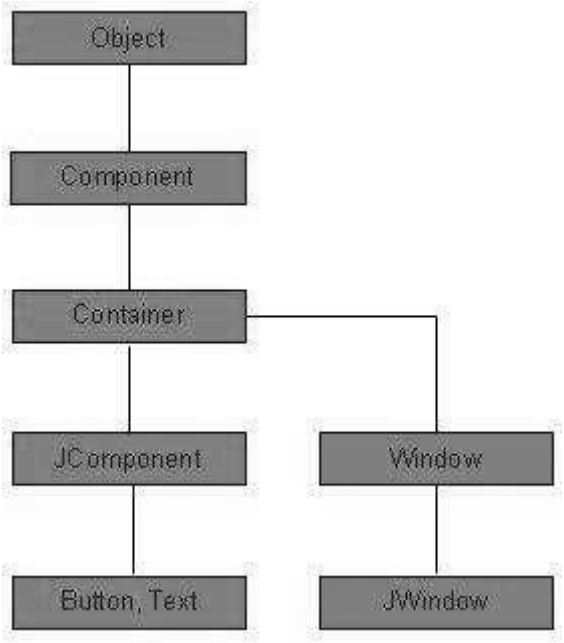
Prac 5) Write a program for creating Edit menu for Notepad using Frame

SWING :

Swing API is a set of extensible GUI Components to ease the developer's life to create JAVA based Front End/GUI Applications. It is build on top of AWT API and acts as a replacement of AWT API, since it has almost every control corresponding to AWT controls.

Every user interface considers the following three main aspects −

- **UI Elements** − These are the core visual elements the user eventually sees and interacts with. GWT provides a huge list of widely used and common elements varying from basic to complex, which we will cover in this tutorial.
- **Layouts** − They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface). This part will be covered in the Layout chapter.
- **Behavior –** These are the events which occur when the user interacts with UI elements. This part will be covered in the Event Handling chapter.



Every SWING controls inherits properties from the following Component class hiearchy.

S.No.	Class & Description
1	**Component** A Component is the abstract base class for the non menu user-interface controls of SWING. Component represents an object with graphical representation
2	**Container** A Container is a component that can contain other SWING components
3	**JComponent** A JComponent is a base class for all SWING UI components. In order to use a SWING component that inherits from JComponent, the component must be in a containment hierarchy whose root is a top-level SWING container

SWING UI Elements :

1 JLabel :

A JLabel object is a component for placing text in a container.

2 JButton :

This class creates a labeled button.

3 JCheck Box :

A JCheckBox is a graphical component that can be in either an on (true) or off (false) state.

4 JRadioButton :

The JRadioButton class is a graphical component that can be in either an on (true) or off (false) state. in a group.

5 JList :

A JList component presents the user with a scrolling list of text items.

6 JTextField :

A JTextField object is a text component that allows for the editing of a single line of text.

7 JPasswordField :

A JPasswordField object is a text component specialized for password entry.

8 JTextArea :

A JTextArea object is a text component that allows editing of a multiple lines of text.

9 JScrollbar :

A Scrollbar control represents a scroll bar component in order to enable the user to select from range of values.

10 JFileChooser :

A JFileChooser control represents a dialog window from which the user can select a file.

15 JProgressBar :

As the task progresses towards completion, the progress bar displays the task's percentage of completion.

Event Handling :

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has a code which is known as an event handler, that is executed when an event occurs.

Step 1 – The user clicks the button and the event is generated.

Step 2 – The object of concerned event class is created automatically and information about the source and the event get populated within the same object.

Step 3 – Event object is forwarded to the method of the registered listener class.

Step 4 – The method is gets executed and returns.

SWING Event Classes:

1 AWTEvent

It is the root event class for all SWING events. This class and its subclasses supercede the original java.awt.Event class.

2 ActionEvent

The ActionEvent is generated when the button is clicked or the item of a list is double-clicked.

3 InputEvent

The InputEvent class is the root event class for all component-level input events.

4 KeyEvent

On entering the character the Key event is generated.

5 MouseEvent

This event indicates a mouse action occurred in a component.

6 WindowEvent

The object of this class represents the change in the state of a window.

7 MouseMotionEvent

The object of this class represents the change in the state of a window.

SWING – Layouts :

Layout refers to the arrangement of components within the container.

The layout manager automatically positions all the components within the container. Even if you do not use the layout manager, the components are still positioned by the default layout manager.

1 BorderLayout

The borderlayout arranges the components to fit in the five regions: east, west, north, south, and center.

2 CardLayout

The CardLayout object treats each component in the container as a card. Only one card is visible at a time.

3 FlowLayout

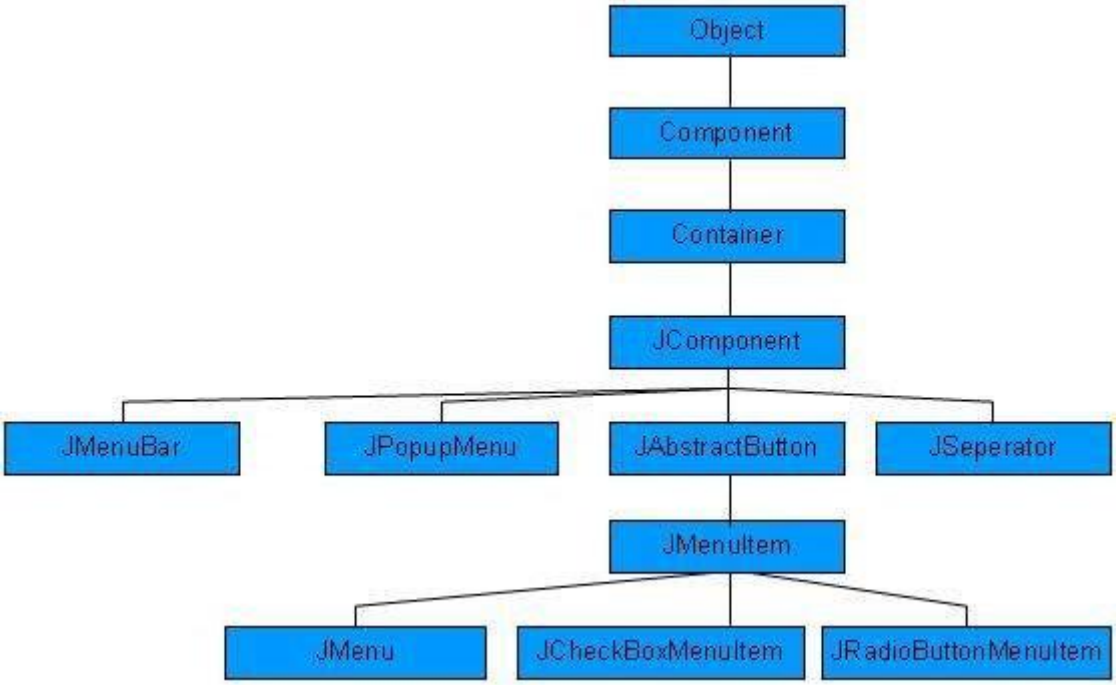
The FlowLayout is the default layout. It layout the components in a directional flow.

4 GridLayout

The GridLayout manages the components in the form of a rectangular grid.

SWING - Menu Classes

Every top-level window has a menu bar associated with it. This menu bar consists of various menu choices available to the end user. Further, each choice contains a list of options, which is called drop-down menus. Menu and MenuItem controls are subclass of MenuComponent class.



Menu Controls :

1 JMenuBar

The JMenuBar object is associated with the top-level window.

2 JMenuItem

The items in the menu must belong to the JMenuItem or any of its subclass.

3 JMenu

The JMenu object is a pull-down menu component which is displayed from the menu bar.

4 JCheckboxMenuItem

JCheckboxMenuItem is the subclass of JMenuItem.

5 JRadioButtonMenuItem

JRadioButtonMenuItem is the subclass of JMenuItem.

6 JPopupMenu

JPopupMenu can be dynamically popped up at a specified position within a component.

SWING – Containers :

Containers are an integral part of SWING GUI components. A container provides a space where a component can be located. A Container in AWT is a component itself and it provides the capability to add a component to itself. Following are certain noticable points to be considered.

- Sub classes of Container are called as Container. For example, JPanel, JFrame and JWindow.
- Container can add only a Component to itself.
- A default layout is present in each container which can be overridden using setLayout method.

1. Panel

   JPanel is the simplest container. It provides space in which any other component can be placed, including other panels.

2. Frame

   A JFrame is a top-level window with a title and a border.

3. Window

   A JWindow object is a top-level window with no borders and no menubar.

CODE :

Notepad.java

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

public class Notepad extends JFrame implements ActionListener {

    JTextArea textArea;
    JMenuBar menuBar;
    JMenu editMenu;
    JMenuItem cutItem, copyItem, pasteItem, selectAllItem;

    public Notepad() {
        setTitle("Simple Notepad");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        textArea = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(textArea);
        add(scrollPane, BorderLayout.CENTER);

        menuBar = new JMenuBar();

        editMenu = new JMenu("Edit");
```

```
        cutItem = new JMenuItem("Cut");
        copyItem = new JMenuItem("Copy");
        pasteItem = new JMenuItem("Paste");
        selectAllItem = new JMenuItem("Select All");

        editMenu.add(cutItem);
        editMenu.add(copyItem);
        editMenu.add(pasteItem);
        editMenu.addSeparator();
        editMenu.add(selectAllItem);

        menuBar.add(editMenu);

        setJMenuBar(menuBar);

        cutItem.addActionListener(this);
        copyItem.addActionListener(this);
        pasteItem.addActionListener(this);
        selectAllItem.addActionListener(this);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        switch (command) {
            case "Cut":
                textArea.cut();
                break;
            case "Copy":
                textArea.copy();
                break;
            case "Paste":
                textArea.paste();
                break;
            case "Select All":
                textArea.selectAll();
                break;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new Notepad());
    }
}
```