CHAPTER 07: Graphical User Interface in Java

By: Muleta T.

E-mail: muleta.taye@aastu.edu.et

May 14, 2023

CH07: Topics to be covered

CH-07: GUI

- GUI (pronounced "GOO-ee")

- GUI (pronounced "GOO-ee")
  - presents a user-friendly mechanism for interacting with an application.
  - built from small widgets called GUI Components.

    ↓

    java object with which the user interacts via any input devices
  - Three sets of Java APIs for graphics programming:
    1. AWT (Abstract Windowing Toolkit) ↦ java.awt package
       - AWT UI components have become obsolete and replaced by newer Swing UI components.
    2. Swing ↦ javax.swing package
       - more comprehensive set of graphics libraries that enhances the AWT
    3. JavaFX ↦ javafx.uti package
       - modern and adavanced graphical programming API

Abstract window toolkit (AWT)

- The java.awt package contains the core AWT graphics classes:
  - GUI Component classes $\mapsto$ Button, TextField, and Label.
  - GUI Container classes $\mapsto$ Frame and Panel.
  - Layout managers $\mapsto$ FlowLayout, BorderLayout and GridLayout.
  - Custom classes $\mapsto$ Graphics, Color and Font.
- How about Events ?
  - Recap "what is Events are?"
  - java.awt.event package supports event handling:

    $\downarrow$

    - Event classes $\mapsto$ ActionEvent, MouseEvent, KeyEvent and WindowEvent,
    - Event Listener Interfaces $\mapsto$ ActionListener, MouseListener, MouseMotionListener, KeyListener and WindowListener
    - Event Listener Adapter classes $\mapsto$ MouseAdapter, KeyAdapter and WindowAdapter.

- Any UI has 3 entities:
  1. UI elements
     - Core visual element visible to the user + used for interacting with the application.
  2. Layouts
     - How to organize UI element in the screen.
  3. Behavior
     - the events which should occur when a user interacts with UI elements.

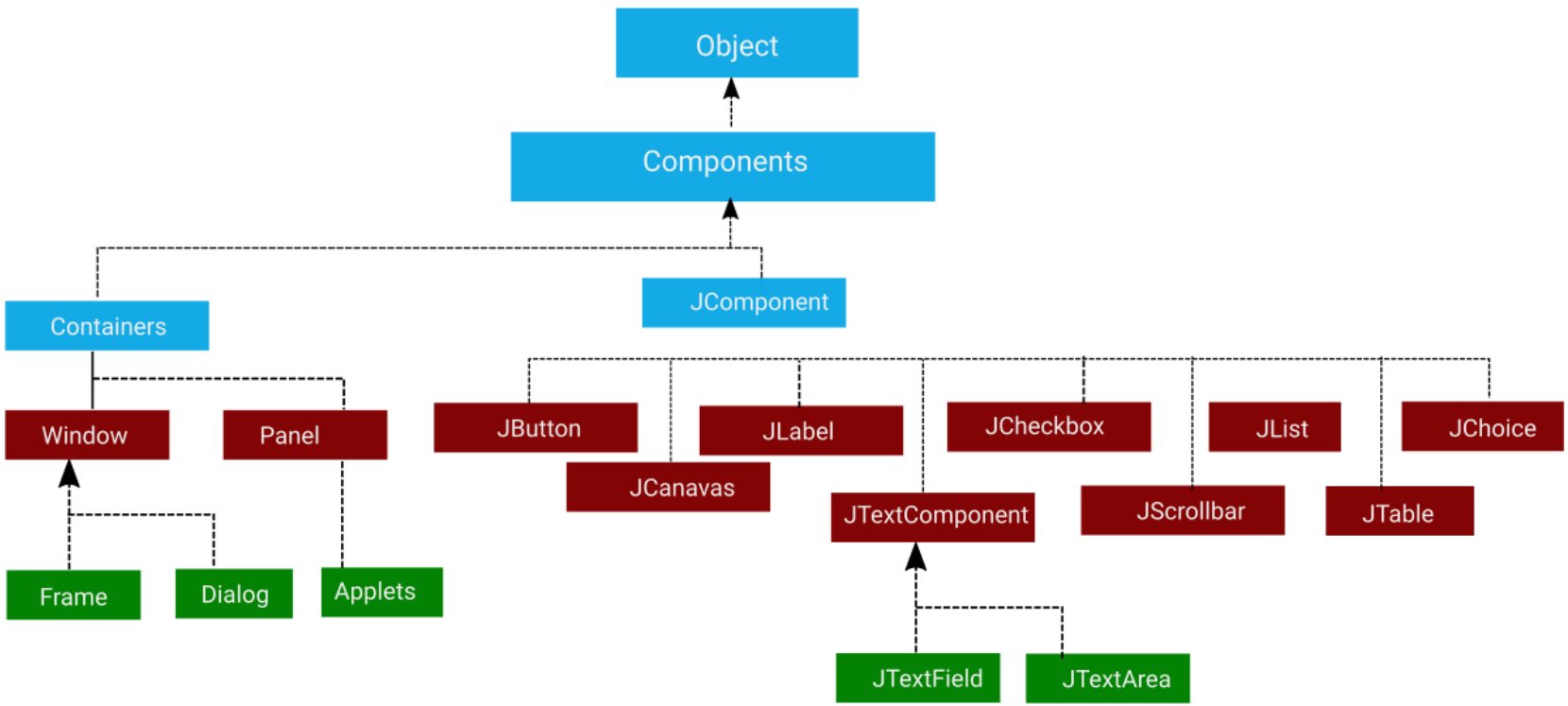


Figure 1: Hierarchy Of Swing API

UI Components

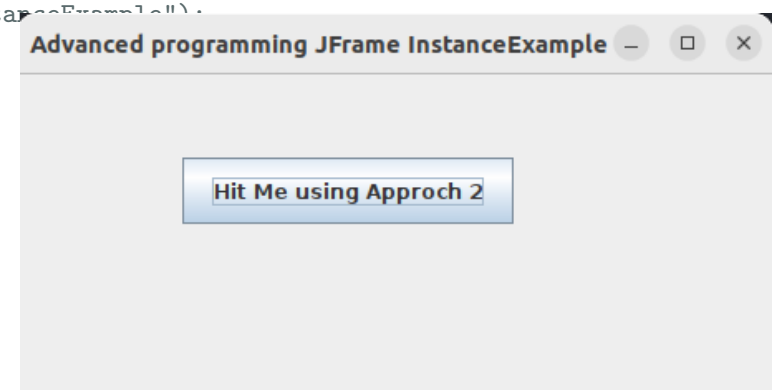
- Containers
  - Container in Java Swing is a component that is used to hold other components such as text fields, buttons, etc.
  - It is a subclass of javax.swing.JComponent
  - 4 types:
    - **Window**: It is an instance of the Window class having neither border nor title
    - **Frame:** Frame – A fully functioning window with icons and titles
    - **Dialog:** a subclass of Window and comes with the border as well as the title.
    - **Panel** : the concrete subclass of Container and doesn't contain any title bar, menu bar or border.
      - Panel class is a generic container for holding the GUI components.

- JFrame : is like the main window of the GUI application using swing.
- 2 different ways
  - By Extending The JFrame Class

```
import javax.swing.*;
class JframeDemoApprochOne extends JFrame{

    JframeDemoApprochOne(){
        //create button object
        JButton button=new JButton("Hit Me!");
        button.setBounds(100,50,150, 40);

        add(button);//add button on frame
        setSize(300,200);
        setLayout(null);
        setVisible(true);
    }
}
public class Main {
    public static void main(String[] args) {
      new JframeDemoApprochOne(); //create an object of FrameInherited class
    }
}
```

- 2 different ways ...
  - By Instantiating The JFrame Class

```
import javax.swing.*;
public class Main {

    public static void main(String[] args) {
        //create a JFrame object
        JFrame jframe= new JFrame("Advanced programming JFrame InstanceExample");

        //create instance of JButton
        JButton btn=new JButton("Hit Me using Approch 2");

        //dimensions of JButton object
        btn.setBounds(100,50,200, 40);

        jframe.add(btn);//add button in JFrame

        //set frame width = 300 and height = 200
        jframe.setSize(300,200);
        jframe.setLayout(null);
        jframe.setVisible(true);//make the frame visible
    }
}
```



- JPanel
  - is a component that is contained inside a frame window
  - panels are primarily used to partition the frame
  - panel groups several other components inside it
  - JPanel class inherits from JComponent and has FlowLayout as its default layout.

```
 import javax.swing.*;
public class Main {
    public static void main(String args[]) {
        JFrame jframe = new JFrame("Advanced Programming GUI");
        JPanel panelInFrame = new JPanel();
        panelInFrame.setBounds(50, 85, 100, 100);

        JButton button = new JButton("Button in JPanel");
        panelInFrame.add(button);

        JLabel text = new JLabel("Label");
        panelInFrame.add(text);

        jframe.add(panelInFrame);
        jframe.setSize(500, 150);
        jframe.setVisible(true);
        jframe.setLayout(null);

    }
}
```

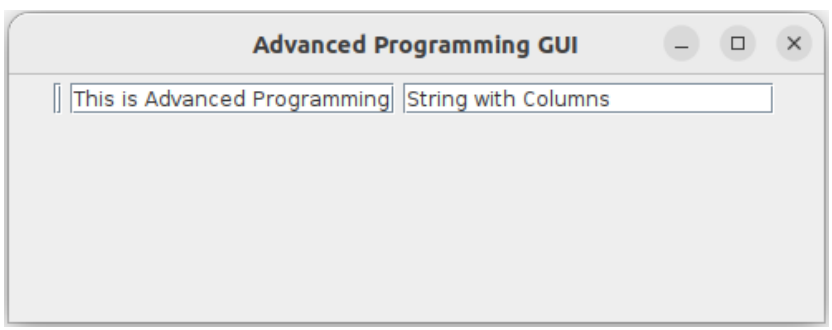
- JPanel
  - is a component that is contained inside a frame window
  - panels are primarily used to partition the frame
  - panel groups several other components inside it
  - JPanel class inherits from JComponent and has FlowLayout as its default layout.

```
 import javax.swing.*;
public class Main {
    public static void main(String args[]) {
        JFrame jframe = new JFrame("Advanced Programming GUI");
        JPanel panelInFrame = new JPanel();
        panelInFrame.setBounds(50, 85, 100, 100);

        JButton button = new JButton("Button in JPanel");
        panelInFrame.add(button);

        JLabel text = new JLabel("Label");
        panelInFrame.add(text);

        jframe.add(panelInFrame);
        jframe.setSize(500, 150);
        jframe.setVisible(true);
        jframe.setLayout(null);

    }
}
```

- Output:

- JTextField Class
  - It inherits the JTextComponent class and allow as to edit of single line text.
  - has 4 Constructors:

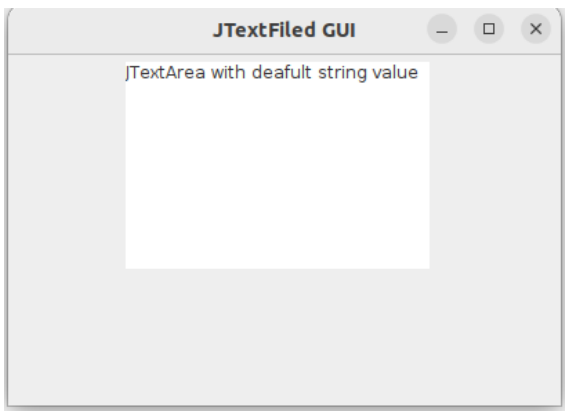
- JTextField() ↦ Creates a new TextField
- JTextField(String text) ↦ Creates a new TextField initialized with the specified text.
- JTextField(String text, int columns) ↦ Creates a new TextField initialized with the specified text and columns.
- JTextField(int columns) ↦ Creates a new empty TextField with the specified number of columns.




```
import javax.swing.*;
public class Main {
   public static void main(String args[]) {

        JFrame jframe = new JFrame("JTextFiled GUI");

        JTextField text = new JTextField();
text.setBounds(50, 100, 200, 30);
        JTextField textWithString = new JTextField(
                "This is Advanced Programming");
       textWithString.setBounds(50, 150, 200, 30);
        JTextField textCol = new JTextField("
                String with Columns", 20);



        jframe.add(text);
        jframe.add(textWithString);
        jframe.add9(textCol);
        jframe.setSize(500, 150);
        jframe.setVisible(true);
        jframe.setLayout(null);

    }
};
```

- JTextArea Class
  - an editable text field. It can have multiple lines.
  - has 4 Constructors:

- JTextArea () ↦ Default constructor. Create an empty text area.
- JTextArea (String text) ↦ Creates a textarea with s as the default value.
- JTextArea (int row, int column) ↦Creates a text area with a specified row x column.
- JTextArea (String text, int row, int column): ↦ Creates a text are2a with specified row x column and default value text.



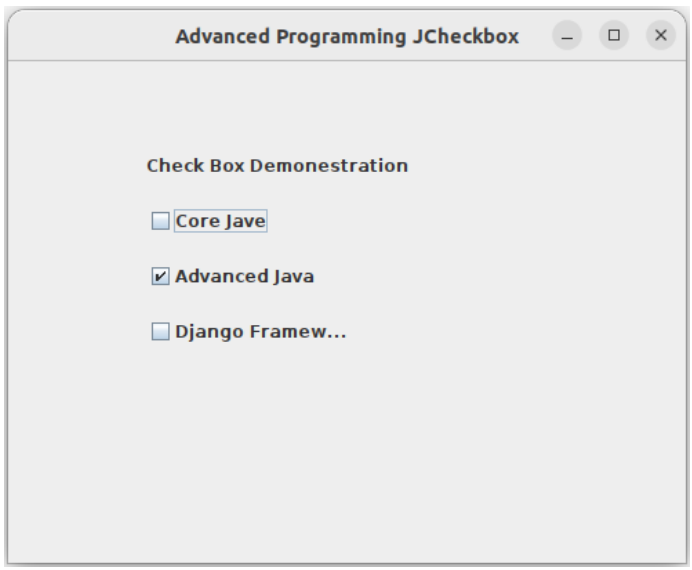

```
import javax.swing.*;

public class JTextAreaDemonestration {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("JTextFiled GUI");
        JPanel jpanel = new JPanel();
        jpanel.setBounds(10, 200, 250, 100);

        JTextArea txtarea = new JTextArea(
            "JTextArea with deafult string value", 10, 20
        );
        txtarea.setBounds(0, 30, 200, 200);
        jpanel.add(txtarea);

        jframe.add(jpanel);

        jframe.setSize(400, 400);
        jframe.setVisible(true);
        jframe.setLayout(null);

    }
};
```

- JCheckBox class
  - The JCheckBox class is used to create a checkbox.
  - It is used to turn an option on (true) or off (false).

Methods	Description
JCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a string representation of this JCheckBox.

```
import javax.swing.*;
public class Main {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Programming JCheckbox");
        JLabel label = new JLabel("Check Box Demonestration");
        label.setBounds(100, 50, 200, 50);
        JCheckBox checkBox1 = new JCheckBox("Core Jave");
        checkBox1.setBounds(100, 90, 150, 50);
        JCheckBox checkBox2 = new JCheckBox("Advanced Java", true);
        checkBox2.setBounds(100, 130, 150, 50);
        JCheckBox checkBox3 = new JCheckBox("Django Framework");
        checkBox3.setBounds(100, 170, 150, 50);

        jframe.add(label);
        jframe.add(checkBox1);
        jframe.add(checkBox2);
        jframe.add(checkBox3);

        jframe.setSize(400, 400);
        jframe.setLayout(null);
        jframe.setVisible(true);

    }
};
```



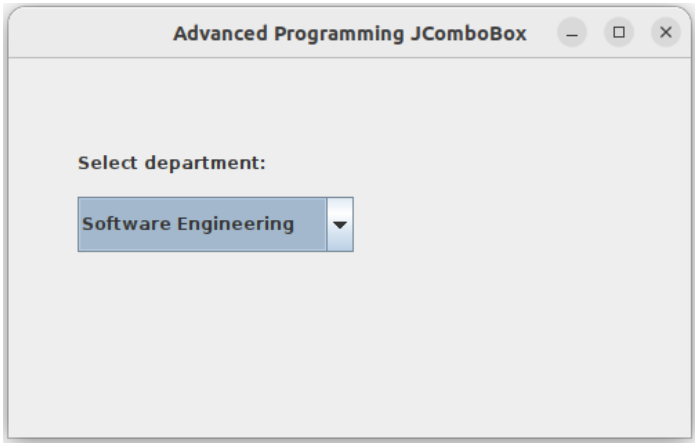
- JComboBox class
  - is used to show popup menu of choices.
  - Choice selected by user is shown on the top of a menu
  - has Constructor :

Methods	Description
JComboBox()	Creates a JComboBox with a default data model.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified array.
JComboBox(Vector<?> items)	Creates a JComboBox that contains the elements in the specified Vector.

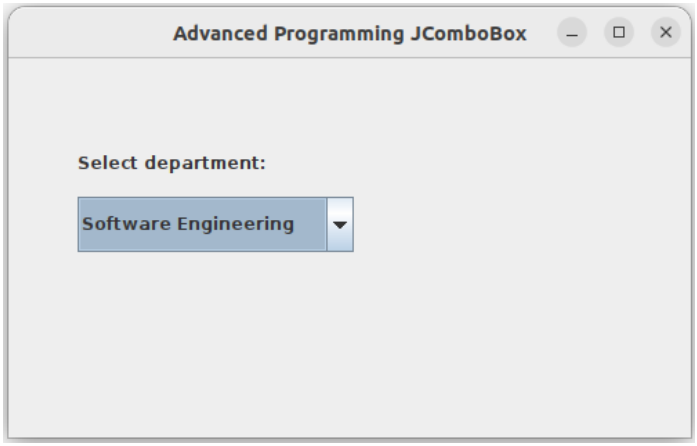
  - has various methods:

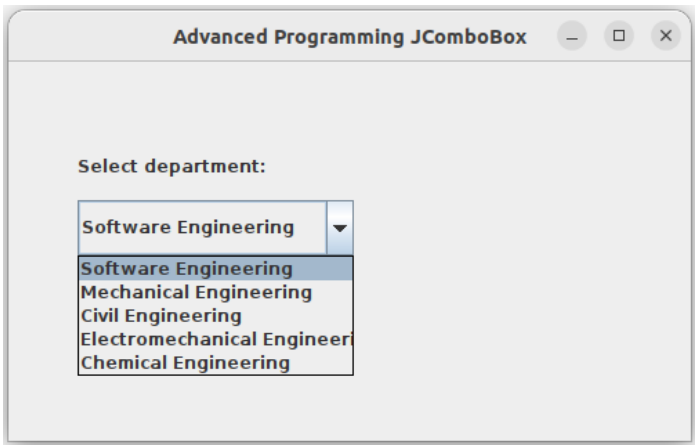
Methods	Description
void addItem(Object anObject)	It is used to add an item to the item list.
void removeItem(Object anObject)	It is used to delete an item to the item list.
void removeAllItems()	It is used to remove all the items from the list.
void setEditable(boolean b)	It is used to determine whether the JComboBox is editable.
void addActionListener(ActionListener a)	It is used to add the ActionListener.
void addItemListener(ItemListener i)	It is used to add the ItemListener.

```
import javax.swing.*;

public class Main {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Programming JComboBox");
        JLabel label = new JLabel("Select department:");
        label.setBounds(50, 50, 200, 50);
        String department[] = {
           "Software Engineering", "Mechanical Engineering",
           "Civil Engineering", "Electromechanical Engineering",
           "Chemical Engineering"
        };
        JComboBox box = new JComboBox<>(department);
        box.setBounds(50, 100, 200, 40);

        jframe.add(box);
        jframe.add(label);
        jframe.setSize(400, 400);
        jframe.setLayout(null);
        jframe.setVisible(true);

    }
};
```



```
import javax.swing.*;

public class Main {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Programming JComboBox");
        JLabel label = new JLabel("Select department:");
        label.setBounds(50, 50, 200, 50);
        String department[] = {
           "Software Engineering", "Mechanical Engineering",
           "Civil Engineering", "Electromechanical Engineering",
           "Chemical Engineering"
        };
        JComboBox box = new JComboBox<>(department);
        box.setBounds(50, 100, 200, 40);

        jframe.add(box);
        jframe.add(label);
        jframe.setSize(400, 400);
        jframe.setLayout(null);
        jframe.setVisible(true);

    }
};
```



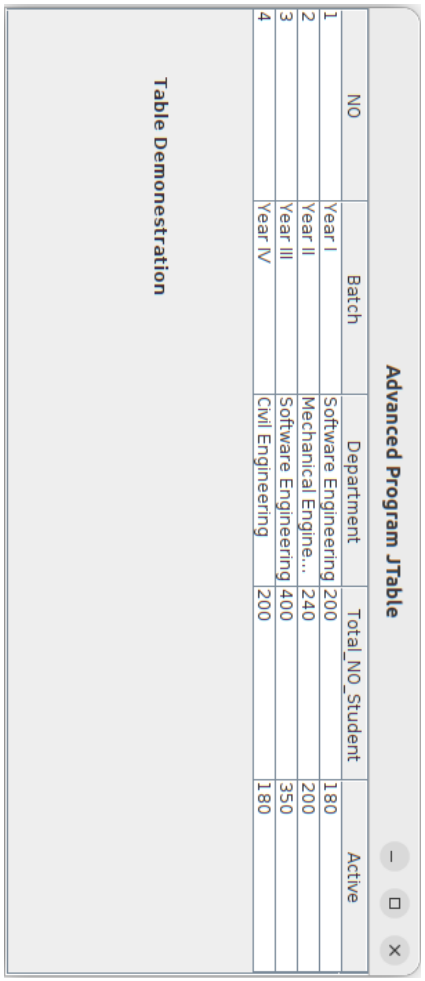


- JTable class
  - The JTable class is used to display data in tabular form. It is composed of rows and columns.
  - javax.swing.JTable
  - has Constructor :

Methods	Description
JTable()	Creates a JComboBox with a default data model.
JTable(Object[ ][ ] data, Object[ ] headers)	Creates a table with the specified data.

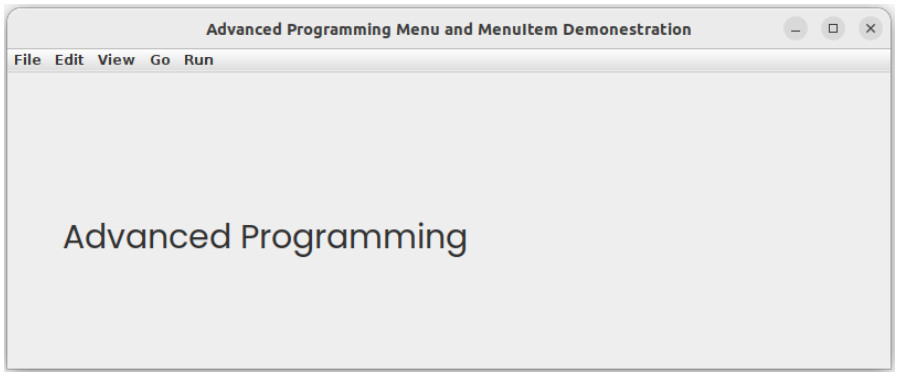
- JMenuBar, JMenu and JMenuItem
  - The JMenuBar class is used to display menubar on the window or frame. It may have several menus.
  - The object of JMenu class is a pull down menu component which is displayed from the menu bar.
  - The object of JMenuItem class adds a simple labeled menu item.
  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

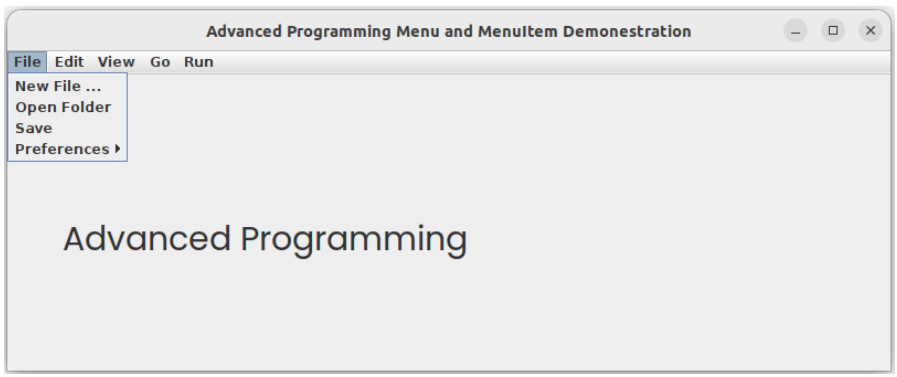
```
import javax.swing.*;
public class Main {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Program JTable");
        JLabel jlabel = new JLabel("Table Demonestration");
        jlabel.setBounds(50, 0, 200, 300);
        String data[][] = {
                { "1", "Year I", "Software Engineering", "200", "180" },
                { "2", "Year II", "Mechanical Engineering", "240", "200" },
                { "3", "Year III", "Software Engineering", "400", "350" },
                { "4", "Year IV", "Civil Engineering", "200", "180" },
        };
        String column[] = { "NO", "Batch", "Department", "Total_NO_Student",
         "Active" };

        JTable jtable = new JTable(data, column);
        jtable.setBounds(50, 100, 200, 300);

        jframe.add(jlabel);
        JScrollPane scroll = new JScrollPane(jtable);
        jframe.add(scroll);
        jframe.setSize(700, 300);
        jframe.setVisible(true);

    }
};
```

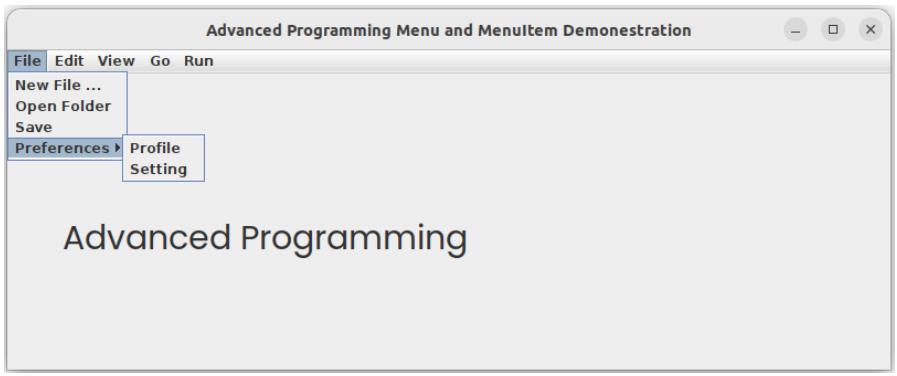


```
    import java.awt.*;
import javax.swing.*;
public class Main {
    public static void main(String args[]) {

        JMenu menu, submenu, edit, view, go, run;
        JMenuItem item1, item2, item3, item4, item5;

        JFrame jframe = new JFrame("
           Advanced Programming Menu and MenuItem Demonestration
        ");
        JMenuBar mb = new JMenuBar();
        JLabel jlabel = new JLabel("Advanced Programming");
        jlabel.setBounds(50, 100, 400, 100);
        jlabel.setFont(new Font("Poppins", Font.PLAIN, 30));

        menu = new JMenu("File");
        edit = new JMenu("Edit");
        view = new JMenu("View");
        go = new JMenu("Go");
        run = new JMenu("Run");

        submenu = new JMenu("Preferences");
        item1 = new JMenuItem("New File ...");
        item2 = new JMenuItem("Open Folder");
        item3 = new JMenuItem("Save");
        item4 = new JMenuItem("Profile");
        item5 = new JMenuItem("Setting");

        menu.add(item1);
          menu.add(item2);
          menu.add(item3);
          submenu.add(item4);
          submenu.add(item5);
          menu.add(submenu);
          mb.add(menu);
          mb.add(edit);
          mb.add(view);
          mb.add(go);
          mb.add(run);

          jframe.add(jlabel);

          jframe.setJMenuBar(mb);
          jframe.setSize(800, 400);
          jframe.setLayout(null);
          jframe.setVisible(true);

    }
};
```

- Output


Advanced Programming Menu and MenuItem Demonestration
File Edit View Go Run
Advanced Programming


Advanced Programming Menu and MenuItem Demonestration
File Edit View Go Run
New File ...
Open Folder
Save
Preferences
Advanced Programming


Advanced Programming Menu and MenuItem Demonestration
File Edit View Go Run
New File ...
Open Folder
Save
Preferences
Profile
Setting
Advanced Programming

- JTree class
  - used to display the tree structured data or hierarchical data.
  - JTree has a 'root node' $\mapsto$ Top-most parent of all nodes
  - java swing API defines JTree

    ```
    public class JTree extends JComponent implements Scrollable, Accessible
    ```

  - nodes represented in TreeNode inerface in Swing API.

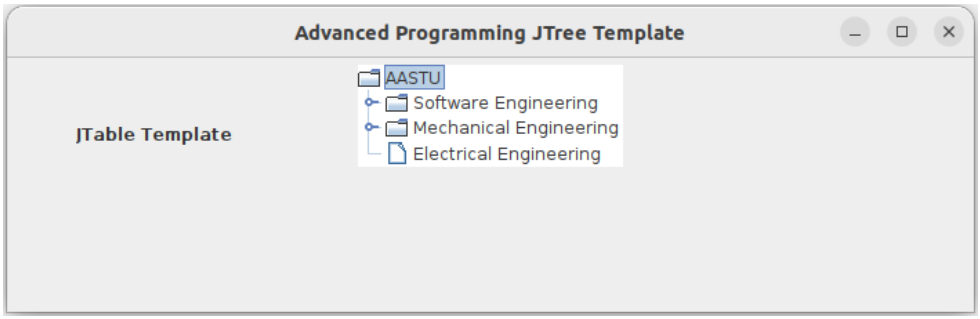
    $\uparrow$ extends

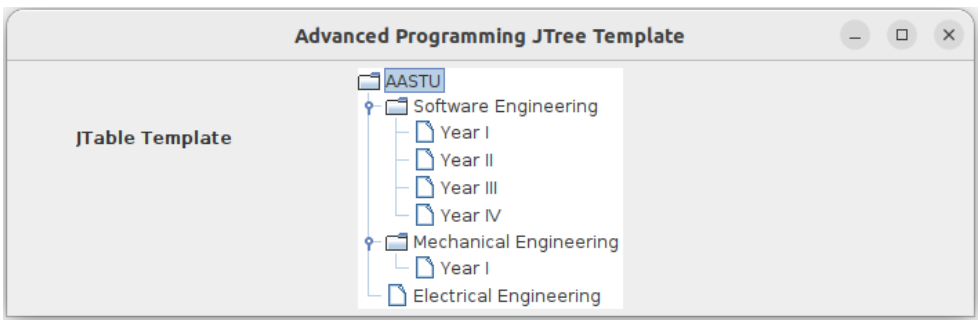
    MutableTreeNode interface $\mapsto$ DefaultMutableTreeNode class
  - has 3 constructors:

Methods	Description
JTree()	Creates a JTree with a sample model.
JTree(Object[] value)	Creates a JTree with every element of the specified array as the child of a new root node.
JTree(TreeNode root)	Creates a JTree with the specified TreeNode as its root, which displays the root node.

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

public class JtreeDemonestration {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Programming JTree Template");
        JPanel jpanel = new JPanel();
        jpanel.setBounds(100, 500, 200, 200);
        JLabel header = new JLabel("JTable Template");
        header.setBounds(50, 30, 200, 50);

        DefaultMutableTreeNode aastu = new DefaultMutableTreeNode("AASTU");
        DefaultMutableTreeNode sweng = new DefaultMutableTreeNode("Software Engineering");
        DefaultMutableTreeNode mech = new DefaultMutableTreeNode("Mechanical Engineering");
        DefaultMutableTreeNode elec = new DefaultMutableTreeNode("Electrical Engineering");

        DefaultMutableTreeNode year1 = new DefaultMutableTreeNode("Year I");
        aastu.add(sweng);
        aastu.add(mech);
        aastu.add(elec);

        DefaultMutableTreeNode y1 = new DefaultMutableTreeNode("Year I");
        DefaultMutableTreeNode y2 = new DefaultMutableTreeNode("Year II");
        DefaultMutableTreeNode y3 = new DefaultMutableTreeNode("Year III");
        DefaultMutableTreeNode y4 = new DefaultMutableTreeNode("Year IV");
```

```
        sweng.add(y1);
        sweng.add(y2);
        sweng.add(y3);
        sweng.add(y4);

        mech.add(year1);
        jframe.add(header);
        JTree jtree = new JTree(aastu);
        jpanel.add(jtree);
        jframe.add(jpanel);
        jframe.setSize(700, 400);
        jframe.setVisible(true);

    }
};
```

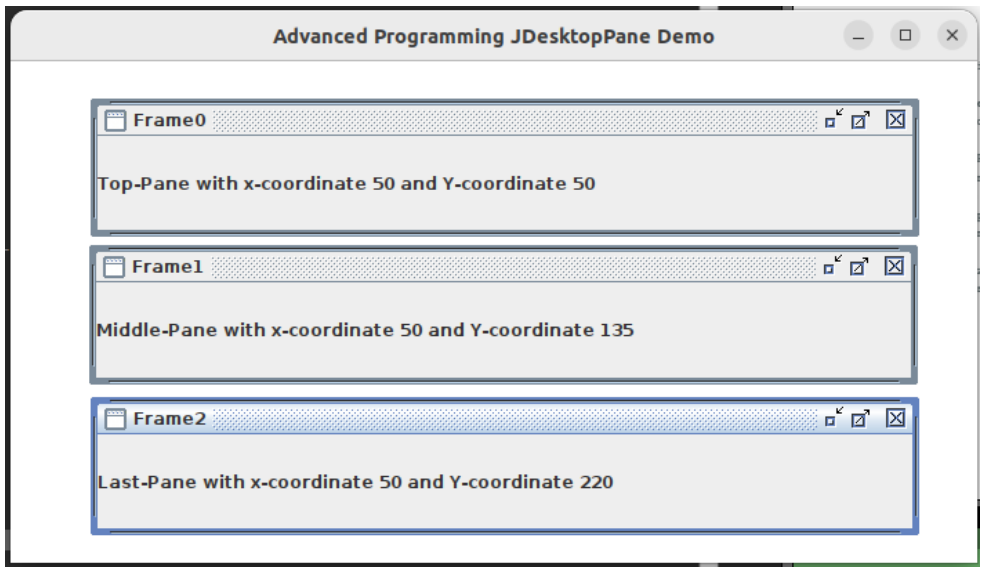




- JDesktopPane class
  - can be used to create "multi-document" applications.
  - A multi-document application can have many windows included in it.
  - How ?
    - First making the contentPane in the main window as an instance of the JDesktopPane class or a subclass.
    - Then, Internal windows add instances of JInternalFrame to the JdesktopPane instance.
  - See the example:

```
import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
class ChildDesktopPane extends JDesktopPane {
    int no_of_frame = 3;
    int x = 50, y = 50;
    public void showInternalFrame(ChildDesktopPane childpane) {
        for (int i = 0; i < no_of_frame; ++i) {
            JInternalFrame jf = new JInternalFrame(
                "Frame" + i, true, true, true, true
            );
            jf.setBounds(x, y, 600, 100);
            Container cnt = jf.getContentPane();
            if (i == 0) {
                cnt.add(new JLabel("Top-Pane with x-coordinate "
                    + x + " and Y-coordinate " + y));
            } else if (i == 1) {
                cnt.add(new JLabel("Middle-Pane with x-coordinate
                    + x + " and Y-coordinate " + y));
            } else {
                cnt.add(new JLabel("Last-Pane with x-coordinate "
                    + x + " and Y-coordinate " + y));
            }
            childpane.add(jf);
            jf.setVisible(true);
            y += 50;
        }
    }
}
```

```
public class Main {
    public static void main(String args[]) {

        JFrame jframe = new JFrame(
            "Advanced Programming JTree Template"
        );
        ChildDesktopPane childpane;
        chilepane = new ChildDesktopPane();
        Container cp = jframe.getContentPane();
        cp.add(childpane, BorderLayout.CENTER);
        dp.showInternalFrame(childpane);

        jframe.setSize(700, 400);
        jframe.setVisible(true);

    }
}
```



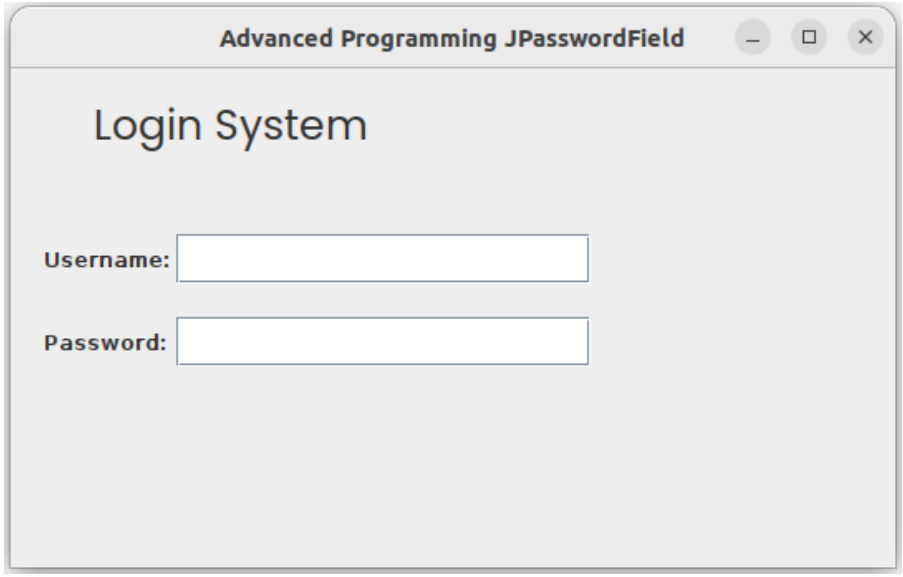
- JPasswordField class
  - A text component for password entry. $\mapsto$ JTextField class.
  - It allows the editing of a single line of text.
  - defined

    ↘

    public class JPasswordField extends JTextField

Constructor	Description
JPasswordField()	Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width.
JPasswordField(int columns)	Constructs a new empty JPasswordField with the specified number of columns.
JPasswordField(String text)	Constructs a new JPasswordField initialized with the specified text.
JPasswordField(String text, int columns)	Construct a new JPasswordField initialized with the specified text and columns.

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

public class Main {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("
           Advanced Programming JPasswordField
        ");

        JTextField username = new JTextField();
        JLabel jlabel = new JLabel("Login System");
        JLabel textFiled = new JLabel("Username:");

        JPasswordField password = new JPasswordField();
        JLabel passwordLabel = new JLabel("Password:");
        password.setToolTipText("Enter Password");

        textFiled.setBounds(20, 100, 80, 30);
        username.setBounds(100, 100, 250, 30);
        username.setToolTipText("Enter Username");

        jlabel.setBounds(50, 20, 250, 30);
        jlabel.setFont(new Font("Poppins", Font.PLAIN, 25));
        passwordLabel.setBounds(20, 150, 80, 30);
        password.setBounds(100, 150, 250, 30);
```




```
 jframe.add(textFiled);
        jframe.add(username);

        jframe.add(jlabel);
        jframe.add(password);
        jframe.add(passwordLabel);
        jframe.setSize(500, 500);
        jframe.setLayout(null);
        jframe.setVisible(true);

    }
}
```

↦ How Java Handles Graphics elements ?

↦ How Java Handles Graphics elements ?

- Java provides a class called Graphics class
- has a constructor
  - Graphics(): It is used to create an object by using the new keyword.
- has methods :

Method	Description
public abstract void drawString(String str, int x, int y)	is used to draw the specified string.
public void drawRect(int x, int y, int width, int height)	draws a rectangle with the specified width and height.
public abstract void fillRect(int x, int y, int width, int height)	is used to fill rectangle with the default color and specified width and height.
public abstract void drawOval(int x, int y, int width, int height):	is used to draw oval with the specified width and height.
public abstract void fillOval(int x, int y, int width, int height)	is used to fill oval with the default color and specified width and height.
public abstract void drawLine(int x1, int y1, int x2, int y2)	: is used to draw line between the points(x1, y1) and (x2, y2)
public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)	: is used draw the specified image.

↦ Any other UI Components ?

↦ Any other UI Components ?

- JList
- JOptionPane
- JScrollBar
- JavaSpinner
- JSlider
- JDialog
- JSlider
- JTextPane
- JRootPane
- JEditorPane
- JFileChooser
- JScrollPane

- Component Arrangement or Organization ?
  - LayoutManager is responsible for the components' layout in GUI applications.
  - LayoutManager is an interface and it is implemented by all the layout manager classes.

LayoutManager	Description
java.awt.BorderLayout	Components are laid out to fit in five directions namely center, east, west, south, north.
java.awt.FlowLayout	This is the default layout. It lays the components in the directional flow.
java.awt.GridLayout	Arranges the components in a rectangular grid.
javax.swing.BoxLayout	Components are arranged in a box.
java.awt.CardLayout	Each component is viewed as a card in a deck and at a time only one component is visible.
javax.swing.GroupLayout	groups its components and places them in a Container hierarchically.
java.awt.GridBagLayout	Arranges components vertically, horizontally, or even along their baselines. Components need not be of the same size..
javax.swing.ScrollPaneLayout	Used by JScrollPane class and is responsible for arranging components in scrollable containers.
javax.swing.SpringLayout	A set of constraints like the horizontal and vertical distance between components etc and Also is provided and the components are arranged according to these set of constraints.

- BorderLayout class
  - to arrange the components in five regions:

    ↓

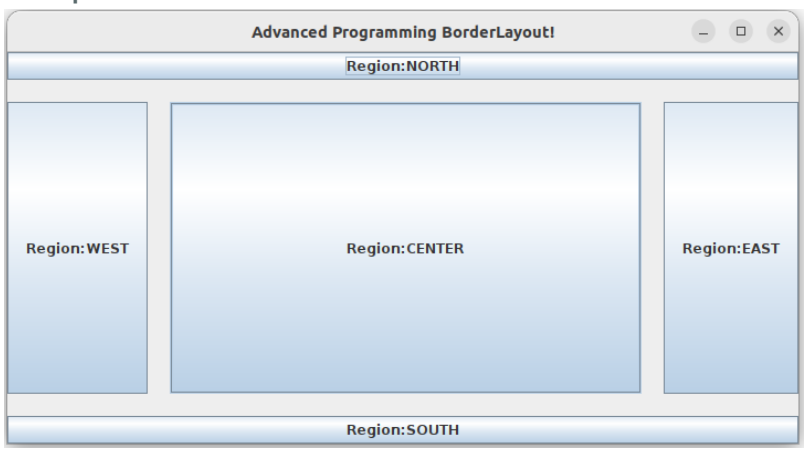
    north, south, east, west and center
  - The BorderLayout provides five constants for each region:
    1. public static final int NORTH
    2. public static final int SOUTH
    3. public static final int EAST
    4. public static final int WEST
    5. public static final int CENTER
  - has 2 Constructor:
    - BorderLayout():
      - creates a border layout but with no gaps between the components.
    - BorderLayout(int horizontalgap, int verticalgap):
      - creates a border layout with the given horizontal and vertical gaps between the components.

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Programming BorderLayout!");

        JButton jbtn1 = new JButton("Region:NORTH");
        JButton jbtn2 = new JButton("Region:SOUTH");
        JButton jbtn3 = new JButton("Region:EAST");
        JButton jbtn4 = new JButton("Region:WEST");
        JButton jbtn5 = new JButton("Region:CENTER");
        jframe.setLayout(new BorderLayout(20, 20));

        jframe.add(jbtn1, BorderLayout.NORTH);
        jframe.add(jbtn2, BorderLayout.SOUTH);
        jframe.add(jbtn3, BorderLayout.EAST);
        jframe.add(jbtn4, BorderLayout.WEST);
        jframe.add(jbtn5, BorderLayout.CENTER);

        jframe.setSize(500, 500);
        jframe.setVisible(true);

    }
}
```

Output:

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {

        JFrame jframe = new JFrame("Advanced Programming BorderLayout!");

        JButton jbtn1 = new JButton("Region:NORTH");
        JButton jbtn2 = new JButton("Region:SOUTH");
        JButton jbtn3 = new JButton("Region:EAST");
        JButton jbtn4 = new JButton("Region:WEST");
        JButton jbtn5 = new JButton("Region:CENTER");
        jframe.setLayout(new BorderLayout(20, 20));

        jframe.add(jbtn1, BorderLayout.NORTH);
        jframe.add(jbtn2, BorderLayout.SOUTH);
        jframe.add(jbtn3, BorderLayout.EAST);
        jframe.add(jbtn4, BorderLayout.WEST);
        jframe.add(jbtn5, BorderLayout.CENTER);

        jframe.setSize(500, 500);
        jframe.setVisible(true);

    }
}
```

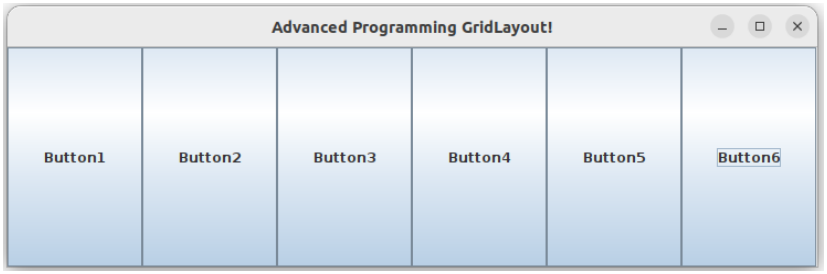
Output:



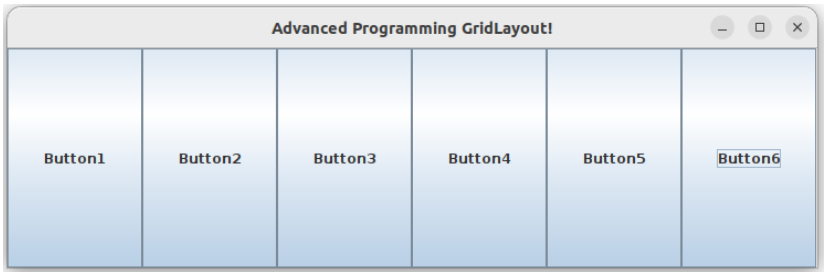
- GridLayout class
  - To arrange the components in a rectangular grid.
  - Each component is displayed in each rectangle.
  - has 2 Constructor:
    - GridLayout():
      - creates a grid layout with one column per component in a row.
    - GridLayout(int rows, int columns): :
      - creates a grid layout with the given rows and columns but no gaps between the components.
    - GridLayout(int rows, int columns, int hgap, int vgap)
      - creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

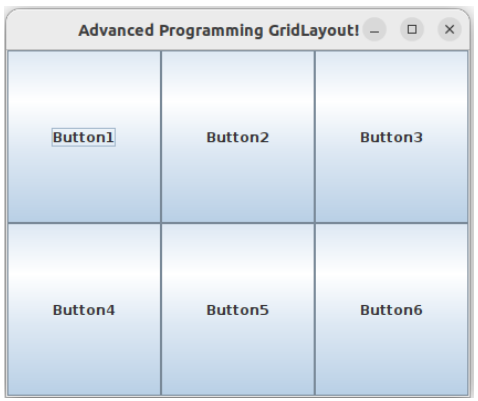
```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {

        JFrame jframe = new JFrame(
             "Advanced Programming BorderLayout!"
         );

        JButton jbtn1 = new JButton("Button1");
        JButton jbtn2 = new JButton("Button2");
        JButton jbtn3 = new JButton("Button3");
        JButton jbtn4 = new JButton("Button4");
        JButton jbtn5 = new JButton("Button5");
        JButton jbtn6 = new JButton("Button6");

        jframe.setLayout(new GridLayout());
        jframe.setLayout(new GridLayout(2,3));
        jframe.setLayout(new GridLayout(3,2,20,20));

        jframe.add(jbtn1);
        jframe.add(jbtn2);
        jframe.add(jbtn3);
        jframe.add(jbtn4);
        jframe.add(jbtn5);
        jframe.add(jbtn6);

        jframe.setSize(500, 500);
        jframe.setVisible(true);

    }
}
```

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {

        JFrame jframe = new JFrame(
             "Advanced Programming BorderLayout!"
         );

        JButton jbtn1 = new JButton("Button1");
        JButton jbtn2 = new JButton("Button2");
        JButton jbtn3 = new JButton("Button3");
        JButton jbtn4 = new JButton("Button4");
        JButton jbtn5 = new JButton("Button5");
        JButton jbtn6 = new JButton("Button6");

        jframe.setLayout(new GridLayout());
        jframe.setLayout(new GridLayout(2,3));
        jframe.setLayout(new GridLayout(3,2,20,20));

        jframe.add(jbtn1);
        jframe.add(jbtn2);
        jframe.add(jbtn3);
        jframe.add(jbtn4);
        jframe.add(jbtn5);
        jframe.add(jbtn6);

        jframe.setSize(500, 500);
        jframe.setVisible(true);

    }
}
```

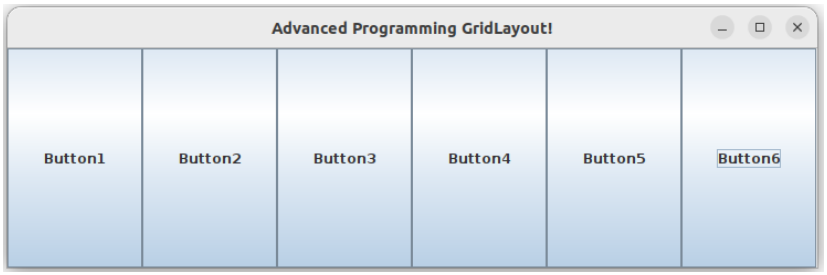


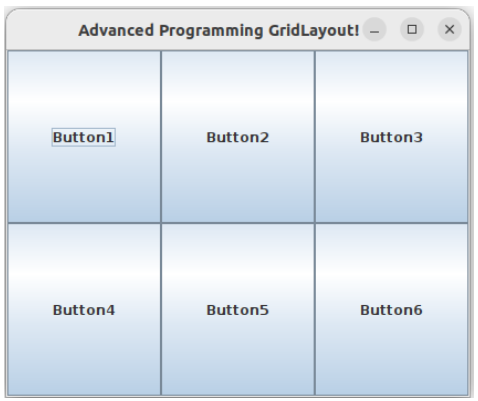
```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {

        JFrame jframe = new JFrame(
             "Advanced Programming BorderLayout!"
         );

        JButton jbtn1 = new JButton("Button1");
        JButton jbtn2 = new JButton("Button2");
        JButton jbtn3 = new JButton("Button3");
        JButton jbtn4 = new JButton("Button4");
        JButton jbtn5 = new JButton("Button5");
        JButton jbtn6 = new JButton("Button6");

        jframe.setLayout(new GridLayout());
        jframe.setLayout(new GridLayout(2,3));
        jframe.setLayout(new GridLayout(3,2,20,20));

        jframe.add(jbtn1);
        jframe.add(jbtn2);
        jframe.add(jbtn3);
        jframe.add(jbtn4);
        jframe.add(jbtn5);
        jframe.add(jbtn6);

        jframe.setSize(500, 500);
        jframe.setVisible(true);

    }
}
```





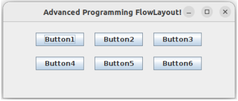
```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {

        JFrame jframe = new JFrame(
             "Advanced Programming BorderLayout!"
         );

        JButton jbtn1 = new JButton("Button1");
        JButton jbtn2 = new JButton("Button2");
        JButton jbtn3 = new JButton("Button3");
        JButton jbtn4 = new JButton("Button4");
        JButton jbtn5 = new JButton("Button5");
        JButton jbtn6 = new JButton("Button6");

        jframe.setLayout(new GridLayout());
        jframe.setLayout(new GridLayout(2,3));
        jframe.setLayout(new GridLayout(3,2,20,20));

        jframe.add(jbtn1);
        jframe.add(jbtn2);
        jframe.add(jbtn3);
        jframe.add(jbtn4);
        jframe.add(jbtn5);
        jframe.add(jbtn6);

        jframe.setSize(500, 500);
        jframe.setVisible(true);

    }
}
```





- FlowLayout class
  - To arrange the components in a line, one after another.
  - Fields of FlowLayout class
    - public static final int LEFT
    - public static final int RIGHT
    - public static final int CENTER
    - public static final int LEADING
    - public static final int TRAILING
  - has 2 Constructor:
    - FlowLayout():
      - creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
    - FlowLayout(int align)
      - creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
    - FlowLayout(int align, int hgap, int vgap)
      - creates a flow layout with the given alignment and the given horizontal and vertical gap.

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo {
    public static void main(String args[]) {
        JFrame jframe = new JFrame(
             Advanced Programming FlowLayout!"
         );
        JButton jbtn1 = new JButton("Button1");
        JButton jbtn2 = new JButton("Button2");
        JButton jbtn3 = new JButton("Button3");
        JButton jbtn4 = new JButton("Button4");
        JButton jbtn5 = new JButton("Button5");
        JButton jbtn6 = new JButton("Button6");

        jframe.setLayout(new FlowLayout());
        jframe.setLayout(new FlowLayout(FlowLayout.LEFT));
        jframe.setLayout(new FlowLayout(FlowLayout.RIGHT));
        // jframe.setLayout(new FlowLayout(FlowLayout.LEADING));
        // jframe.setLayout(new FlowLayout(FlowLayout.TRAILING));
        // jframe.setLayout(new FlowLayout(FlowLayout.CENTER));
        jframe.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));
        jframe.add(jbtn1);
        jframe.add(jbtn2);
        jframe.add(jbtn3);
        jframe.add(jbtn4);
        jframe.add(jbtn5);
        jframe.add(jbtn6);
        jframe.setSize(500, 500);
        jframe.setVisible(true); }}
```



- CardLayout class
  - devised to manages the components in such a manner that only one component is visible at a time.
  - It treats each component as a card that is why it is known as CardLayout.
  - has 2 Constructor:
    - CardLayout():
      - creates a card layout with zero horizontal and vertical gap.
    - CardLayout(int hgap, int vgap)
      - creates a card layout with the given horizontal and vertical gap.
  - Has methods!

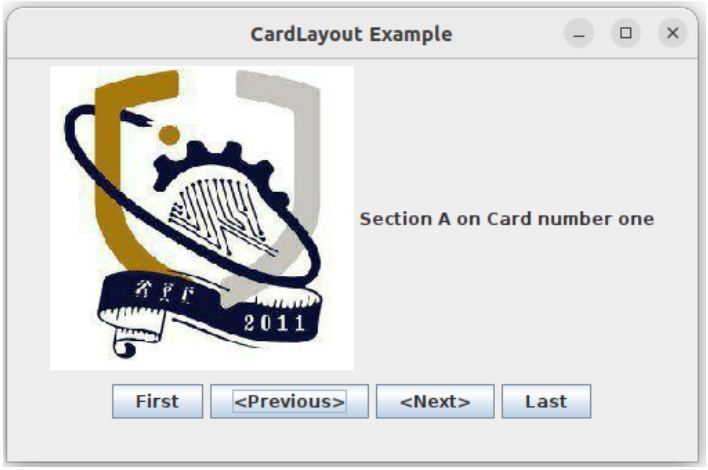
Methods	Description
public void next(Container parent)	is used to flip to the next card of the given container.
public void previous(Container parent):	is used to flip to the previous card of the given container.
public void first(Container parent)	is used to flip to the first card of the given container.
public void last(Container parent)	is used to flip to the last card of the given container.
public void show(Container parent, String name)	is used to flip to the specified card with the given name.

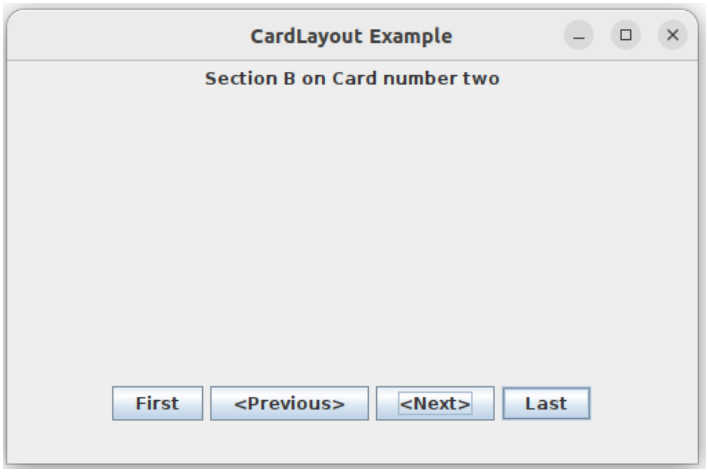
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class CardLayoutDemoMain extends JFrame {

    CardLayout cardlayout;
    ImageData img = new ImageData();
    private int state = 1;

    CardLayoutDemoMain() {
        setTitle("Advanced Programming CardLayout Example");
        setSize(500, 500);

        ImageIcon aastu = new ImageIcon("logo.png");
        Image aastulogo = aastu.getImage();
        Image resizedLogo =
            aastulogo.getScaledInstance(120, 120,
            java.awt.Image.SCALE_SMOOTH);

        JPanel cpane = new JPanel();
        cardlayout = new CardLayout();
        cpane.setLayout(cardlayout);

        JPanel jpanel1 = new JPanel();
        JPanel jpanel2 = new JPanel();
        JPanel jpanel3 = new JPanel();
```

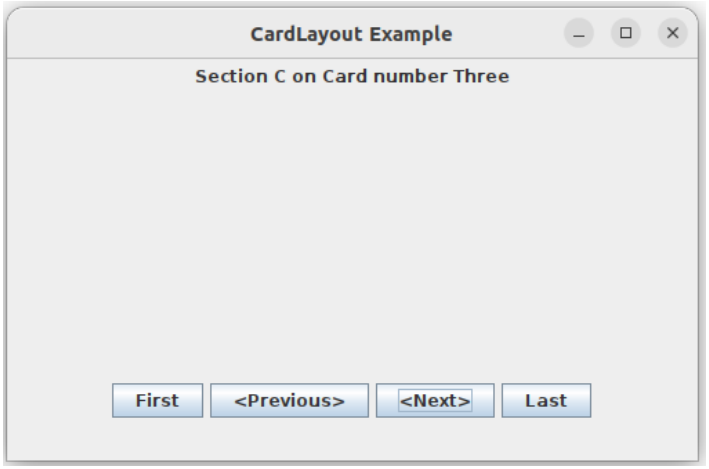
```
fButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                cardlayout.first(cpane);
                state = 1;
            }
        });

        lButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                cardlayout.last(cpane);
                state = 4;
            }
        });

        nButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                if (state < 3) {
                    state = state + 1;
                    cardlayout.show(cpane, "" + (state));
                }
            }
        });
```

```
JLabel jlabel = new JLabel("Section A on Card number one");
JLabel jlabel2 = new JLabel("Section B on Card number two");
JLabel jlabel3 = new JLabel("Section C on Card number Three");

jpanel1.add(jlabel);
jpanel1.add(resizedLogo);
jpanel2.add(jlabel2);
jpanel3.add(jlabel3);

cpane.add(jpanel1, "1");
cpane.add(jpanel2, "2");
cpane.add(jpanel3, "3");

JPanel btnPanel = new JPanel();
JButton fButton = new JButton("First");
JButton pButton = new JButton("<Previous>");
JButton nButton = new JButton("<Next>");
JButton lButton = new JButton("Last");

btnPanel.add(fButton);
btnPanel.add(pButton);
btnPanel.add(nButton);
btnPanel.add(lButton);
```

```
pButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent event) {
              if (state > 1) {
                  state = state - 1;
                  cardlayout.show(cpane, "" + (state));
              }
          }
      });

      getContentPane().add(cpane, BorderLayout.NORTH);

      getContentPane().add(btnPanel, BorderLayout.CENTER);

  }

  public static void main(String args[]) {

      Main m = new Main();
      m.setDefaultCloseOperation(EXIT_ON_CLOSE);
      m.setVisible(true);

  }
}
```


CardLayout Example
Section A on Card number one
First
<Previous>
<Next>
Last


CardLayout Example
Section B on Card number two
First
<Previous>
<Next>
Last


CardLayout Example
Section C on Card number Three
First
<Previous>
<Next>
Last

- GroupLayout class
  - groups its components and places them in a Container hierarchically.
  - grouping is done by instances of the Group class.
  - Group is an abstract class. $\mapsto$ has 2 concrete class
    - SequentialGroup $\mapsto$ positions its child sequentially one after another.

      $\downarrow$

      createSequentialGroup()
    - ParallelGroup $\mapsto$aligns its child on top of each other.

      $\downarrow$

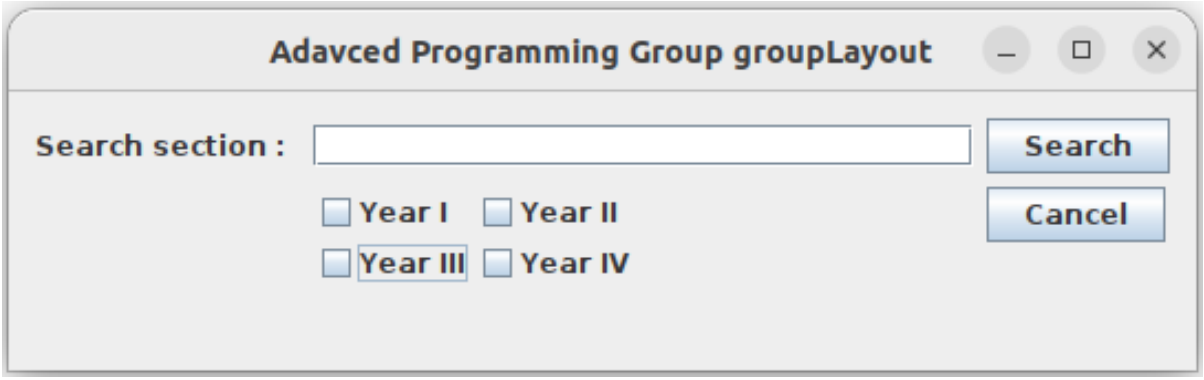
      createParallelGroup()
  - has constructor:
    - GroupLayout(Container c) $\mapsto$ It creates a GroupLayout for the specified Container.
  - Has Fields:
    - DEFAULT_SIZE

      $\downarrow$

      It indicates the size from the component or gap should be used for a particular range value.
    - PREFERRED_SIZE

      $\downarrow$

      It indicates the preferred size from the component or gap should be used for a particular range value.




```
import java.awt.*;
import javax.swing.*;
public class GroupLayoutDemo {
   public static void main(String args[]) {
     JFrame jframe = new JFrame("
        Adavced Programming Group groupLayout
     ");

     JLabel jlabel = new JLabel("Search section :");
     JTextField jtextField = new JTextField(5);
     JCheckBox year1 = new JCheckBox("Year I");
     JCheckBox year2 = new JCheckBox("Year II");
     JCheckBox year3 = new JCheckBox("Year III");
     JCheckBox year4 = new JCheckBox("Year IV");
     JButton findBtn = new JButton("Search");
     JButton cancelBtn = new JButton("Cancel");
```

```
groupLayout.setVerticalGroup(groupLayout.createSequentialGroup()
      .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
         .addComponent(jlabel)
         .addComponent(jtextField)
         .addComponent(findBtn))
      .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
         .addGroup(groupLayout.createSequentialGroup()
             .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.BASE
                 .addComponent(year1)
                 .addComponent(year2))
             .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.BASE
                 .addComponent(year3)
                 .addComponent(year4)))
             .addComponent(cancelBtn))

            );
```

```
Container contentPane = jframe.getContentPane();
GroupLayout groupLayout = new GroupLayout(contentPane);

contentPane.setLayout(groupLayout);
groupLayout.setAutoCreateGaps(true);
groupLayout.setAutoCreateContainerGaps(true);

groupLayout.setHorizontalGroup(groupLayout.createSequentialGroup()
    .addComponent(jlabel)
    .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
        .addComponent(jtextField)
        .addGroup(groupLayout.createSequentialGroup()
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addComponent(year1)
            .addComponent(year3))
            .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                .addComponent(year2)
                .addComponent(year4))))
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                .addComponent(findBtn)
                .addComponent(cancelBtn))
);
```

```
jframe.pack();
jframe.setSize(500, 150);
jframe.setVisible(true);
jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CL
}
}
```

- SpringLayout
  - A arranges the children of its associated container according to a set of constraints.
  - Constraints are nothing but horizontal and vertical distance between two-component edges.
  - Every constraint is represented by a SpringLayout.
  - SpringLayout()
    - The default constructor of the class is used to instantiate the SpringLayout class.
  - Field

Methods	Description
static String BASELINE	specifies the baseline of a component.
static String EAST	Specifies the right edge of a component's bounding rectangle.
static String HEIGHT	Specifies the height of a component's bounding rectangle.
static String HORIZONTAL_CENTER	Specifies the horizontal center of a component's bounding rectangle.
static String NORTH	Specifies the top edge of a component's bounding rectangle.
static String SOUTH	Specifies the bottom edge of a component's bounding rectangle.
static String WIDTH	Specifies the width of a component's bounding rectangle.

- Check the methods and ScrollPaneLaout Manager ....

▶ What is an Event?

- is a situation of changing the state of object.
- when ?
  - result of user interaction with the graphical user interface components
  - Events:
    - clicking on a button,
    - moving the mouse
    - entering a character through keyboard
    - selecting an item from the list, and
    - scrolling the page are the activities that causes an event to occur.

- Check the methods and ScrollPaneLaout Manager ....

▶ What is an Event?

- is a situation of changing the state of object.
- when ?
  - result of user interaction with the graphical user interface components
  - Events:
    - clicking on a button,
    - moving the mouse
    - entering a character through keyboard
    - selecting an item from the list, and
    - scrolling the page are the activities that causes an event to occur.

↓

How Events handled ?

↘

"Event Handling Mechanism"

- Event Handling
  - is the mechanism that controls the event then,

    ↙

    decides what should happen if an event occurs?.
  - snippet which handles the event

    ↓

    EVENT HANDLER $\mapsto$ java Event Model

    ↓

    defines the standard ways to generate and handle the events.
    - Key Participants in event handling
      - Source $\mapsto$ an object that generates an event.
        - Occurs when an internal state of that object changes in some way.
        - A source must register listeners in order for the listeners to receive the notifications about a specific type of event.
        - key in offering information about the event occurred to handler
      - Listener $\mapsto$ the event handler.
        - is an object that is notified when an event occurs.
        - is responsible for generating a response to an event.

        ↓

        import java.awt.event.*;

- Event classes represent the event. Java provides various Event classes
- The root class is EventObject

Event Class	Description	Listener
AWTEvent	It is the root event class for all SWING events. This class and its subclasses supercede the original java.awt.Event class.	
ActionEvent	The ActionEvent is generated when the button is clicked or the item of a list is double-clicked.	ActionListener
KeyEvent	On entering the character the Key event is generated.	KeyListener
MouseEvent	This event indicates a mouse action occurred in a component.	MouseListener and MouseMotionListener
WindowEvent	The object of this class represents the change in the state of a window.	WindowListener
AdjustmentEvent	The object of this class represents the adjustment event emitted by Adjustable objects.	AdjustmentListener
ComponentEvent	The object of this class represents the change in the state of a window.	ComponentListener
ContainerEvent	The object of this class represents the change in the state of a window.	ContainerListener
PaintEvent	The object of this class represents the change in the state of a window.	PaintListener

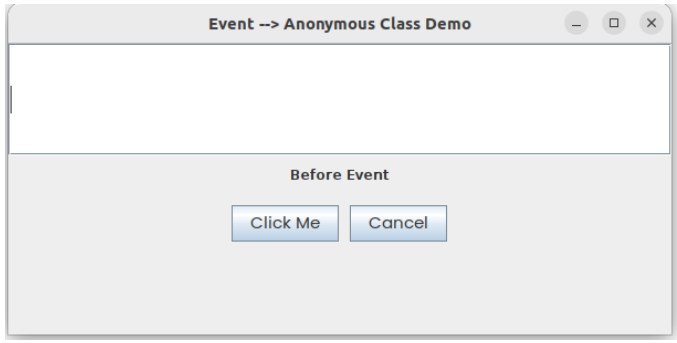
- A Listener has two major requirements.
  1. Listener should be registered to one more source object to receiving event notification
  2. Listener must implement methods to receive and process those notifications.
- How to handle the event ?
  - Register the component with the Listener

Components	Registration Methods
Button	public void addActionListener(ActionListener a)
MenuItem	public void addActionListener(ActionListener a)
TextField	• public void addActionListener(ActionListener a) • public void addTextListener(TextListener a)
TextArea	public void addTextListener(TextListener a)
Checkbox	public void addItemListener(ItemListener a)
Choice	public void addItemListener(ItemListener a)
List	• public void addActionListener(ActionListener a) • public void addItemListener(ItemListener a)

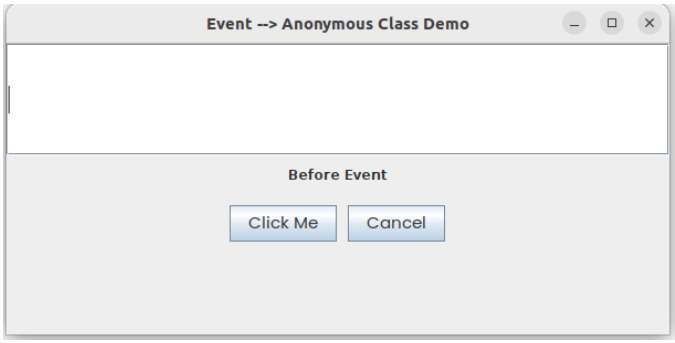
- Event handling snippets implementation
  1. Using Anonymous class

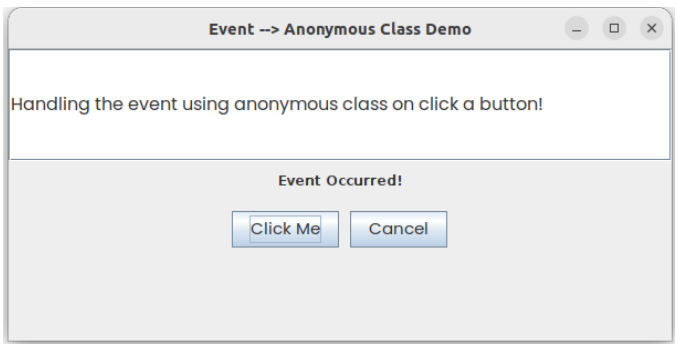
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AnonymousClassDemo {
    public static void main(String[] args) {
        JFrame jframe = new JFrame();

        JPanel jpanel = new JPanel(new BorderLayout(10, 10));

        JPanel jpanel_2 = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
        JLabel result = new JLabel("Before Event", JLabel.CENTER);

        JButton okButton = new JButton("Click Me");
        okButton.setFont(new Font("Poppins", Font.PLAIN, 15));
        JButton cancel = new JButton("Cancel");
        cancel.setFont(new Font("Poppins", Font.PLAIN, 15));

        JTextField textField = new JTextField(20);
        textField.setFont(new Font("Poppins", Font.PLAIN, 16));

        jpanel.add(textField, BorderLayout.CENTER);
        jpanel.add(result, BorderLayout.SOUTH);

        jpanel_2.add(okButton);
        jpanel_2.add(cancel);
```

```
    jframe.add(jpanel);
        jframe.add(jpanel_2);

        // Anonymous class event implmentation
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                textField.setText("
                   Handling the event using anonymous class on
                   click a button!
                ");
                 result.setText("Event Occurred!");
            }
        });

        cancel.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        jframe.setLayout(new GridLayout(2, 1, 10, 10));
        jframe.setTitle("Event --> Anonymous Class Demo");
        jframe.setFont(new Font("Poppins", Font.PLAIN, 15));
        jframe.setSize(600, 300);
        jframe.setVisible(true);
        jframe.setDefaultCloseOperation(jframe.EXIT_ON_CLOSE);
    }

}
```



```
    jframe.add(jpanel);
        jframe.add(jpanel_2);

        // Anonymous class event implmentation
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                textField.setText("
                   Handling the event using anonymous class on
                   click a button!
                ");
                 result.setText("Event Occurred!");
            }
        });

        cancel.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        jframe.setLayout(new GridLayout(2, 1, 10, 10));
        jframe.setTitle("Event --> Anonymous Class Demo");
        jframe.setFont(new Font("Poppins", Font.PLAIN, 15));
        jframe.setSize(600, 300);
        jframe.setVisible(true);
        jframe.setDefaultCloseOperation(jframe.EXIT_ON_CLOSE);
    }

}
```



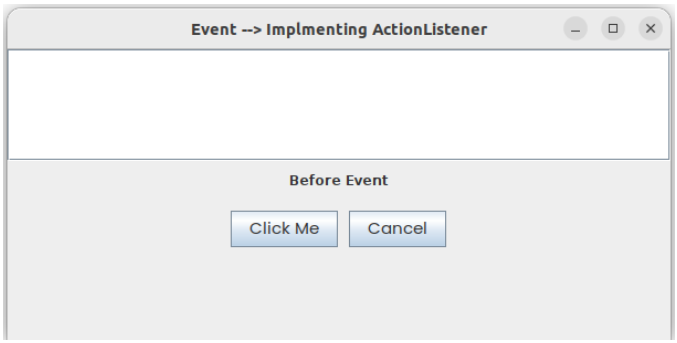


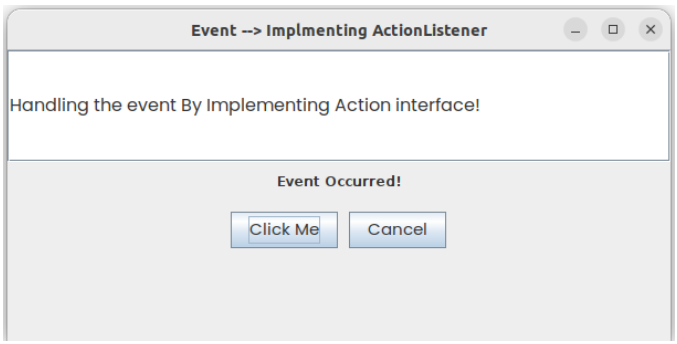
- Event handling snippets implementation

2 Implementing ActionListener

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

public class ImplementingActionListener implements ActionListener {

    JTextField textField;
    JLabel result;
    JButton okButton;
    JButton cancel;

    ImplementingActionListener() {
        JFrame jframe = new JFrame();

        JPanel jpanel = new JPanel(new BorderLayout(10, 10));
        jpanel.setBounds(100, 50, 200, 200);

        JPanel jpanel_2 = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
        result = new JLabel("Before Event", JLabel.CENTER);

        JButton okButton = new JButton("Click Me");
        okButton.setFont(new Font("Poppins", Font.PLAIN, 15));
        okButton.addActionListener(this);

        JButton cancel = new JButton("Cancel");
        cancel.setFont(new Font("Poppins", Font.PLAIN, 15));
        cancel.addActionListener(this);
```

```
        textField = new JTextField(20);
            textField.setFont(new Font("Poppins", Font.PLAIN, 16));

            jpanel.add(textField, BorderLayout.CENTER);
            jpanel.add(result, BorderLayout.SOUTH);

            jpanel_2.add(okButton);
            jpanel_2.add(cancel);

            jframe.add(jpanel);
            jframe.add(jpanel_2);

            jframe.setLayout(new GridLayout(2, 1, 10, 10));
            jframe.setTitle("Event --> Anonymous Class Demo");
            jframe.setFont(new Font("Poppins", Font.PLAIN, 15));
            jframe.setSize(600, 300);
            jframe.setVisible(true);
            jframe.setDefaultCloseOperation(jframe.EXIT_ON_CLOSE);
        }
        // event snippet cod
        public void actionPerformed(ActionEvent event) {
            if (event.getSource() == okButton) {
                textField.setText("Handling the event By Implementing Action
                result.setText("Event Occurred!");
            } else if (event.getSource() == cancel) {
                System.exit(0);
            }
        }

        public static void main(String[] args) {
            new ImplementingActionListener();
        }
}
```





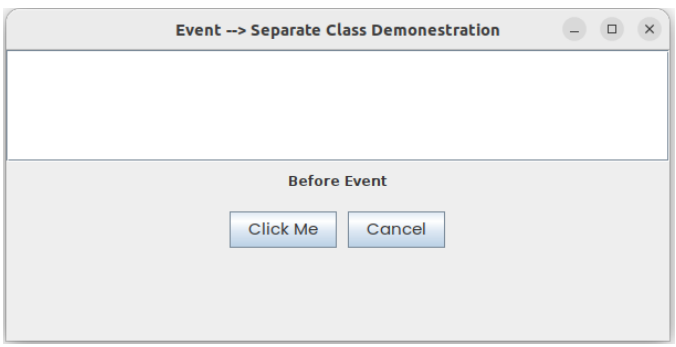
- Event handling snippets implementation

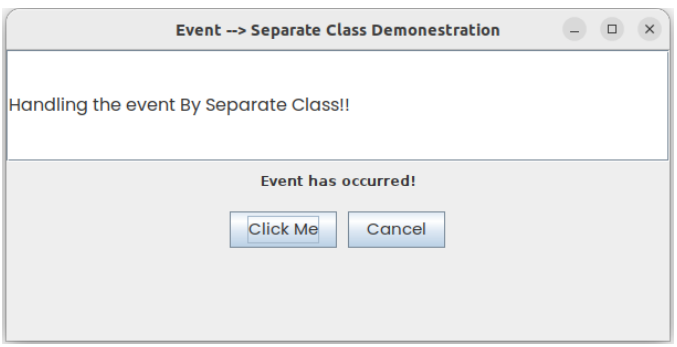
3 Using Separate Class

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

public class SeparateMain extends JFrame {
    JTextField textField;
    JButton okButton, cancel;
    JLabel result;
    JPanel jpanel, jpanel_2;

 SeparateMain() {
    jpanel = new JPanel(new BorderLayout(10, 10));
    jpanel.setBounds(100, 50, 200, 200);

    jpanel_2 = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
    result = new JLabel("Before Event", JLabel.CENTER);

    okButton = new JButton("Click Me");
    okButton.setFont(new Font("Poppins", Font.PLAIN, 15));
    cancel = new JButton("Cancel");
    cancel.setFont(new Font("Poppins", Font.PLAIN, 15));

    textField = new JTextField(20);
    textField.setFont(new Font("Poppins", Font.PLAIN, 16));

    jpanel.add(textField, BorderLayout.CENTER);
    jpanel.add(result, BorderLayout.SOUTH);
```

```
// register the Listners
SeparateActionListenerImplementer evt = new Separa
okButton.addActionListener(evt);
cancel.addActionListener(evt);

jpanel_2.add(okButton);
jpanel_2.add(cancel);

add(jpanel);
add(jpanel_2);

setLayout(new GridLayout(2, 1, 10, 10));
setTitle("Event --> Anonymous Class Demo");
setFont(new Font("Poppins", Font.PLAIN, 15));
setSize(600, 300);
setVisible(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);
}

public static void main(String[] args) {
     new SeparateMain();
}
}
```

```
class SeparateActionListenerImplementer implements ActionListener {
     SeparateMain separate;

     SeparateActionListenerImplementer(SeparateMain separate) {
            this.separate = separate;
     }

     public void actionPerformed(ActionEvent event) {
          if (event.getSource() == separate.okButton) {
             separate.textField.setText("Handling the event By Separate Class!!");
             separate.result.setText("Event has occurred!");
             }
          else if (event.getSource() == separate.cancel) {
             System.exit(0);
          }
    }
}
```



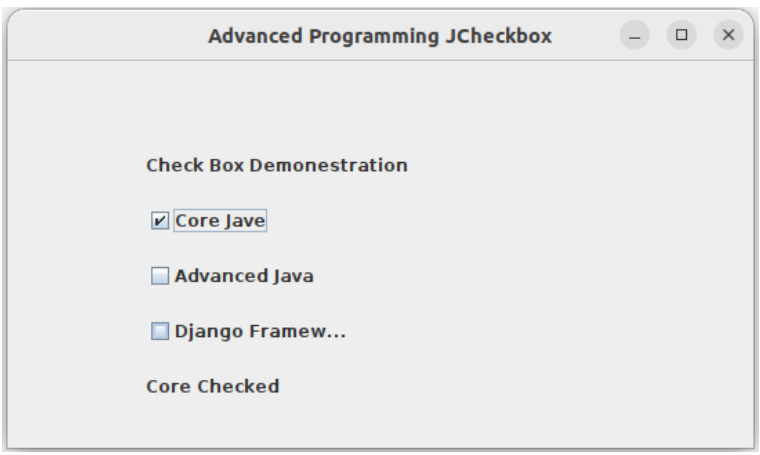


- Java MouseListener Interface
  - notified by MouseEvent when there is a change in the state of the mouse.
  - has five methods
    - public abstract void mouseClicked(MouseEvent evt);
    - public abstract void mouseEntered(MouseEvent evt);
    - public abstract void mouseExited(MouseEvent e);
    - public abstract void mousePressed(MouseEvent e);
    - public abstract void mouseReleased(MouseEvent e);
  - has plenty of methods.

Methods	Registration Methods
int getClickCount()	Returns the number of quick, consecutive clicks the user has made (including this event)
int getX(), int getY(), Point getPoint()	Return the (x,y) position at which the event occurred, relative to the component that fired the event.
int getXOnScreen(), int getYOnScreen(), int getLocationOnScreen()	• Return the absolute (x,y) position of the event. • These coordinates are relative to the virtual coordinate system for the multi-screen environment. • Otherwise, these coordinates are relative to the coordinate system associated with the Component's Graphics Configuration.
int getButton()	Returns which mouse button, if any, has a changed state. One of the following constants is returned: NOBUTTON, BUTTON1, BUTTON2, or BUTTON3.

- Java ItemListener Interface
  - is notified by ItemEvent whenever there is a change in state of Checkboxed;
  - It has only one method $\mapsto$ itemStateChanged()

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ItemListenerExample implements ItemListener {
    JCheckBox checkBox1, checkBox2, checkBox3;
    JLabel label, jLabel;
    ItemListenerExample() {
        JFrame jframe = new JFrame("Advanced Programming JCheckbox");
        JLabel label = new JLabel("Check Box Demonestration");
        label.setBounds(100, 50, 200, 50);
        checkBox1 = new JCheckBox("Core Jave");
        checkBox1.setBounds(100, 90, 150, 50);
        checkBox1.addItemListener(this);
        checkBox2 = new JCheckBox("Advanced Java");
        checkBox2.setBounds(100, 130, 150, 50);
        checkBox2.addItemListener(this);
        checkBox3 = new JCheckBox("Django Framework");
        checkBox3.setBounds(100, 170, 150, 50);
        checkBox3.addItemListener(this);
        jLabel = new JLabel();
        jLabel.setBounds(100, 210, 250, 50);
        jframe.add(label);
        jframe.add(jLabel);
```



```
jframe.add(checkBox1);
        jframe.add(checkBox2);
        jframe.add(checkBox3);
        jframe.setSize(400, 400);
        jframe.setLayout(null);
        jframe.setVisible(true);
    }
     public void itemStateChanged(ItemEvent evt) {
        if (evt.getSource() == checkBox1) {
            jLabel.setText("Core " + (evt.getStateChange() == 1 ? "Checked" : "Uncjhecked"));
        } else if (evt.getSource() == checkBox2) {
            jLabel.setText("Advanced " + (evt.getStateChange() == 1 ? "Checked" : "Unchecked"));
        } else if (evt.getSource() == checkBox3) {
            jLabel.setText("Django Framwork" + (evt.getStateChange() == 1 ? " Checked" : " Unchecked"));
        } else {
            jLabel.setText(null);
        }
    }


    public static void main(String args[]) {
         new ItemListenerExample();


    }
};
```

- Java KeyListener Interface
  - notified by KeyEvent when there is a change in the state of the key.
  - has plenty of methods.

Methods	Meaning
public abstract void keyPressed (KeyEvent e);	It is invoked when a key has been pressed.
public abstract void keyReleased (KeyEvent e);	It is invoked when a key has been released.
public abstract void keyTyped (KeyEvent e)	It is invoked when a key has been typed.

- Java KeyListener Interface
  - notified by KeyEvent when there is a change in the state of the key.
  - has plenty of methods.

Methods	Meaning
public abstract void keyPressed (KeyEvent e);	It is invoked when a key has been pressed.
public abstract void keyReleased (KeyEvent e);	It is invoked when a key has been released.
public abstract void keyTyped (KeyEvent e)	It is invoked when a key has been typed.

refer KeyListener and Other event ...

- Java Adapters
  - Adapter class**is a class in java**

    ↓

    - which is used to simplify **the process of event handling**

      ↓

      How ?

      ↓

    - provides an empty implementation of all the methods which are in the event listener interface.
- Shortly
  - Without adapter class
    - "every methods in the interface class of the Listner must be implemented!"
  - With Adapter class
    - "No need to implement manually, it will be automatically handled by the adapter class."

Adapter Classes	Listener Interface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

Done!

Question!