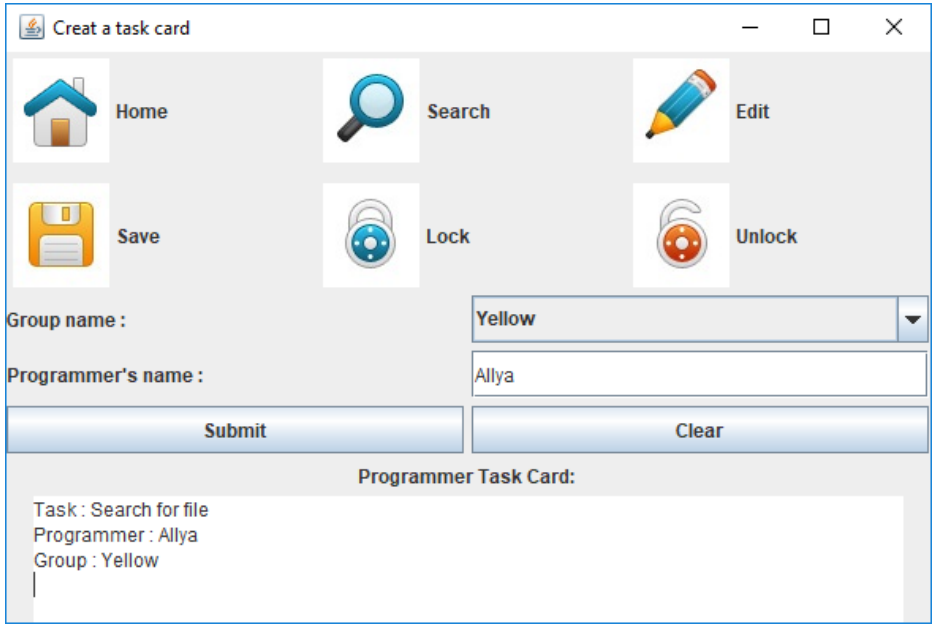
GUI Programming 101



INTRODUCTION

- There are two sets of Java APIs for graphics programming: AWT (Abstract Windowing Toolkit) and Swing.
- AWT API was introduced in JDK 1.0. Most AWT components are obsolete and should be replaced with Swing components
- Swing API, more comprehensive set of graphics libraries, was introduced as part of Java Foundation Classes (JFC) with JDK 1.2
- JFC consists of Swing, Java2D, Accessibility, Internationalization, and Pluggable Look-and-Feel Support APIs

Other Graphics APIs Support

- Eclipse's Standard Widget Toolkit (SWT) (used in Eclipse) – WindowBuilder plugin download

https://www.eclipse.org/windowbuilder/download.php

- Google Web Toolkit (GWT) (used in Android)
- 3D Graphics API such as Java bindings for OpenGL (JOGL) and Java3D

Video Links

- Installing the WindowBuilder plug-in

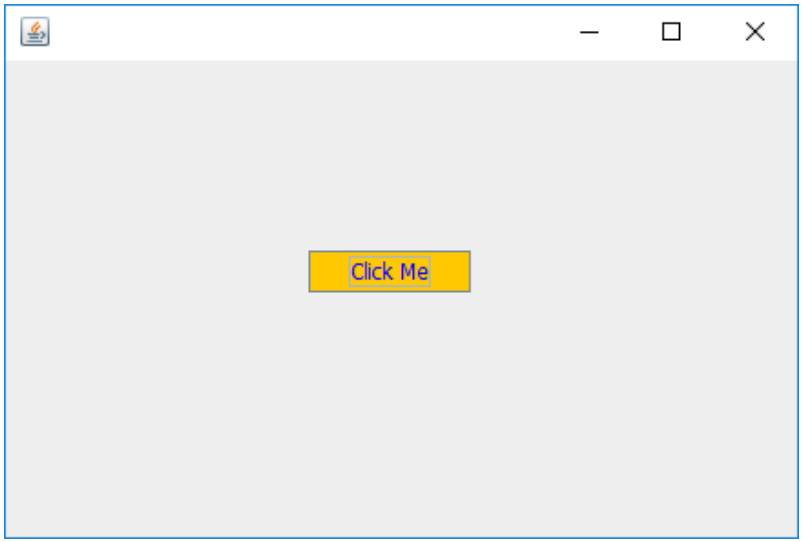
https://www.youtube.com/watch?v=oeswfZz4IW0&index=3&list=PLS1QulWo1RIbYMA5Ijb72QHaHvCrPKfS2

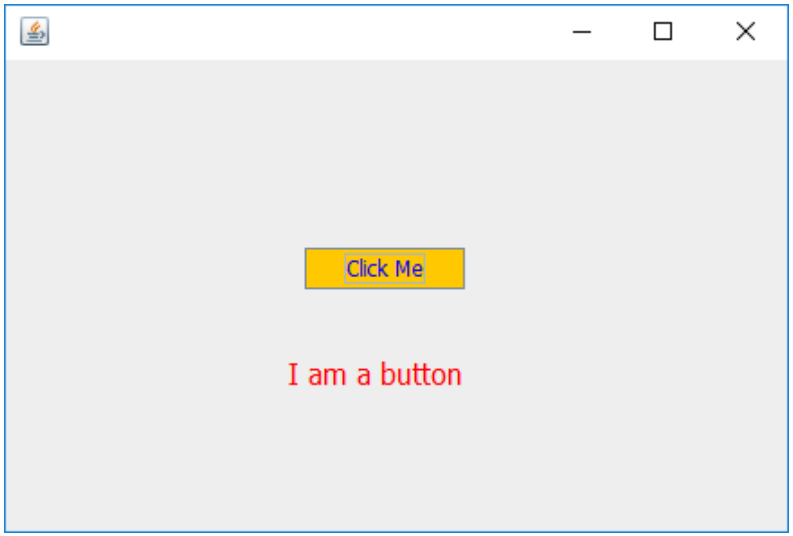
- Tutorial on creating GUI app using WindowBuilder in Eclipse

https://www.youtube.com/watch?v=r8Qiz9Bn1Ag&t=0s&index=4&list=PLS1QulWo1RIbYMA5Ijb72QHaHvCrPKfS2

First GUI App with Event Handling

- Run App





BASIC STEPS FOR BUILDING GUI APP

- Step 0: Design the GUI
- Step 1: Determine the GUI components required
  - `JButton`? `JLabel`? `JTextfield`?
- Step 2: Determine the layout of GUI components for each container
  - Which layout manager to use? `FlowLayout`? `GridLayout`? `BorderLayout`?
- Step 3: Determine how to handle events (*Event-Driven Programming*)
  - What to do when the user clicks the OK button?

Package java.awt & java.awt.event

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image

java.awt

Interfaces

ActiveEvent
Adjustable
Composite
CompositeContext
ItemSelectable
KeyEventDispatcher
KeyEventPostProcessor
LayoutManager
LayoutManager2
MenuContainer

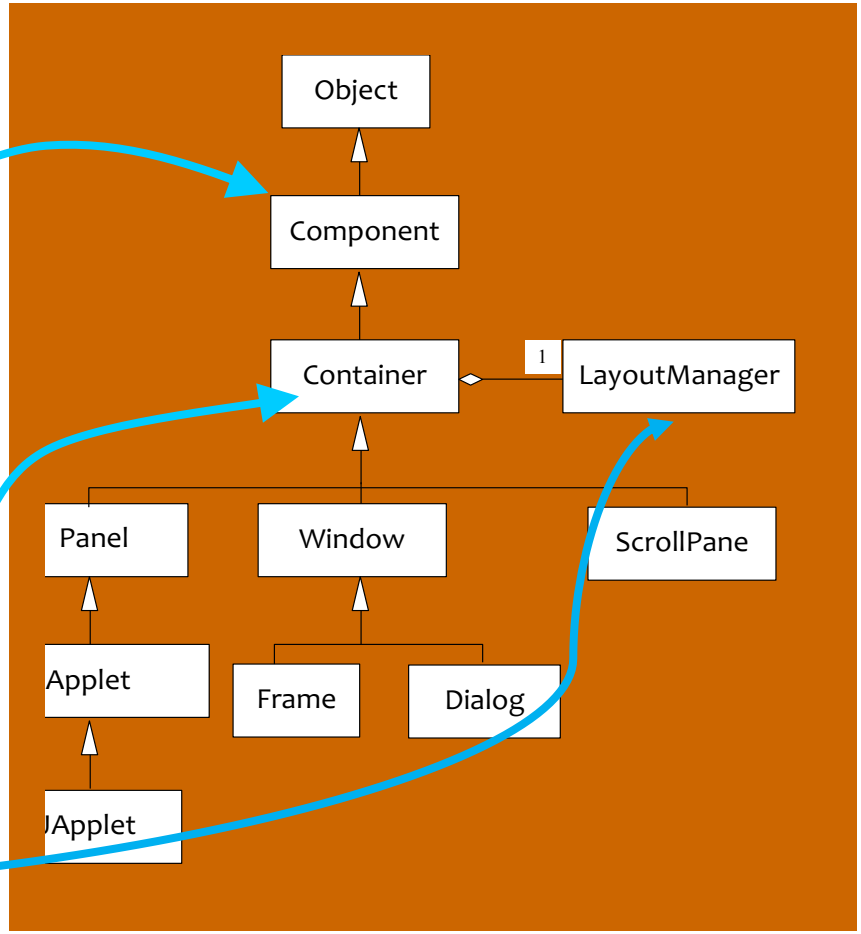
- compact1
- compact2
- compact3

Packages

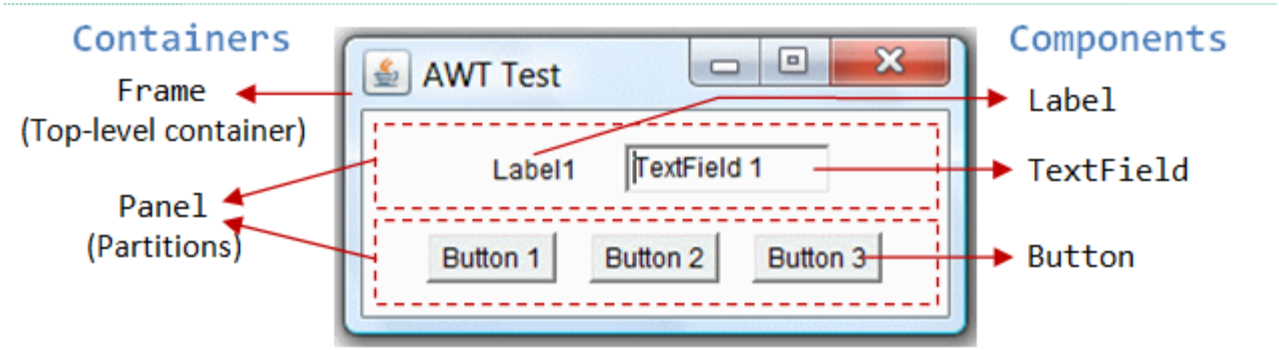
Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.

GUI COMPONENTS (in awt)

- GUI components in Java are `Component` objects.
- `Component` objects are usually put into containers (`Container` objects).
- Each `Container` object has 1 `LayoutManager`



Container and Component



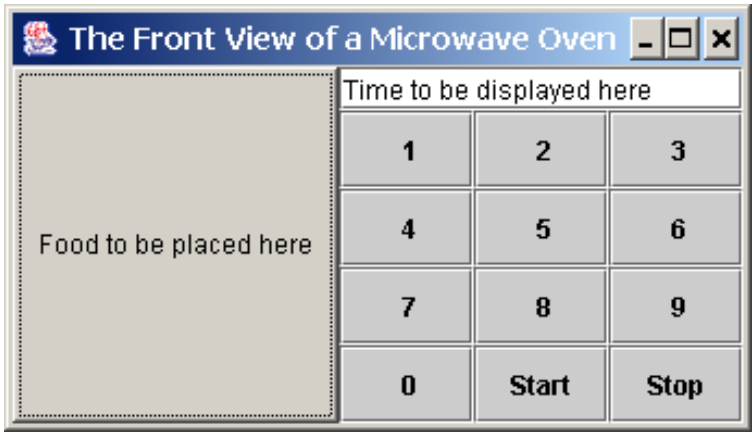
- Components are elementary GUI entities, such as Button, Label & TextField
- Containers such as Applet, Frame & Panel are used to hold components in a specific layout
- Top-level containers in AWT are Frame, Dialog and Applet
- Each container has 1 layout
- Container has method add()

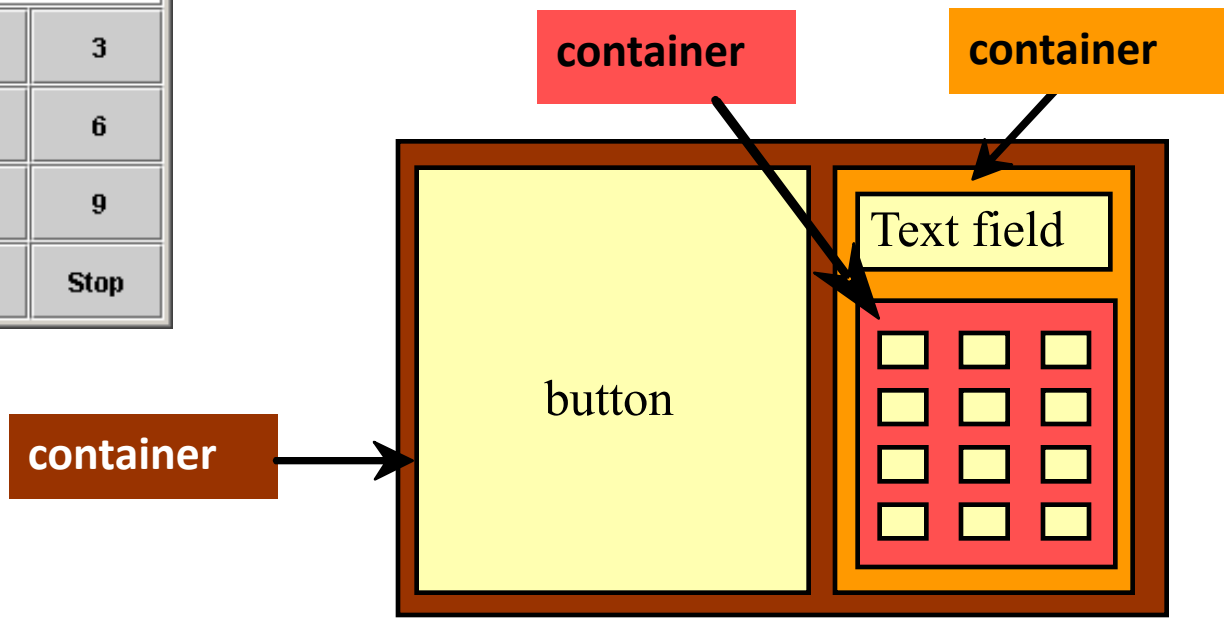
```
Panel pnl = new Panel();              // Panel is a container
Button btn = new Button("Press"); // Button is a component
pnl.add(btn);                         // The Panel container adds a Button component
```

```
import javax.swing.JFrame; // Using JFrame class in package javax.swing

// A GUI program is written as a subclass of Frame - the top-level container
// This subclass inherits all properties from Frame, e.g., title, icon, buttons, content-pane

public class MyGUIProgram extends JFrame {
  // private variables ......

  // Constructor to setup the GUI components
  public MyGUIProgram() { ...... }

  // methods ...... ......

  // The entry main() method
  public static void main(String[] args) {
    // Invoke the constructor (to setup the GUI) by allocating an instance
    new MyGUIProgram();
  }
}
```

Secondary Container

- Panel and ScrollPane are containers that can contain other containers.

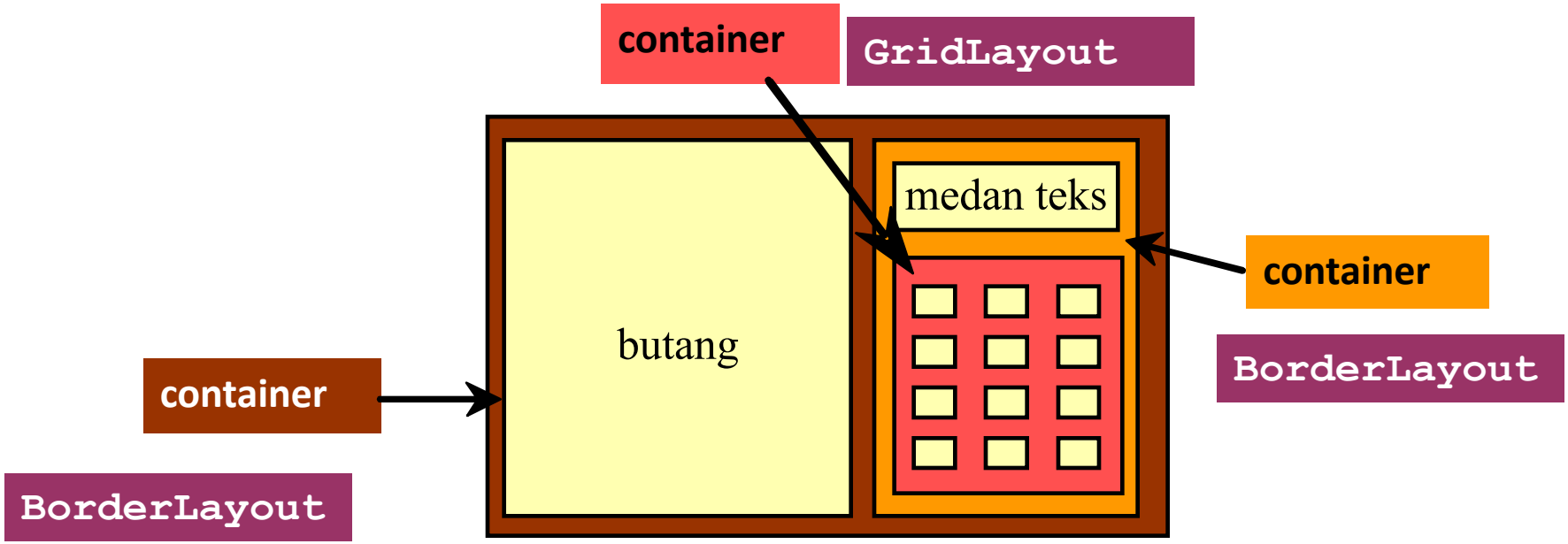




Layout Manager

- Each container (`Container` object) is associated with a layout manager (`LayoutManager` object).
- A `LayoutManager` object is responsible for determining the size of each component in a container as well as its placement.
  - If the size of a container changes, its layout manager will rearrange the components in it.

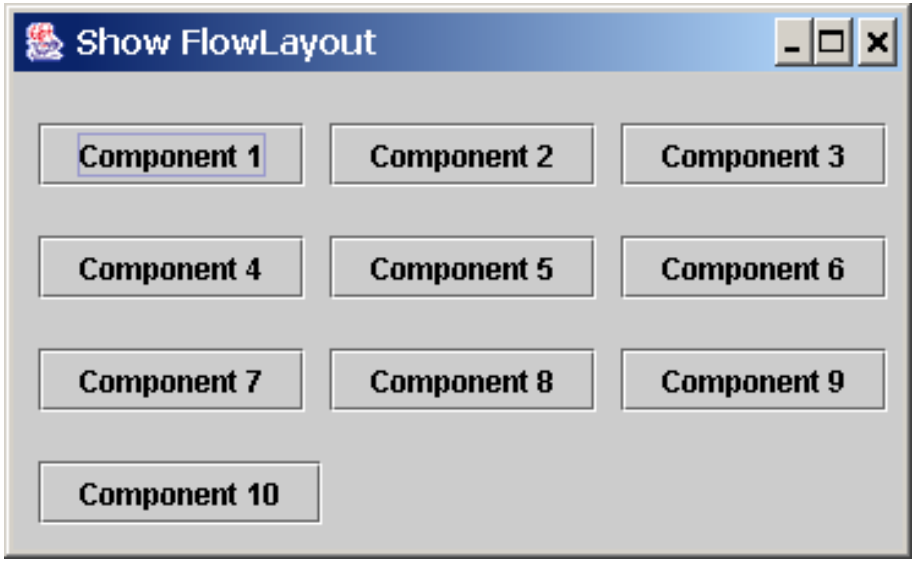
- A `LayoutManager` object is responsible for laying out components in a container.



- A number of layout managers are provided in the `java.awt` package. Some of them are listed below:
  - `BorderLayout`
  - `FlowLayout`
  - `GridLayout`
  - `CardLayout`
  - `GridBagLayout`
  - `BoxLayout`
- The following layout managers will be discussed: `FlowLayout`, `BorderLayout` and `GridLayout`.

`FlowLayout` LAYOUT MANAGER

- A `FlowLayout` object arranges components in a container from left to right, row by row from top to bottom.



- It produces a simple layout but the placement of components is difficult to control.

- `FlowLayout` is the default layout manager for `JPanel` objects.
- To create a `FlowLayout` object:

  `public FlowLayout()`

  - creates a default `FlowLayout` object (components will be center-aligned)

  `public FlowLayout(int a)`

  `public FlowLayout(int a, int hg, int vg)`

where

`a`: alignment

possible values:

```
FlowLayout.LEFT
FlowLayout.RIGHT
FlowLayout.CENTER
```

`hg`: horizontal gap between components

`vg`: vertical gap between components

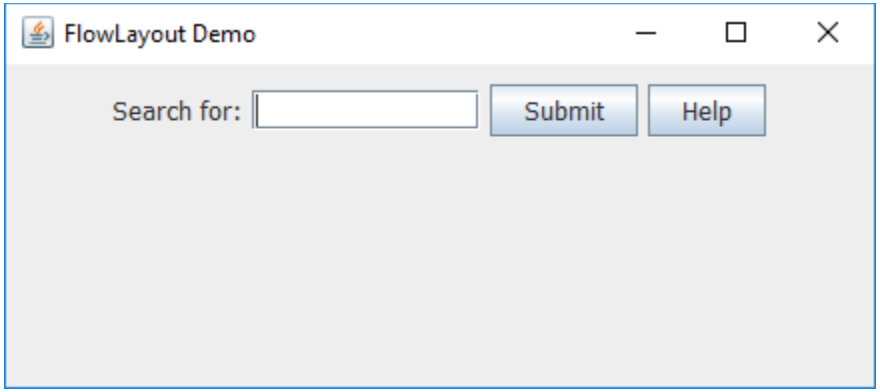
FlowLayout Demo

```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import java.awt.FlowLayout;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.Font;

public class MyGUIFrame extends JFrame {

    private JPanel contentPane;
    private JTextField textField;
    private JButton btnSubmit, btnHelp;
    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    MyGUIFrame frame = new MyGUIFrame();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

FlowLayout.CENTER DEMO

```
/**
 * Create the frame.
 */
public MyGUIFrame() {
    setTitle("FlowLayout Demo");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 200);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

    JLabel lblSearchFor = new JLabel("Search for:");
    lblSearchFor.setFont(new Font("Tahoma", Font.PLAIN, 13));
    contentPane.add(lblSearchFor);

    textField = new JTextField();
    contentPane.add(textField);
    textField.setColumns(10);

    btnSubmit = new JButton("Submit");
    btnSubmit.setFont(new Font("Tahoma", Font.PLAIN, 13));
    contentPane.add(btnSubmit);

    btnHelp = new JButton("Help");
    btnHelp.setFont(new Font("Tahoma", Font.PLAIN, 13));
    contentPane.add(btnHelp);
}
```

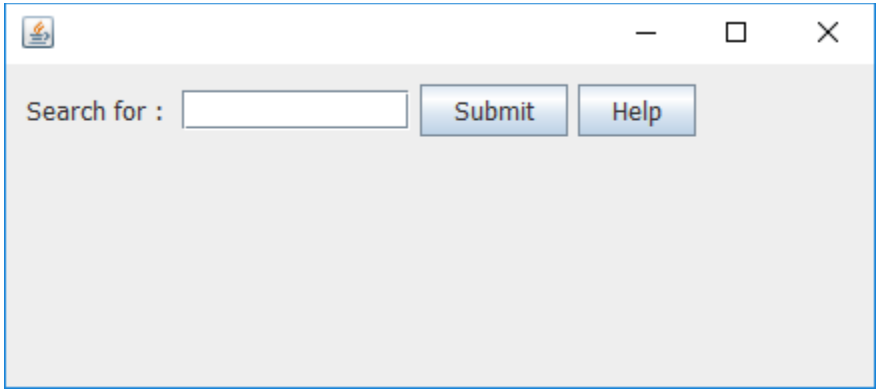
- *FlowLayout Demo* user interface:



FlowLayout.LEFT DEMO

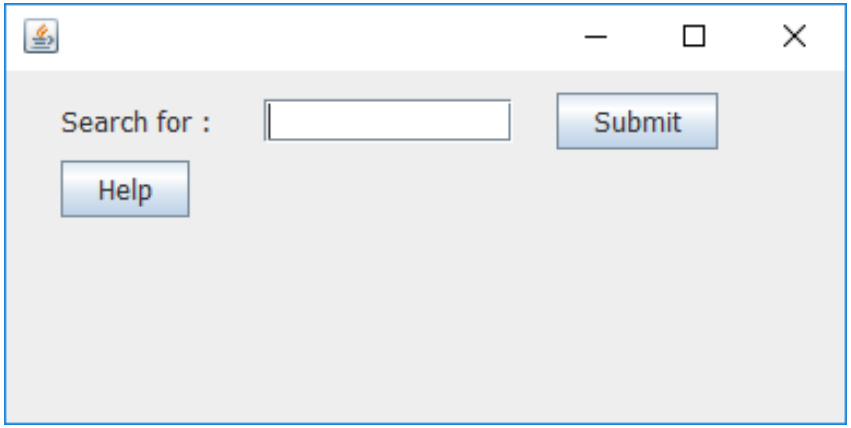
```
/**
 * Create the frame.
 */
public FlowLayoutLeftDemo() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 200);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 5));

    JLabel lblSearchFor = new JLabel("Search for : ");
    lblSearchFor.setFont(new Font("Tahoma", Font.PLAIN, 13));
    contentPane.add(lblSearchFor);

    textField = new JTextField();
    contentPane.add(textField);
    textField.setColumns(10);

    JButton btnSubmit = new JButton("Submit");
    btnSubmit.setFont(new Font("Tahoma", Font.PLAIN, 13));
    contentPane.add(btnSubmit);

    JButton btnHelp = new JButton("Help");
    btnHelp.setFont(new Font("Tahoma", Font.PLAIN, 13));
    contentPane.add(btnHelp);
}
```

- *FlowLayoutLeftDemo* user interface:



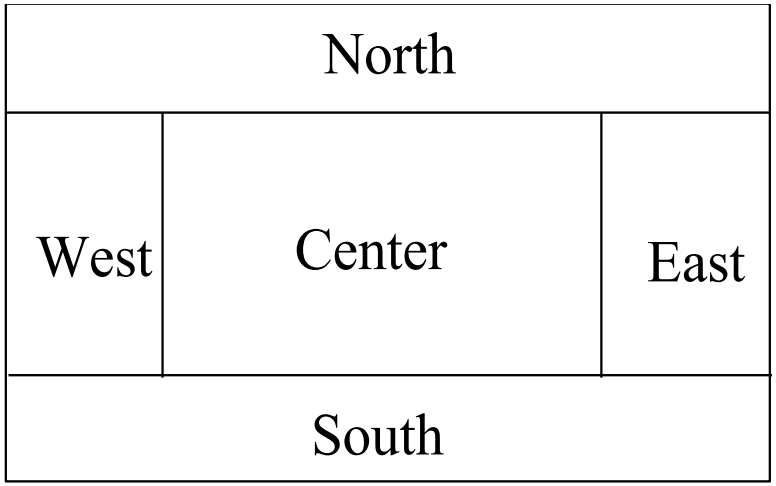
FlowLayoutDemo3 DEMO

```
contentPane.setLayout(new FlowLayout(FlowLayout.LEFT,
20, 5));
```



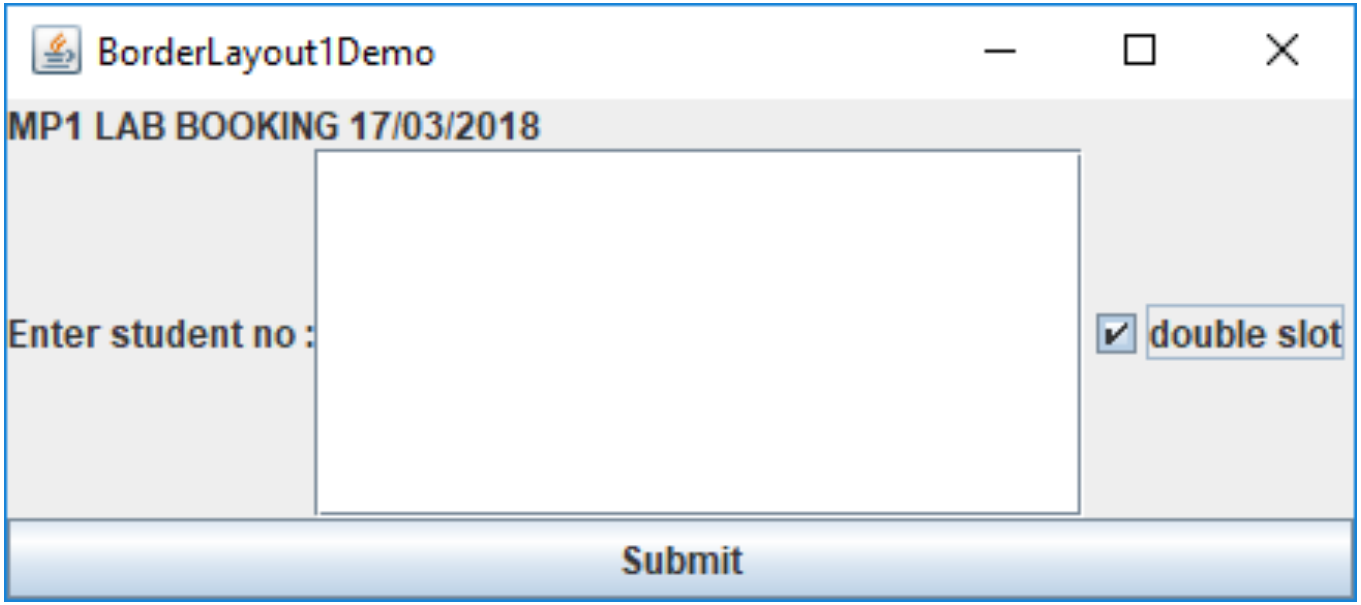
`BorderLayout` LAYOUT MANAGER

- A `BorderLayout` object arranges components according to regions.
- A container is divided into five regions: *North*, *South*, *East*, *West* and *Center*.



- A `BorderLayout` object places <u>at most one</u> component in a region.
- The size of a component will be adapted so that it fills the whole region which contains it.

- *BorderLayoutDemo1* user interface:



- `BorderLayout` is the default layout manager associated with containers of `JApplet` objects.

  Example:

  ```
  Container pane =
  getContentPane();
  ```

  The container returned by `getContentPane()` has a `BorderLayout` layout manager.

- To create a `BorderLayout` object:

  ```
  public BorderLayout()
  ```

- To add a component to a container with a `BorderLayout` layout manager :

  `add(Component comp, Object cnst)`

  where

  `comp` : component to be added

  `cnst` : specifies where the component should be placed

  Possible values:

  `BorderLayout.NORTH`

  `BorderLayout.SOUTH`

  `BorderLayout.EAST`

  `BorderLayout.WEST`

  `BorderLayout.CENTER`

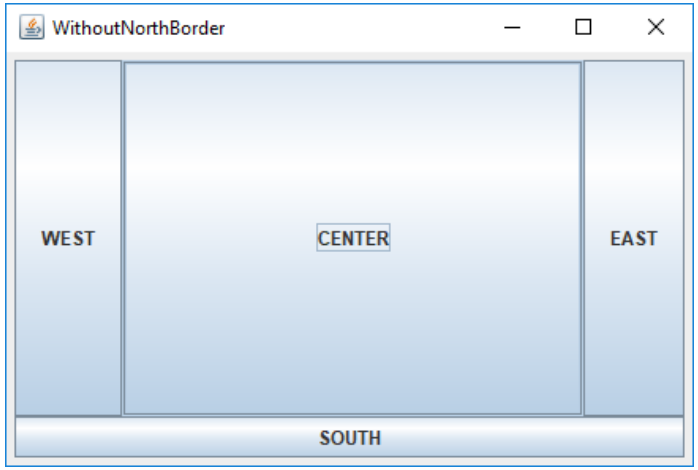
- Examples:

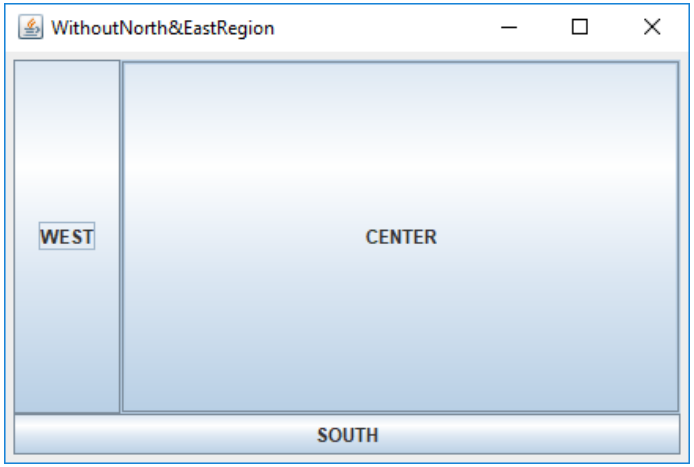
```
pane.add(new JButton("OK"),
         BorderLayout.SOUTH);

pane.add(new
JLabel("LABTIMES"),
BorderLayout.NORTH);
```

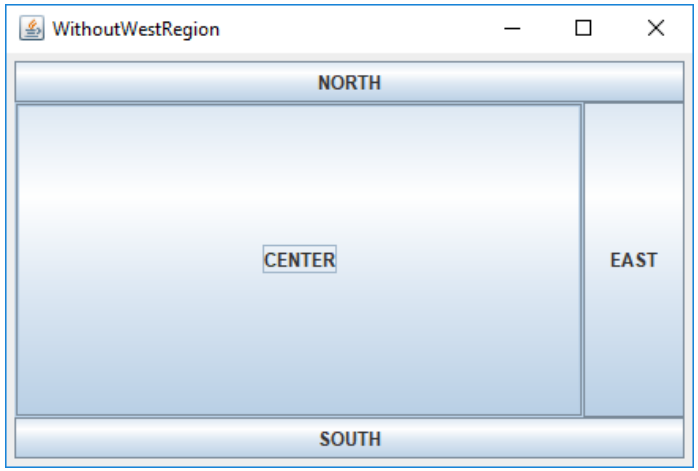
BorderLayoutDemo1 DEMO

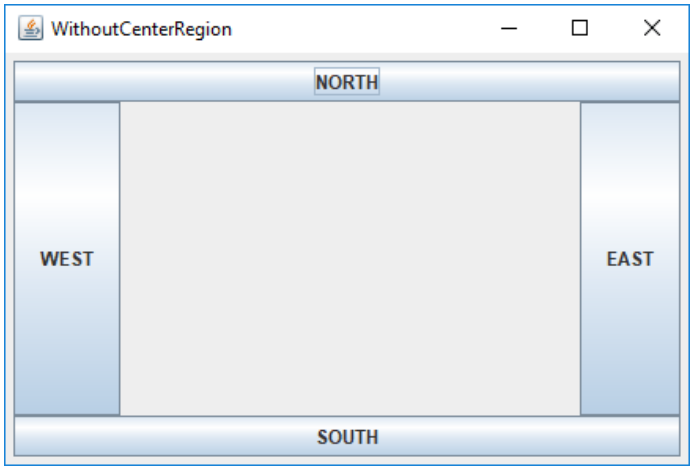
```
public BorderLayout1Demo() {
    setTitle("BorderLayout1Demo");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 200);
    getContentPane().setLayout(new BorderLayout(0, 0));

    JLabel lblLabBooking = new JLabel("MP1 LAB BOOKING 17/03/2018");
    getContentPane().add(lblLabBooking, BorderLayout.NORTH);

    JLabel lblEnterStudentNo = new JLabel("Enter student no :");
    getContentPane().add(lblEnterStudentNo, BorderLayout.WEST);

    textField = new JTextField();
    getContentPane().add(textField, BorderLayout.CENTER);
    textField.setColumns(10);

    JCheckBox chckbxDoubleSlot = new JCheckBox("double slot");
    getContentPane().add(chckbxDoubleSlot, BorderLayout.EAST);

    JButton btnSubmit = new JButton("Submit");
    getContentPane().add(btnSubmit, BorderLayout.SOUTH);

}
```

- Note that if there are empty regions, the layout manager will try to adapt the size of components in other regions to fill up those empty regions.



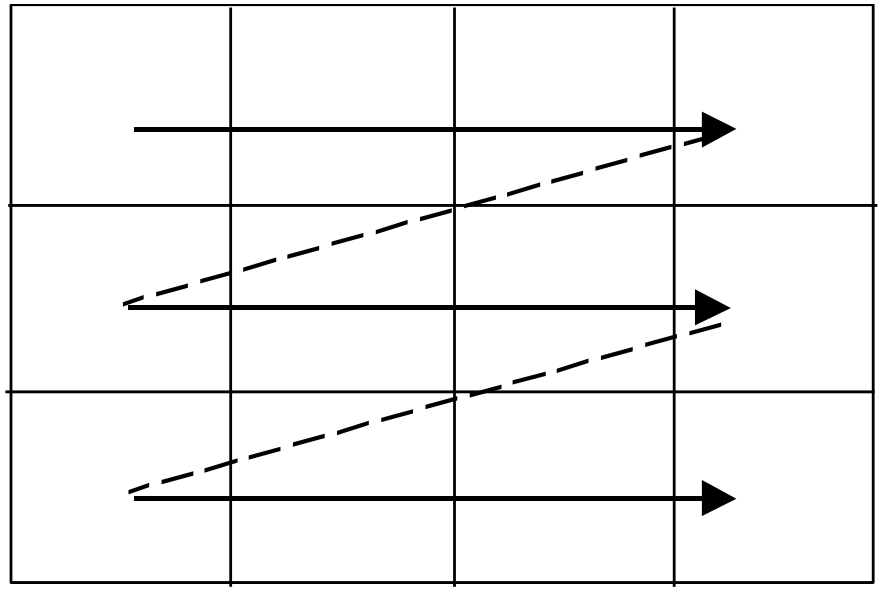






`GridLayout` LAYOUT MANAGER

- A `GridLayout` object arranges components in a grid of cells.
- Components are added to a container by placing them in cells, from left to right, row by row, from top to bottom.



- To create a `GridLayout` object:

```
public GridLayout(int nr, int nc)
public GridLayout(int nr, int nc,
                  int hg, int vg)
```

where

`nr`: number of rows

`nc`: number of columns

`hg`: horizontal gap between components

`vg`: vertical gap between components

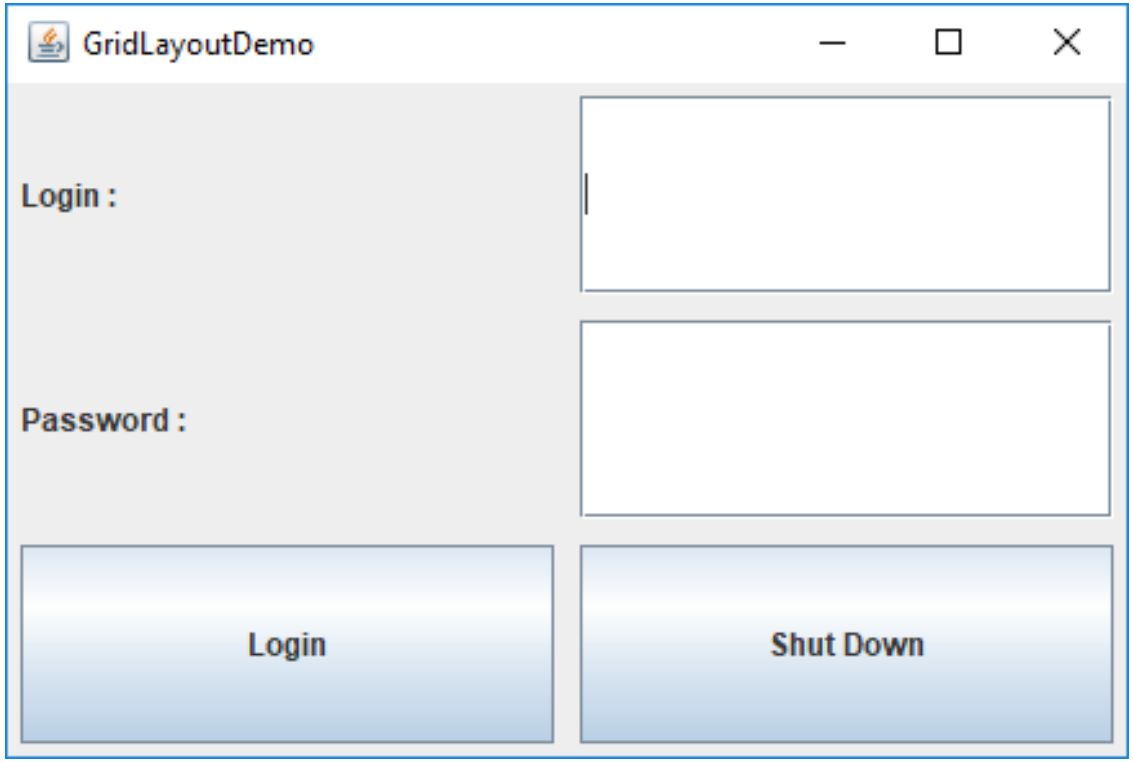
- Examples:

```
pane.setLayout(new GridLayout(5,5));

pane.setLayout(new GridLayout(5,5,10,10));
```

GridLayoutDemo DEMO

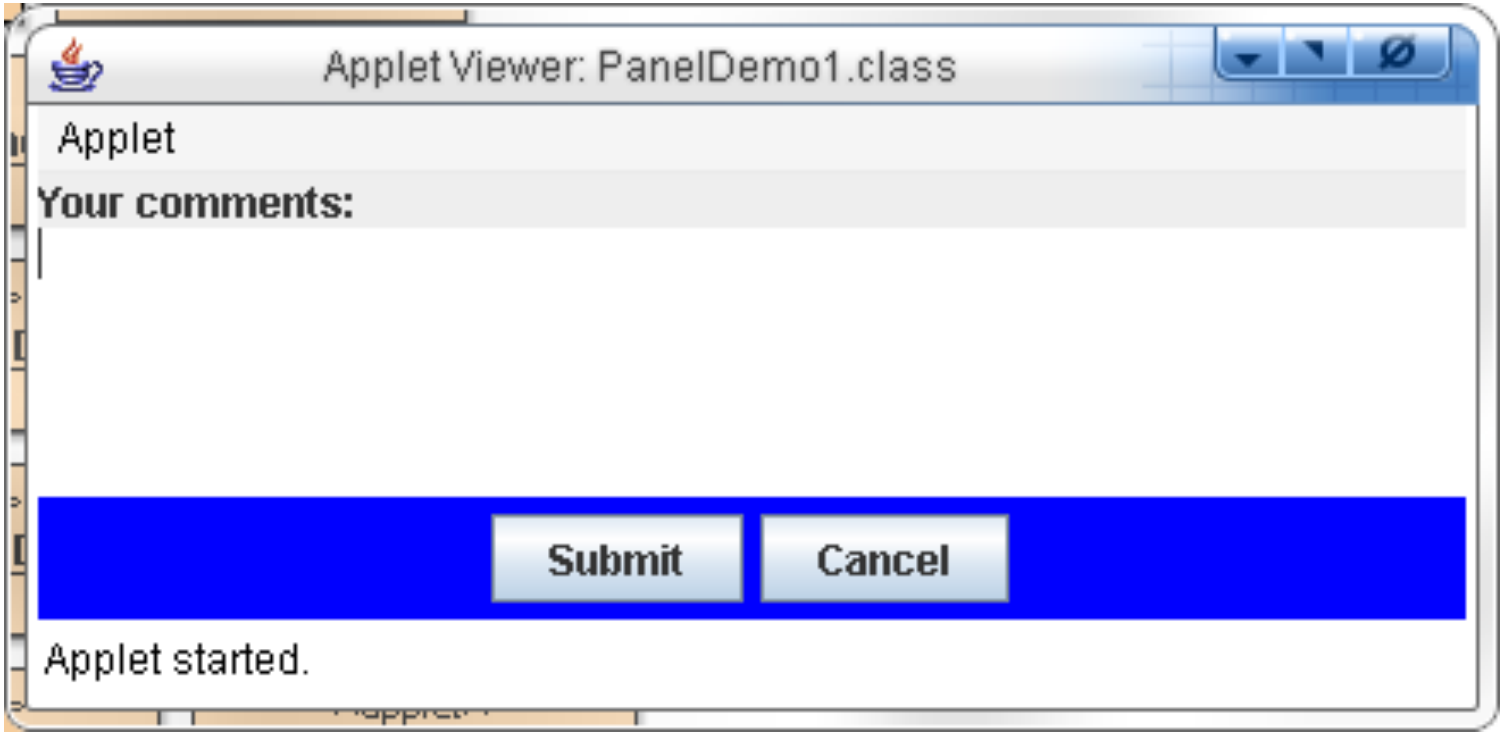
```
public GridLayoutDemo() {
    setTitle("GridLayoutDemo");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new GridLayout(3, 2, 10, 10));

    JLabel lblLogin = new JLabel("Login :");
    contentPane.add(lblLogin);

    textField = new JTextField();
    contentPane.add(textField);
    textField.setColumns(10);

    JLabel lblNewLabel = new JLabel("Password :");
    contentPane.add(lblNewLabel);

    textField_1 = new JTextField();
    contentPane.add(textField_1);
    textField_1.setColumns(10);

    JButton btnNewButton = new JButton("Login");
    contentPane.add(btnNewButton);

    JButton btnShutDown = new JButton("Shut Down");
    contentPane.add(btnShutDown);
```

- *GridLayoutDemo*'s user interface:



PanelDemo1 DEMO

- *PanelDemo1*'s user interface:



USING PANELS TO GROUP COMPONENTS

- Panels can be used to group components.
- When a layout manager arranges components in a container, panels of components are treated as single components.
- Panels can be represented as `JPanel` objects.

- To create a `JPanel` object:

```
public JPanel()
```

- To add a component to a JPanel object, use the `add()` method.

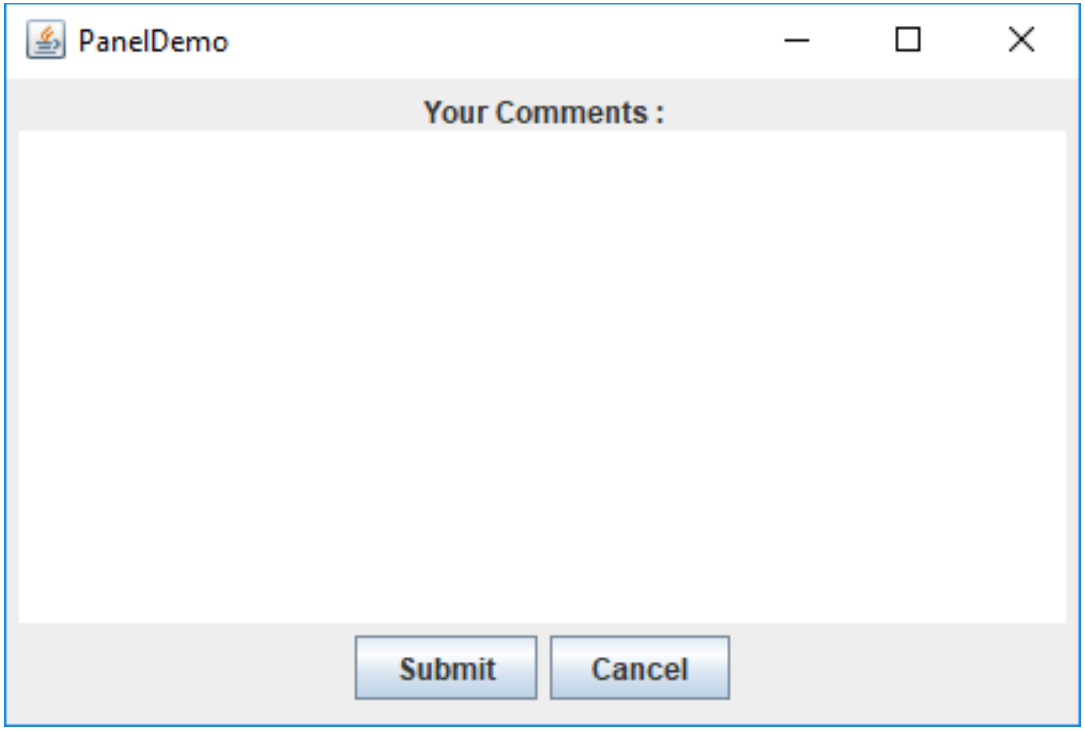
Example 1:

```
JPanel panel = new JPanel();
panel.add(new JButton("OK"));
```

Example 2:

```
JPanel panel = new JPanel();
panel.setLayout(new BorderLayout());
panel.add(new JButton("OK"),
                BorderLayout.SOUTH);
```

Panel Demo

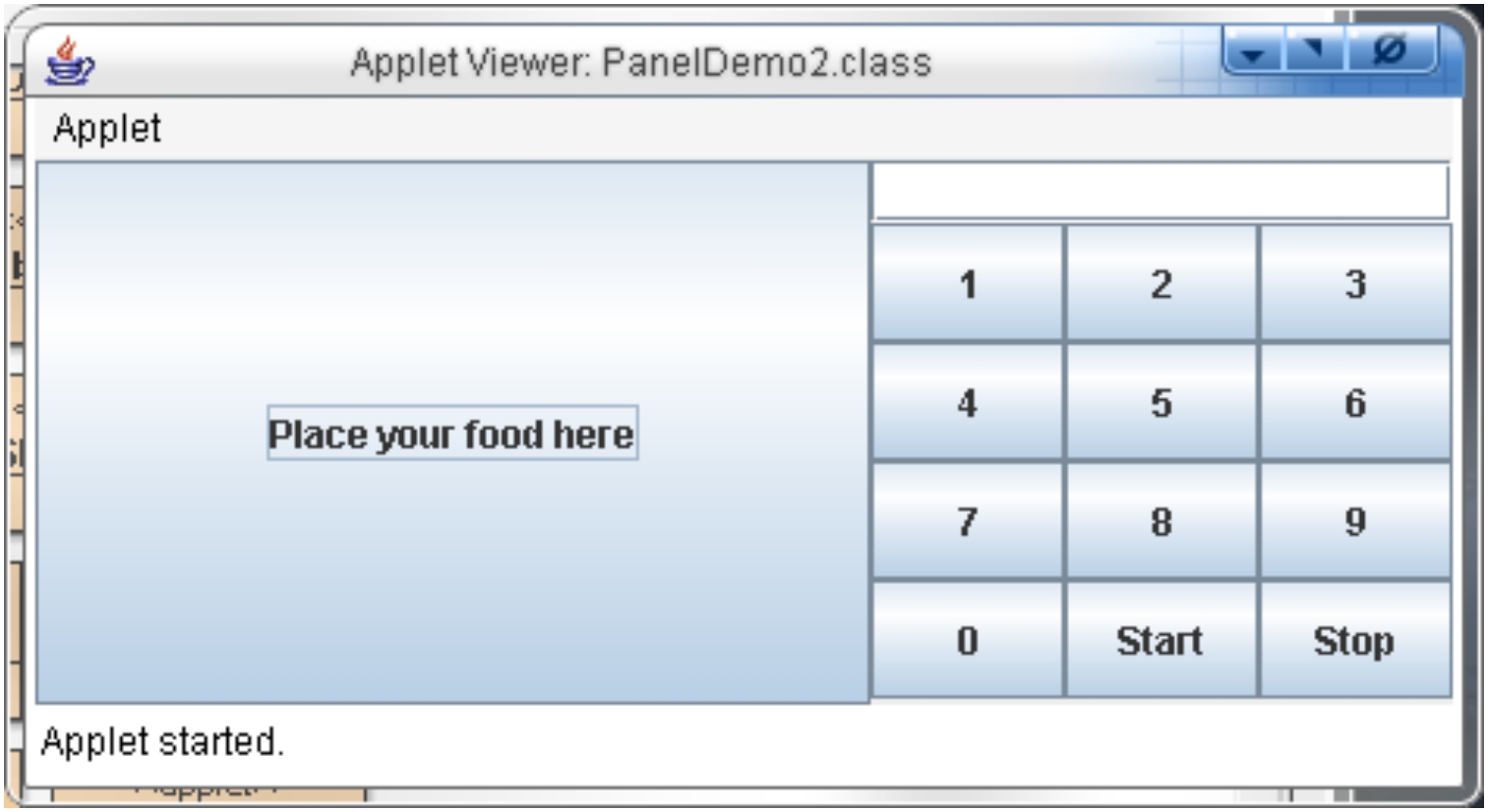


PanelDemo1 DEMO

```java
/**
 * Create the frame.
 */
public PanelDemo() {
    setTitle("PanelDemo");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.SOUTH);

    JButton btnSubmit = new JButton("Submit");
    panel.add(btnSubmit);

    JButton btnCancel = new JButton("Cancel");
    panel.add(btnCancel);

    JLabel lblYourComments = new JLabel("Your Comments :");
    lblYourComments.setHorizontalAlignment(SwingConstants.CENTER);
    lblYourComments.setAlignmentX(CENTER_ALIGNMENT);
    contentPane.add(lblYourComments, BorderLayout.NORTH);

    JTextArea textArea = new JTextArea();
    contentPane.add(textArea, BorderLayout.CENTER);
}
```

EXERCISE

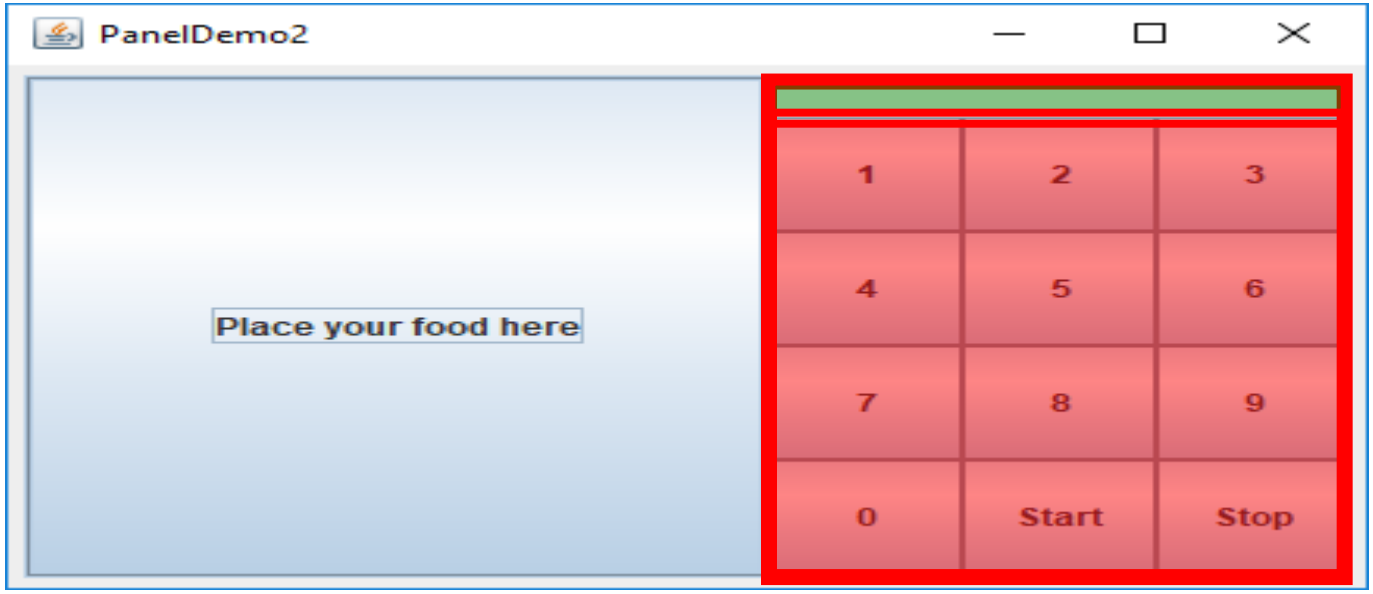
- Write Java code to produce the following user interface:



PanelDemo2 DEMO

```
public PanelDemo2() {
    setTitle("PanelDemo2");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);

    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());
    JPanel keypad = new JPanel();
    keypad.setLayout(new GridLayout(4, 3));
    keypad.add(new JButton("1"));
    keypad.add(new JButton("2"));
    keypad.add(new JButton("3"));
    keypad.add(new JButton("4"));
    keypad.add(new JButton("5"));
    keypad.add(new JButton("6"));
    keypad.add(new JButton("7"));
    keypad.add(new JButton("8"));
    keypad.add(new JButton("9"));
    keypad.add(new JButton("0"));
    keypad.add(new JButton("Start"));
    keypad.add(new JButton("Stop"));
    panel.add(new JTextField(10), BorderLayout.NORTH);
    panel.add(keypad, BorderLayout.CENTER);
    contentPane.add(new JButton("Place your food here"), BorderLayout.CENTER);
    contentPane.add(panel, BorderLayout.EAST);
```


PanelDemo2
Place your food here
1
2
3
4
5
6
7
8
9
0
Start
Stop


PanelDemo2
Place your food here
1
2
3
4
5
6
7
8
9
0
Start
Stop


PanelDemo2
Place your food here
1
2
3
4
5
6
7
8
9
0
Start
Stop

Using WindowBuilder in Eclipse