**Calendar Implementation**

By

**Sudeep Raj**

**12110605**

**Batch Year-2021**

Under the guidance of

**Dr. Bhimasen Moharana**

Logo, company name

Description automatically generated

#### 

#### School of Computer Science and Engineering

Lovely Professional University Phagwara, Punjab (India) Month: April Year:2023

Title: Java Calendar Java Project

**Abstract**

**The aim of this project was to develop a calendar application using java programming language. The application enables users to view, add, and edit events and appointments, as well as set reminders for upcoming events. The project utilises Java’s built in libraries for date and time, and employs the Model View Controller(MVC) design pattern to ensure the application is maintainable and scalable.**

**Introduction**

**The calendar application is a widely used tool in today’s world,Whether for personal or professional purposes. With the advancement of technology, the traditional paper- based calendar has been replaced by digital ones. Java is a popular programming language that provides a rich set of libraries for a date and time manipulation making it an ideal choice for developing a calendar**

**Application.**

**Design**

**The project was designed using the MVC pattern, which separates the application into three distinct components : Model, View and Controller. The Model component encapsulates the data and logic of the application, the View component is responsible for displaying the data to the user , and the controller component acts as the intermediatory between Model and View accordingly.**

**Implementation**

**The application was implemented in Java, utilizing the Java Swing library for the graphical user interface. The model component was implemented using Java’s built in date and time libraries, while the Controller Component was implemented using event listeners. The View Component was implemented using various Swing Components, such as JTable, JDialog, and Jlabel.**

**Result**

**The calendar application was successfully developed and tested, meting all of the project requirements. The application allows users to view,add and edit events and appointments as well as set reminders for upcoming events. The MVC design pattern ensured that the application was maintainable and scalable and the use of Java’s built in libraries reduce the amount of custom code needed .**

Java Program Implementation

import javax.swing.\*;

import javax.swing.event.\*;

import javax.swing.table.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

public class CalendarProgram{

static JLabel lblMonth, lblYear;

static JButton btnPrev, btnNext;

static JTable tblCalendar;

static JComboBox cmbYear;

static JFrame frmMain;

static Container pane;

static DefaultTableModel mtblCalendar; //Table model

static JScrollPane stblCalendar; //The scrollpane

static JPanel pnlCalendar;

static int realYear, realMonth, realDay, currentYear, currentMonth;

public static void main (String args[]){

//Look and feel

try {UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());}

catch (ClassNotFoundException e) {}

catch (InstantiationException e) {}

catch (IllegalAccessException e) {}

catch (UnsupportedLookAndFeelException e) {}

//Prepare frame

frmMain = new JFrame ("Gestionnaire de clients"); //Create frame

frmMain.setSize(330, 375); //Set size to 400x400 pixels

pane = frmMain.getContentPane(); //Get content pane

pane.setLayout(null); //Apply null layout

frmMain.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //Close when X is clicked

//Create controls

lblMonth = new JLabel ("January");

lblYear = new JLabel ("Change year:");

cmbYear = new JComboBox();

btnPrev = new JButton ("&lt;&lt;");

btnNext = new JButton ("&gt;&gt;");

mtblCalendar = new DefaultTableModel(){public boolean isCellEditable(int rowIndex, int mColIndex){return false;}};

tblCalendar = new JTable(mtblCalendar);

stblCalendar = new JScrollPane(tblCalendar);

pnlCalendar = new JPanel(null);

//Set border

pnlCalendar.setBorder(BorderFactory.createTitledBorder("Calendar"));

//Register action listeners

btnPrev.addActionListener(new btnPrev\_Action());

btnNext.addActionListener(new btnNext\_Action());

cmbYear.addActionListener(new cmbYear\_Action());

//Add controls to pane

pane.add(pnlCalendar);

pnlCalendar.add(lblMonth);

pnlCalendar.add(lblYear);

pnlCalendar.add(cmbYear);

pnlCalendar.add(btnPrev);

pnlCalendar.add(btnNext);

pnlCalendar.add(stblCalendar);

//Set bounds

pnlCalendar.setBounds(0, 0, 320, 335);

lblMonth.setBounds(160-lblMonth.getPreferredSize().width/2, 25, 100, 25);

lblYear.setBounds(10, 305, 80, 20);

cmbYear.setBounds(230, 305, 80, 20);

btnPrev.setBounds(10, 25, 50, 25);

btnNext.setBounds(260, 25, 50, 25);

stblCalendar.setBounds(10, 50, 300, 250);

//Make frame visible

frmMain.setResizable(false);

frmMain.setVisible(true);

//Get real month/year

GregorianCalendar cal = new GregorianCalendar(); //Create calendar

realDay = cal.get(GregorianCalendar.DAY\_OF\_MONTH); //Get day

realMonth = cal.get(GregorianCalendar.MONTH); //Get month

realYear = cal.get(GregorianCalendar.YEAR); //Get year

currentMonth = realMonth; //Match month and year

currentYear = realYear;

//Add headers

String[] headers = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"}; //All headers

for (int i=0; i<7; i++){

mtblCalendar.addColumn(headers[i]);

}

tblCalendar.getParent().setBackground(tblCalendar.getBackground()); //Set background

//No resize/reorder

tblCalendar.getTableHeader().setResizingAllowed(false);

tblCalendar.getTableHeader().setReorderingAllowed(false);

//Single cell selection

tblCalendar.setColumnSelectionAllowed(true);

tblCalendar.setRowSelectionAllowed(true);

tblCalendar.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

//Set row/column count

tblCalendar.setRowHeight(38);

mtblCalendar.setColumnCount(7);

mtblCalendar.setRowCount(6);

//Populate table

for (int i=realYear-100; i<=realYear+100; i++){

cmbYear.addItem(String.valueOf(i));

}

//Refresh calendar

refreshCalendar (realMonth, realYear); //Refresh calendar

}

public static void refreshCalendar(int month, int year){

//Variables

String[] months = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

int nod, som; //Number Of Days, Start Of Month

//Allow/disallow buttons

btnPrev.setEnabled(true);

btnNext.setEnabled(true);

if (month == 0 && year <= realYear-10){btnPrev.setEnabled(false);} //Too early

if (month == 11 && year >= realYear+100){btnNext.setEnabled(false);} //Too late

lblMonth.setText(months[month]); //Refresh the month label (at the top)

lblMonth.setBounds(160-lblMonth.getPreferredSize().width/2, 25, 180, 25); //Re-align label with calendar

cmbYear.setSelectedItem(String.valueOf(year)); //Select the correct year in the combo box

//Clear table

for (int i=0; i<6; i++){

for (int j=0; j<7; j++){

mtblCalendar.setValueAt(null, i, j);

}

}

//Get first day of month and number of days

GregorianCalendar cal = new GregorianCalendar(year, month, 1);

nod = cal.getActualMaximum(GregorianCalendar.DAY\_OF\_MONTH);

som = cal.get(GregorianCalendar.DAY\_OF\_WEEK);

//Draw calendar

for (int i=1; i<=nod; i++){

int row = new Integer((i+som-2)/7);

int column = (i+som-2)%7;

mtblCalendar.setValueAt(i, row, column);

}

//Apply renderers

tblCalendar.setDefaultRenderer(tblCalendar.getColumnClass(0), new tblCalendarRenderer());

}

static class tblCalendarRenderer extends DefaultTableCellRenderer{

public Component getTableCellRendererComponent (JTable table, Object value, boolean selected, boolean focused, int row, int column){

super.getTableCellRendererComponent(table, value, selected, focused, row, column);

if (column == 0 || column == 6){ //Week-end

setBackground(new Color(255, 220, 220));

}

else{ //Week

setBackground(new Color(255, 255, 255));

}

if (value != null){

if (Integer.parseInt(value.toString()) == realDay && currentMonth == realMonth && currentYear == realYear){ //Today

setBackground(new Color(220, 220, 255));

}

}

setBorder(null);

setForeground(Color.black);

return this;

}

}

static class btnPrev\_Action implements ActionListener{

public void actionPerformed (ActionEvent e){

if (currentMonth == 0){ //Back one year

currentMonth = 11;

currentYear -= 1;

}

else{ //Back one month

currentMonth -= 1;

}

refreshCalendar(currentMonth, currentYear);

}

}

static class btnNext\_Action implements ActionListener{

public void actionPerformed (ActionEvent e){

if (currentMonth == 11){ //Foward one year

currentMonth = 0;

currentYear += 1;

}

else{ //Foward one month

currentMonth += 1;

}

refreshCalendar(currentMonth, currentYear);

}

}

static class cmbYear\_Action implements ActionListener{

public void actionPerformed (ActionEvent e){

if (cmbYear.getSelectedItem() != null){

String b = cmbYear.getSelectedItem().toString();

currentYear = Integer.parseInt(b);

refreshCalendar(currentMonth, currentYear);

}

}

}

}

DFD for Calendar



In this DFD the users interacts with the calendar through the User Interface which allows them to view,

add,edit and delete events. The user interface sends request to the Event Manager which manages the event and their associated data.

The database may also send data back to the Event

Manager which can then update the user interface as necessary. The user interface may also sends request to the Event Manager to update the display.

Overall, this DFD shows a simple flow of data between user interface, the event manager, and the database which allows for the creation,management and storage of calendar events.

E-R Diagram of Calendar

