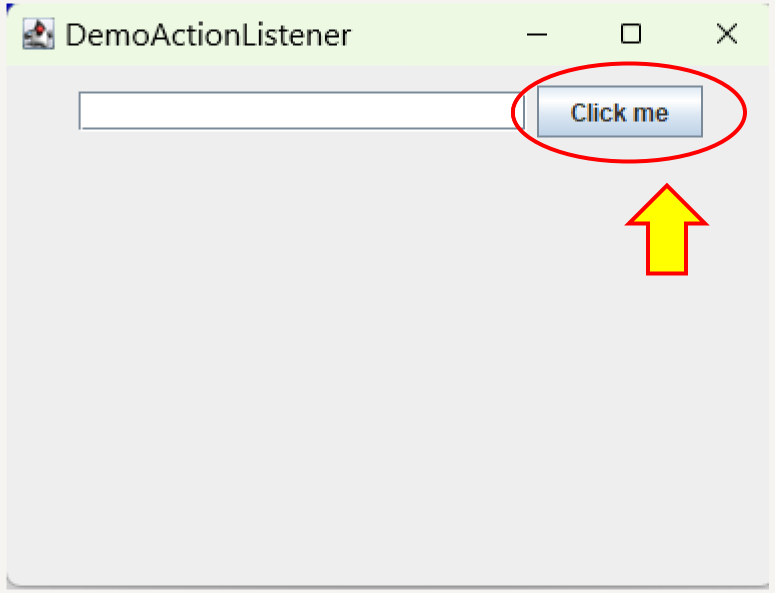
ActionListener Example: Cont.

3

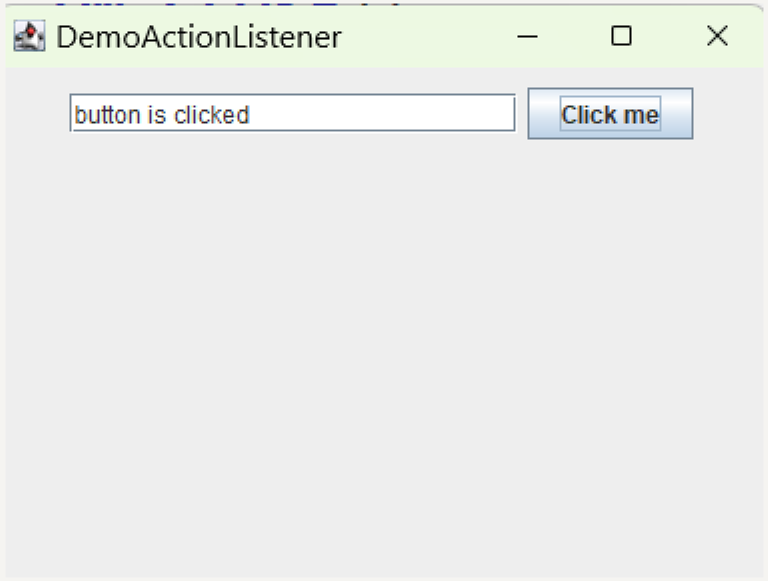


```
private class btnListener implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        text.setText("button is clicked");
    }
}
public static void main(String[] args) {
    new DemoActionEvent();
}}
```

Output



By clicking "click me", the content appeared



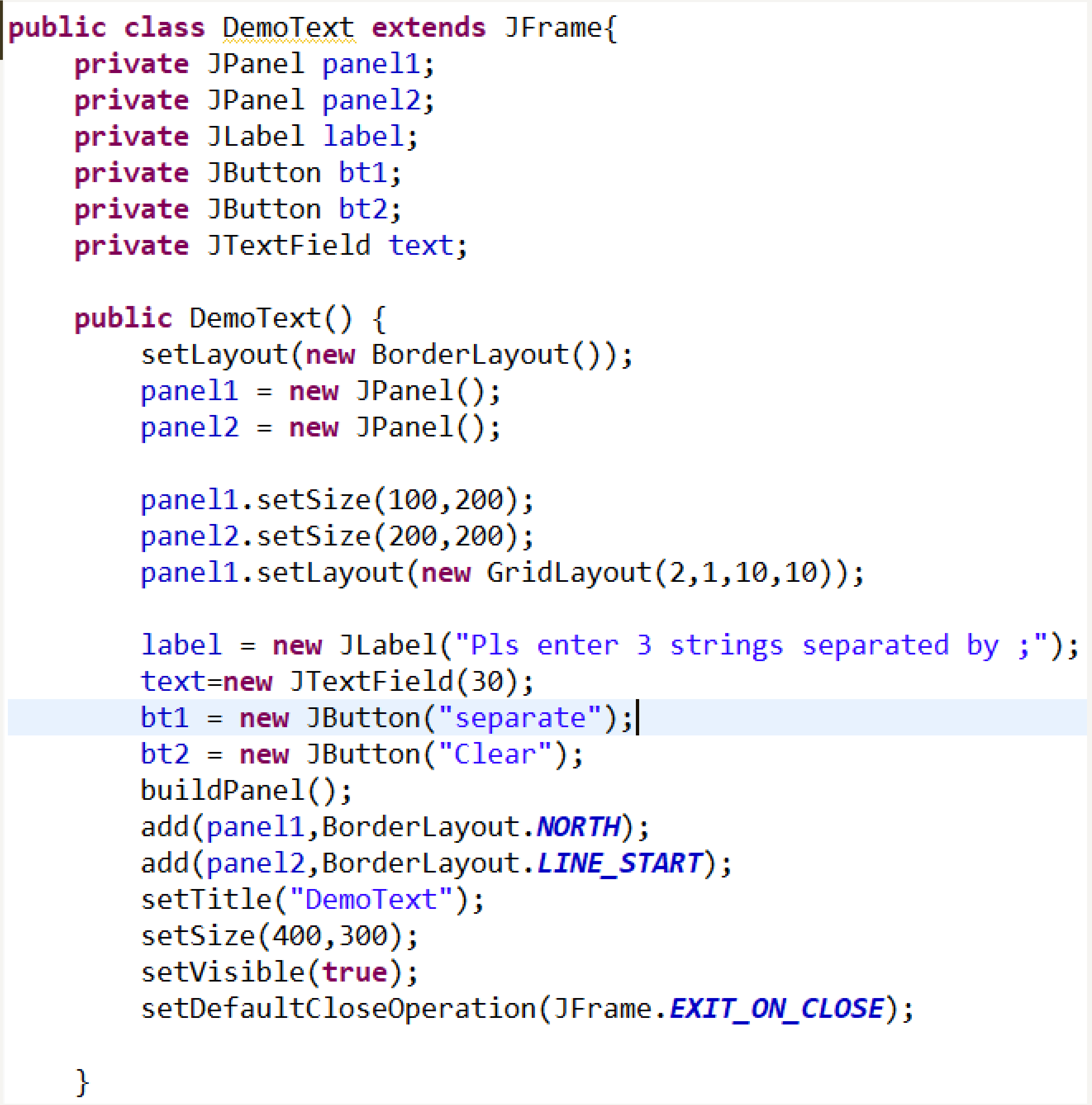
```
button is clicked
```

Methods about JTextField

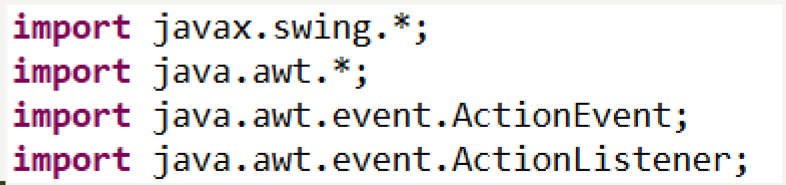
- Create one JTextField component:
  - JTextField tt = new JTextField(length of characters);
  - JTextField tt = new JTextField(length of characters); tt.setText(*content*);
- setText(content): insert the content to the textfield.
- getText(): will return the String text in textfield.
  - If you input a double, using Double.*parseDouble to convert* getText() to double.
  - If you input a int, the Integer.*parseInt to convert* getText() to int.
- getText().split(symbol): split the text by symbol and store the results in one array.
- getText().isEmpty(): return a boolean value.
  - True: TextField is empty.
  - False: TextField is not empty.

ActionListener Example 2

1

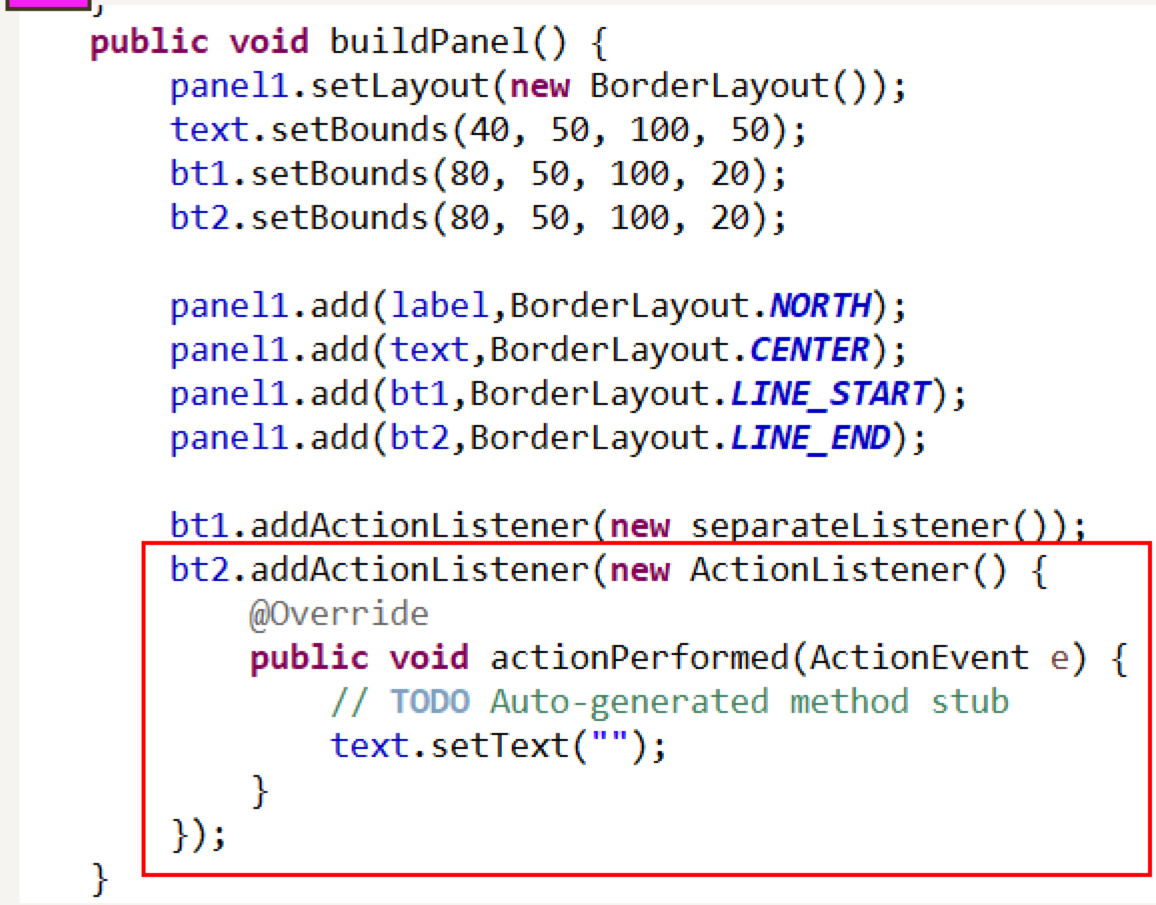


```java
public class DemoText extends JFrame{
    private JPanel panel1;
    private JPanel panel2;
    private JLabel label;
    private JButton bt1;
    private JButton bt2;
    private JTextField text;

    public DemoText() {
        setLayout(new BorderLayout());
        panel1 = new JPanel();
        panel2 = new JPanel();

        panel1.setSize(100,200);
        panel2.setSize(200,200);
        panel1.setLayout(new GridLayout(2,1,10,10));

        label = new JLabel("Pls enter 3 strings separated by ;");
        text=new JTextField(30);
        bt1 = new JButton("separate");
        bt2 = new JButton("Clear");
        buildPanel();
        add(panel1,BorderLayout.NORTH);
        add(panel2,BorderLayout.LINE_START);
        setTitle("DemoText");
        setSize(400,300);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
```



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

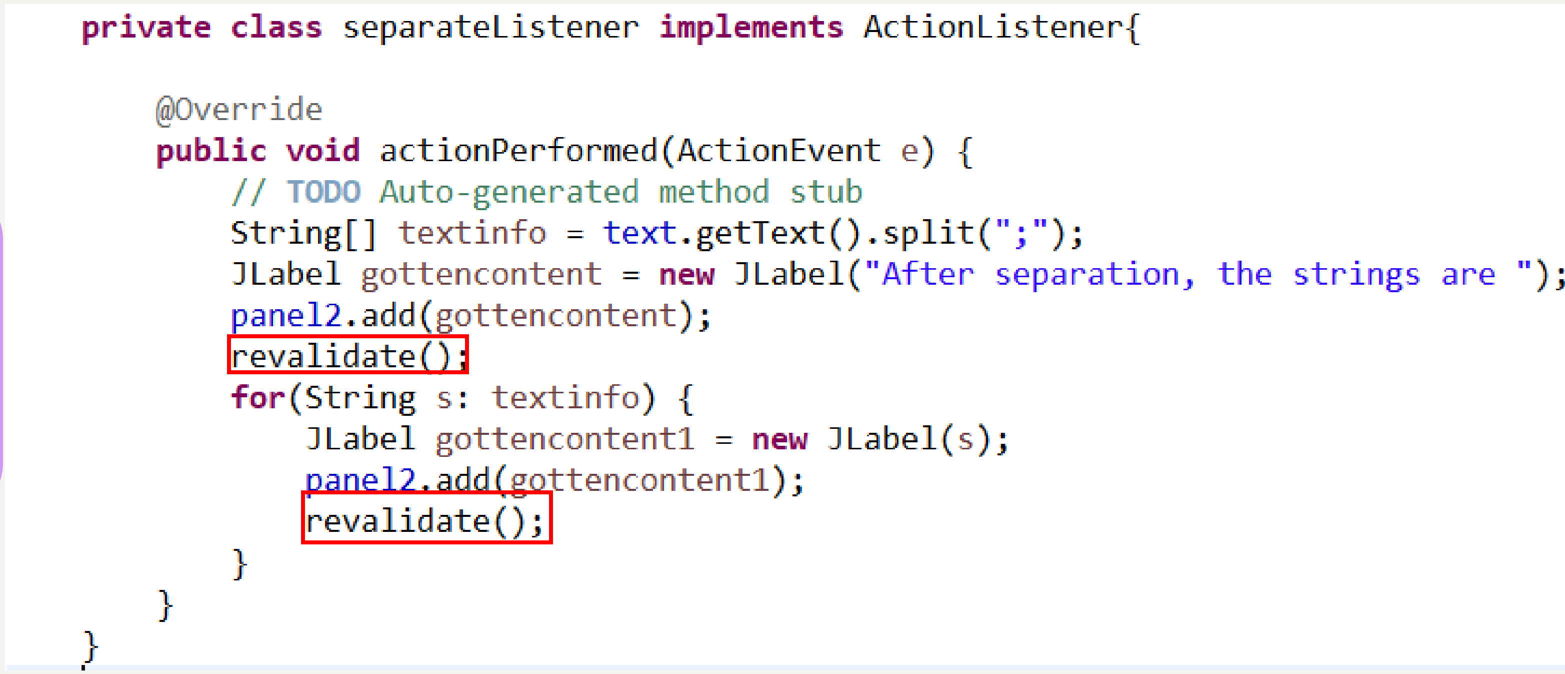
2



```java
public void buildPanel() {
    panel1.setLayout(new BorderLayout());
    text.setBounds(40, 50, 100, 50);
    bt1.setBounds(80, 50, 100, 20);
    bt2.setBounds(80, 50, 100, 20);

    panel1.add(label,BorderLayout.NORTH);
    panel1.add(text,BorderLayout.CENTER);
    panel1.add(bt1,BorderLayout.LINE_START);
    panel1.add(bt2,BorderLayout.LINE_END);

    bt1.addActionListener(new separateListener());
    bt2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            text.setText("");
        }
    });
}
```

ActionListener Example 2: Cont.

3



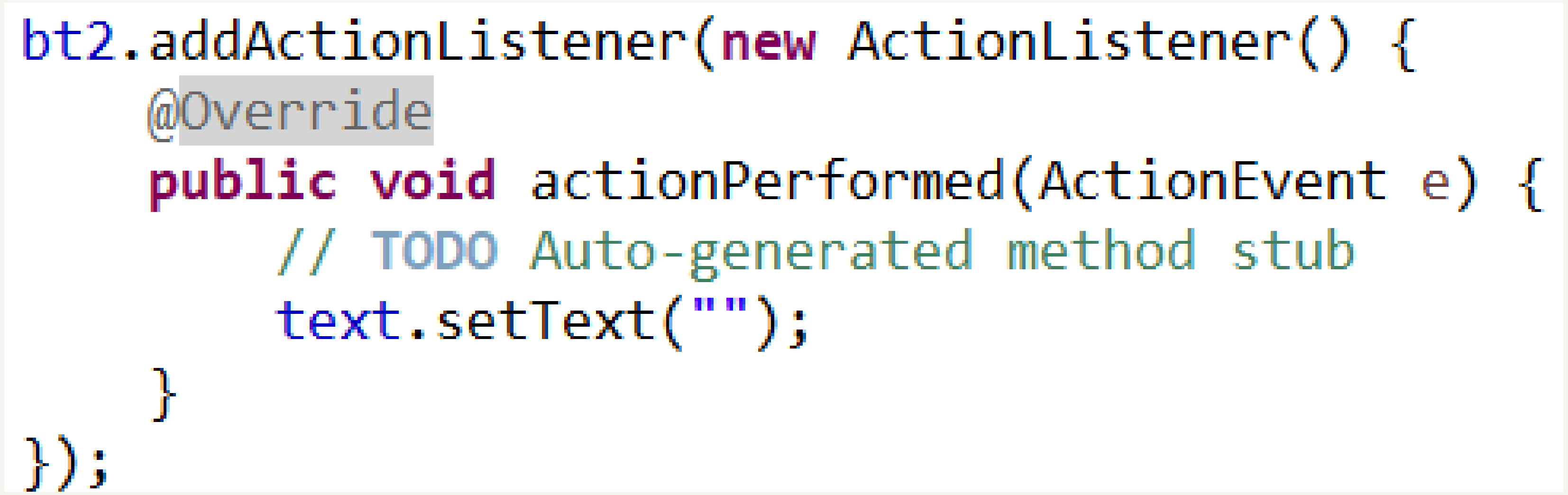
```
private class separateListener implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String[] textinfo = text.getText().split(";");
        JLabel gottencontent = new JLabel("After separation, the strings are ");
        panel2.add(gottencontent);
        revalidate();
        for(String s: textinfo) {
            JLabel gottencontent1 = new JLabel(s);
            panel2.add(gottencontent1);
            revalidate();
        }
    }
}
```

If we want to add some components to the window after some events occurred, such as clicking a button, we could use revalidate() method.

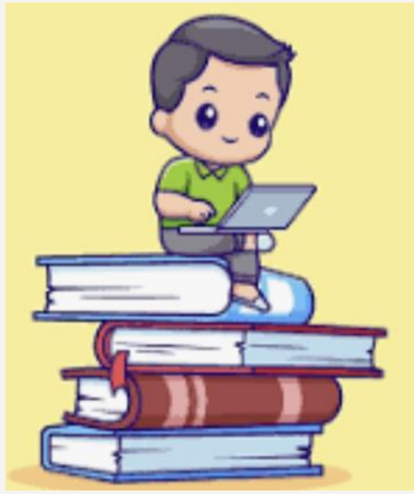
For JLabel, you could still use setText() and getText() method.

Another way to create Event Listener

We could add action listener to the component directly and still override the actionPerformed method.

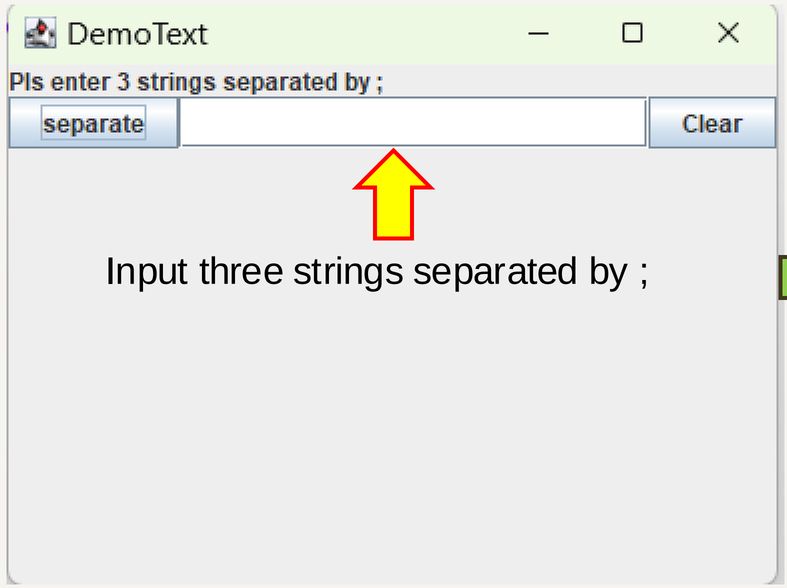


```
bt2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        text.setText("");
    }
});
```



Output

1

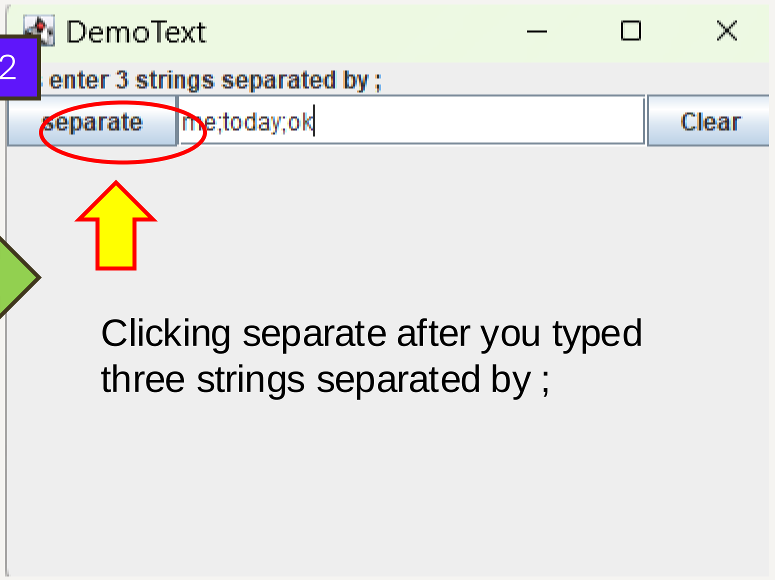


DemoText

Pls enter 3 strings separated by ;

Input three strings separated by ;

2



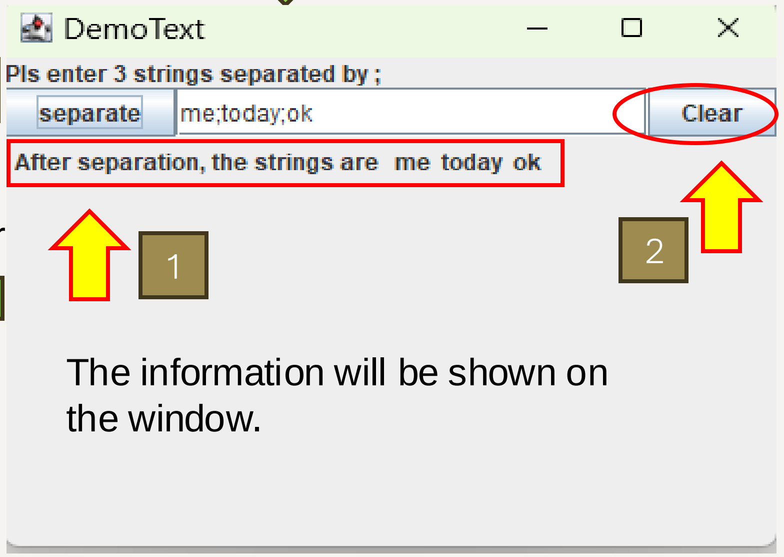
DemoText

enter 3 strings separated by ;

me;today;ok

Clicking separate after you typed three strings separated by ;

3



DemoText

Pls enter 3 strings separated by ;

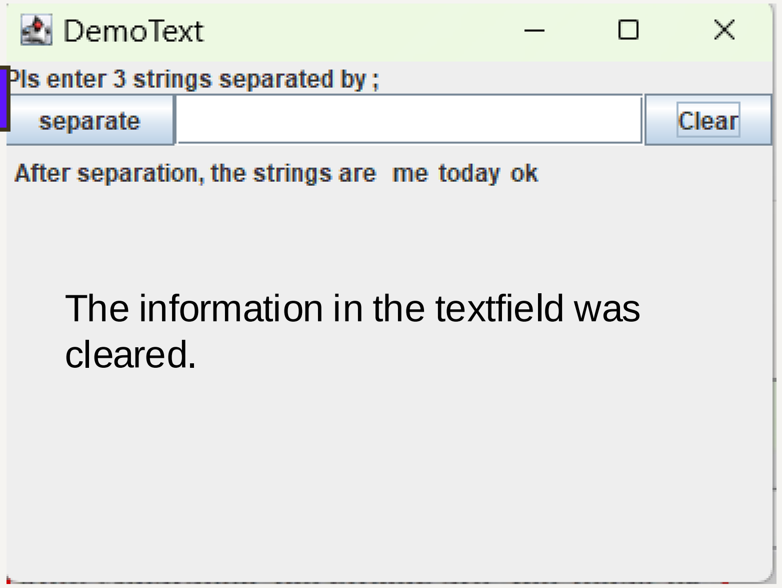
me;today;ok

After separation, the strings are me today ok

The information will be shown on the window.

Click clear

4



DemoText

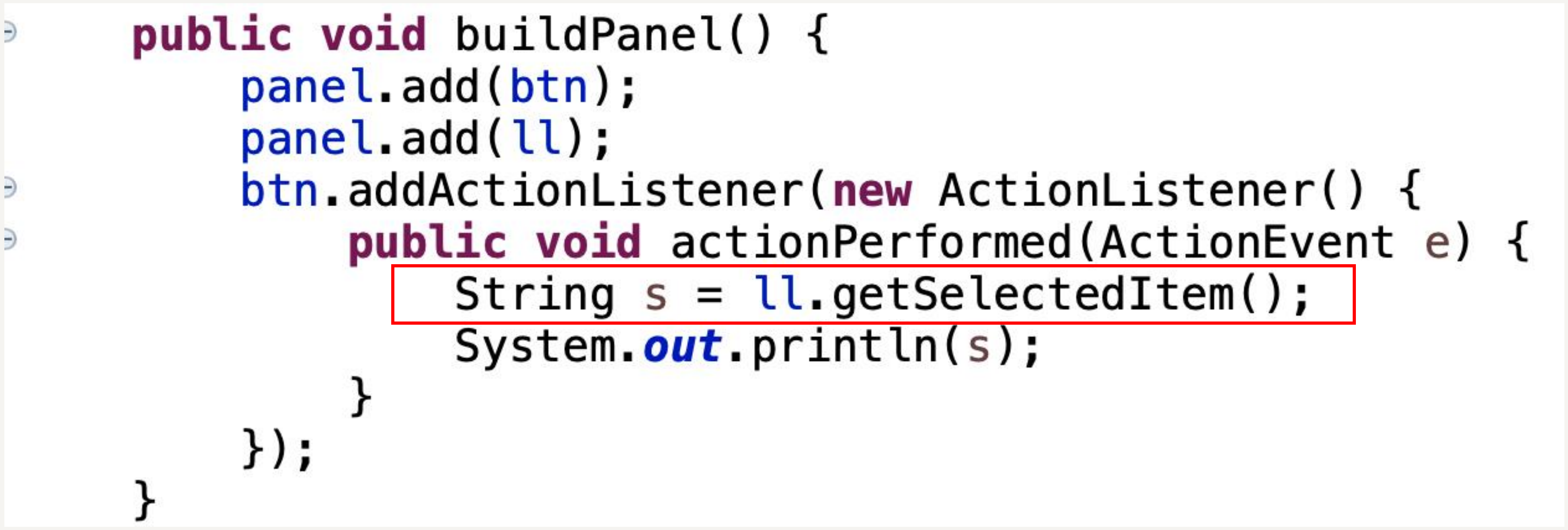
Pls enter 3 strings separated by ;

After separation, the strings are me today ok

The information in the textfield was cleared.

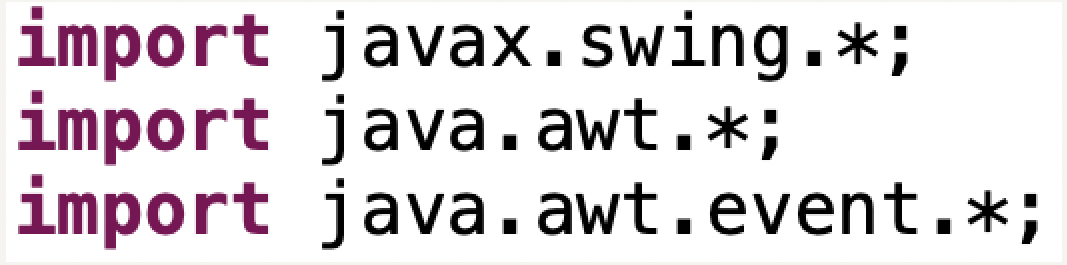
Get the selected item of one Choice component

- getSelectedItem() will give us the String data currently shown.

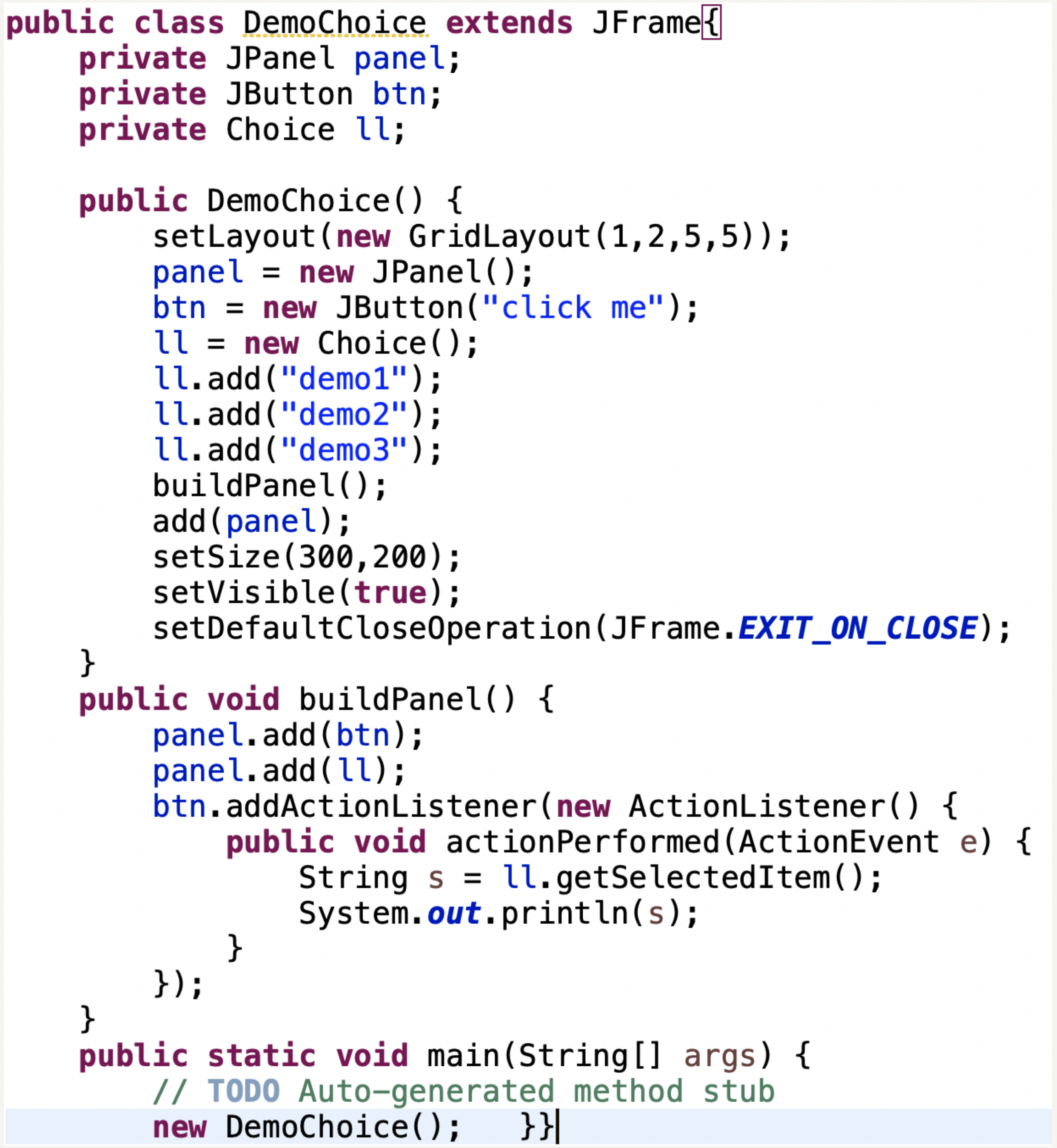


```
public void buildPanel() {
    panel.add(btn);
    panel.add(ll);
    btn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String s = ll.getSelectedItem();
            System.out.println(s);
        }
    });
}
```

The full example

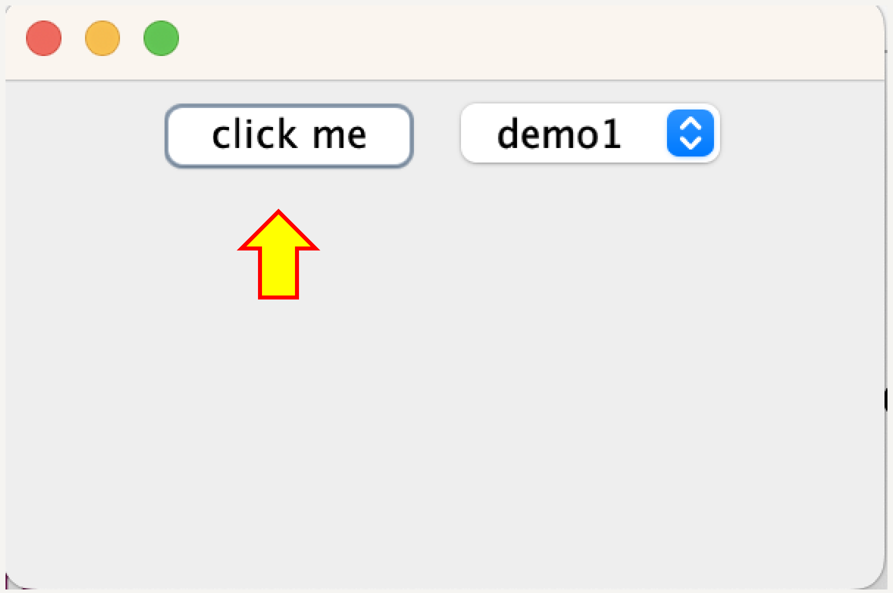


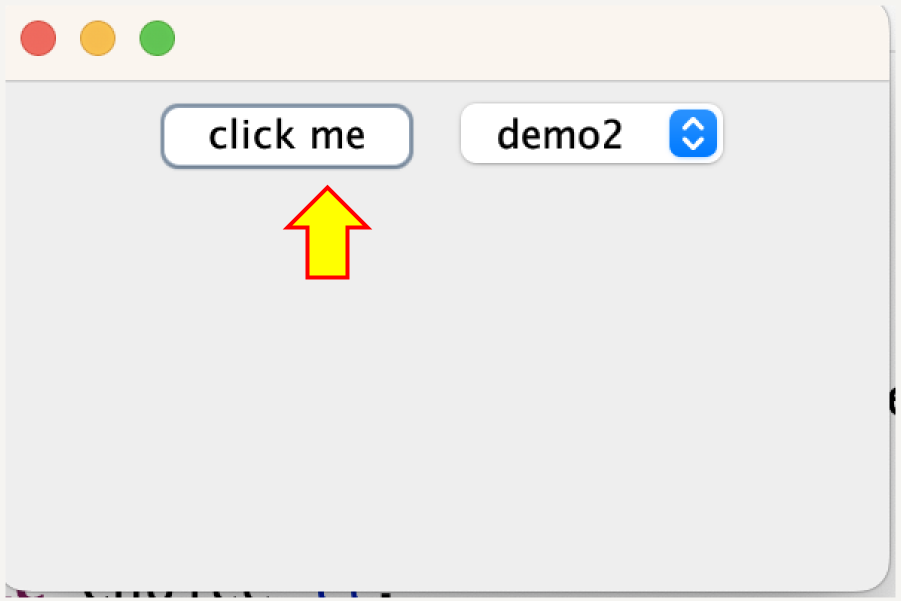
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

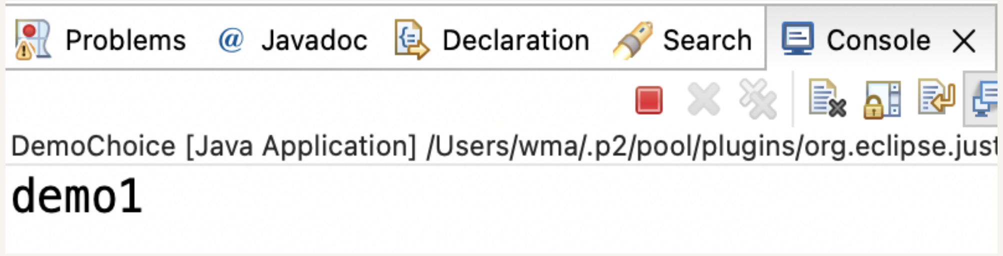


```java
public class DemoChoice extends JFrame{
    private JPanel panel;
    private JButton btn;
    private Choice ll;

    public DemoChoice() {
        setLayout(new GridLayout(1,2,5,5));
        panel = new JPanel();
        btn = new JButton("click me");
        ll = new Choice();
        ll.add("demo1");
        ll.add("demo2");
        ll.add("demo3");
        buildPanel();
        add(panel);
        setSize(300,200);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public void buildPanel() {
        panel.add(btn);
        panel.add(ll);
        btn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String s = ll.getSelectedItem();
                System.out.println(s);
            }
        });
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new DemoChoice();   }}
```

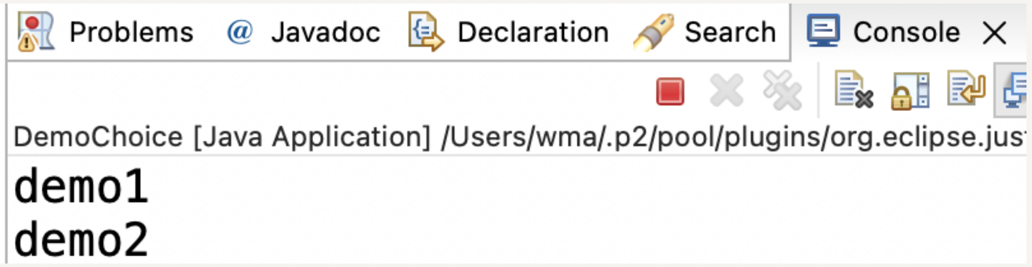
Output







```
DemoChoice [Java Application] /Users/wma/.p2/pool/plugins/org.eclipse.just
demo1
```



```
DemoChoice [Java Application] /Users/wma/.p2/pool/plugins/org.eclipse.jus
demo1
demo2
```

By clicking the button, the current shown item will be printed on Console.

End