## Introduction

The Calendar Application is a user-friendly desktop application built using Java Swing. It allows users to view a calendar, navigate through different months and years, and optionally add events or notes for specific dates.

## Components

1. **Main Frame**: The main window that holds the calendar.
2. **Calendar Panel**: The panel displaying the calendar grid.
3. **Navigation Panel**: Controls for navigating between months and years.
4. **Event Management**: Optional feature to add, view, and manage events.

## Basic Implementation

### 1. Setup the Main Frame

First, create the main frame that will hold all the components.

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

public class CalendarApp extends JFrame {

    public CalendarApp() {
        setTitle("Calendar Application");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        initUI();
    }

    private void initUI() {
        // Create and add components
        CalendarPanel calendarPanel = new CalendarPanel();
        NavigationPanel navigationPanel = new NavigationPanel(calendarPanel);

        add(navigationPanel, BorderLayout.NORTH);
        add(calendarPanel, BorderLayout.CENTER);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            CalendarApp app = new CalendarApp();
            app.setVisible(true);
        });
    }
}
```

## 2. Create the Calendar Panel

The CalendarPanel will display the days of the month in a grid layout.

```
java
Copy code
import javax.swing.*;
import java.awt.*;
import java.util.Calendar;

public class CalendarPanel extends JPanel {
    private JLabel[][] dayLabels;
    private Calendar calendar;

    public CalendarPanel() {
        calendar = Calendar.getInstance();
        setLayout(new GridLayout(7, 7));  // 7 days a week, 7 rows

        dayLabels = new JLabel[7][7];
        String[] daysOfWeek = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
"Sat"};

        // Add days of the week labels
        for (int i = 0; i < 7; i++) {
            dayLabels[0][i] = new JLabel(daysOfWeek[i],
SwingConstants.CENTER);
            add(dayLabels[0][i]);
        }

        // Initialize day labels
        for (int i = 1; i < 7; i++) {
            for (int j = 0; j < 7; j++) {
                dayLabels[i][j] = new JLabel("", SwingConstants.CENTER);
                add(dayLabels[i][j]);
            }
        }

        updateCalendar();
    }

    public void updateCalendar() {
        calendar.set(Calendar.DAY_OF_MONTH, 1);
        int firstDayOfWeek = calendar.get(Calendar.DAY_OF_WEEK) - 1;
        int daysInMonth = calendar.getActualMaximum(Calendar.DAY_OF_MONTH);

        // Clear previous labels
        for (int i = 1; i < 7; i++) {
            for (int j = 0; j < 7; j++) {
                dayLabels[i][j].setText("");
            }
        }

        // Fill in days
        int row = 1;
        int column = firstDayOfWeek;
        for (int day = 1; day <= daysInMonth; day++) {
```

```
            dayLabels[row][column].setText(String.valueOf(day));
            column++;
            if (column == 7) {
                column = 0;
                row++;
            }
        }
    }

    public void setMonth(int month) {
        calendar.set(Calendar.MONTH, month);
        updateCalendar();
    }

    public void setYear(int year) {
        calendar.set(Calendar.YEAR, year);
        updateCalendar();
    }
}
```

### 3. Create the Navigation Panel

The NavigationPanel allows the user to navigate through different months and years.

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

public class NavigationPanel extends JPanel {
    private CalendarPanel calendarPanel;
    private JComboBox<String> monthComboBox;
    private JComboBox<Integer> yearComboBox;

    public NavigationPanel(CalendarPanel calendarPanel) {
        this.calendarPanel = calendarPanel;
        setLayout(new FlowLayout());

        // Month ComboBox
        String[] months = {
            "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "November", "December"
        };
        monthComboBox = new JComboBox<>(months);
        monthComboBox.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int month = monthComboBox.getSelectedIndex();
                calendarPanel.setMonth(month);
            }
        });
        add(monthComboBox);

        // Year ComboBox
```

```
        yearComboBox = new JComboBox<>();
        for (int i = 1900; i <= 2100; i++) {
            yearComboBox.addItem(i);
        }
        yearComboBox.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int year = (int) yearComboBox.getSelectedItem();
                calendarPanel.setYear(year);
            }
        });
        add(yearComboBox);
    }
}
```

## Optional: Event Management

To add event management, you can create a separate dialog for adding and viewing events. This part can be expanded based on specific requirements.

## Conclusion

This is a basic outline and implementation of a Calendar Application using Java Swing. You can expand this by adding features like event management, persistent storage, and more sophisticated UI elements.