EASYSCHED : A USER-FRIEND MANUAL SCHEDULER

Final Laboratory Project
Submitted to the Faculty of the
Department of Computer Studies
Cavite State University
Bacoor City Campus
Bacoor, Cavite

In Partial Fulfillment
of the requirements for Course of
DCIT50: Object Oriented Programming

RANCE GABRIELLE G. SIROY
MARC PITON D. EBREO
SEAN JAMES T. GALINDO
MARK COMENDADOR
January 2025

DCIT50FL2025-

EASYSCHED : A USER-FRIEND MANUAL SCHEDULER

RANCE GABRIELLE G. SIROY
MARC PITON D. EBREO
SEAN JAMES T. GALINDO
MARK COMENDADOR

A final project manuscript submitted to the faculty of the Department of Computer Studies, Cavite State University – Bacoor City Campus, City of Bacoor, Cavite in partial fulfilment of the requirements for Course of DCIT50: Object Oriented Programming with Abstract No. ____. Prepared under the supervision of Mr. Julios M. Mojas.

INTRODUCTION

Abstract

EasySched is a Java-based personal planner that helps people organize their work and time efficiently. This program has a calendar and to-do list feature, giving users an easy way to manage their daily and monthly tasks. The system was created using Java Swing and follows an object-oriented programming style, which ensures modularity and scalability. The calendar section highlights the selected day, while the to-do list module allows users to add, update, and mark activities as done. The development method was iterative, with phases including requirement analysis, design, implementation, and testing. Aligning components within the UI and updating dynamic tasks were among the challenges. Future enhancements include notifications, integration with Google Calendar, and theme customization.

Keywords: Java Swing; Calendar Functionality; To-Do List; Personal Scheduler; Task Management

I. Introduction

Background

As people look for better ways to manage their time and activities, the need for effective scheduling solutions is increasing. Organizing daily and monthly activities is a challenge for many, which results in lost opportunities and lower production. EasySched was created to solve these issues. This personal scheduler, which is built on Java, offers a smooth and user-friendly platform for planning and tracking tasks. By combining calendar and to-do list features, the system provides a useful solution that is suited to busy people's requirements.

Research Problem

Despite the availability of numerous scheduling tools, many of them lack designs that are easy to use and accessible. They are therefore inappropriate for non-technical users who need simple yet effective work management. In order to address this problem, EasySched offers a useful and aesthetically pleasing scheduler that prioritizes utility and simplicity, making it an invaluable tool for anyone looking to improve their time management.

Objectives

The development of EasySched focuses on using Java Swing to create a personal planner that integrates to-do list and calendar features. It displays a modular design that enables future scaling and improvements, and it strives to provide an intuitive user interface for effective task management. By fulfilling these goals, the study advances the creation of user-friendly productivity solutions that serve a range of users.

Significance

Anyone looking for an efficient task management tool, including professionals and students, will benefit greatly from EasySched's ability to increase productivity. Time management is improved and tension is decreased by the application's simplification of organizational procedures. Better scheduling techniques are also promoted by its design, which eventually helps people in both their personal and professional lives.

II. Methods

Research Design

In order to design, develop, and assess EasySched as a stand-alone scheduling tool, this study used a developmental research methodology. Modularity and usability were prioritized in order to guarantee that the system satisfied the requirements of its intended users..

Participants/Respondents

The intended users of EasySched include students and professionals who provided valuable feedback during the testing phase. Their input helped refine the system's features and improve its overall usability.

Procedures

To determine the fundamental features of a scheduler, a needs analysis was conducted before the development process started. These results served as the basis for designing the system architecture and defining its structure using UML diagrams. Following an object-oriented programming methodology, Java Swing was used for feature integration and coding during the implementation phase. Following development, user testing was done to get input on the application's usability and functionality. To make sure the application achieved its goals, the review step compared it to predetermined standards.

Tools and Technologies

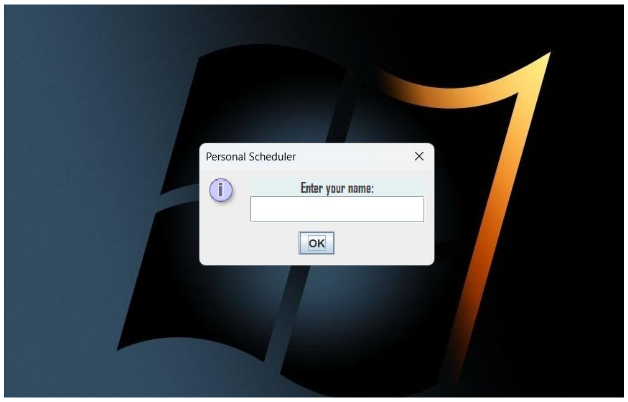
Java was used as the programming language in the Eclipse IDE development environment for the creation of EasySched. Javax.swing, Java.awt, and Java.util were important libraries..

III. Results

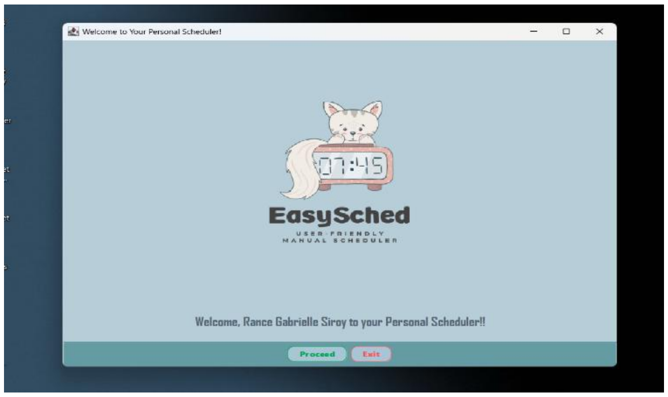
Development Output

By producing a useful and intuitive scheduling program, EasySched's development effectively met its goals. With its interactive day buttons and monthly view, the calendar module lets users choose a day while automatically resetting the ones they've already chosen. This guarantees lucidity and minimizes misunderstanding. With each task containing information like its name, time, and related day, the to-do list module facilitates task addition, modifications, and completion tracking. Task management is made thorough and effective by these qualities.

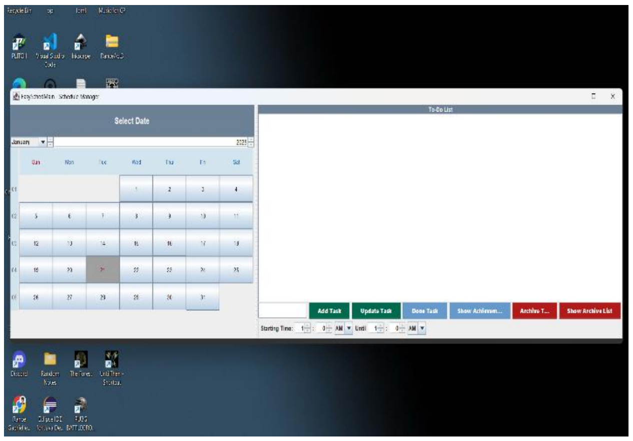
Screenshots:

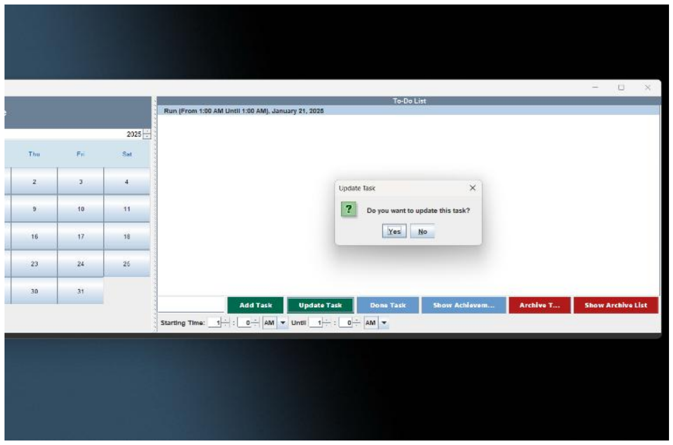

Personal Scheduler

Enter your name:


Welcome to Your Personal Scheduler!

Welcome, Rance Gabrielle Siroy to your Personal Scheduler!!

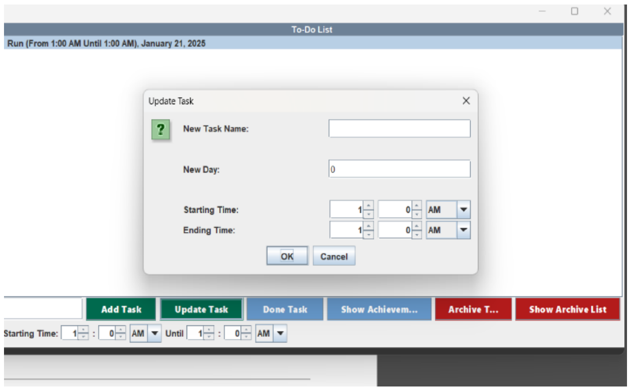



To-Do List

Run (From 1:00 AM Until 1:00 AM), January 21, 2025

Update Task

Do you want to update this task?


To-Do List

Run (From 1:00 AM Until 1:00 AM), January 21, 2025

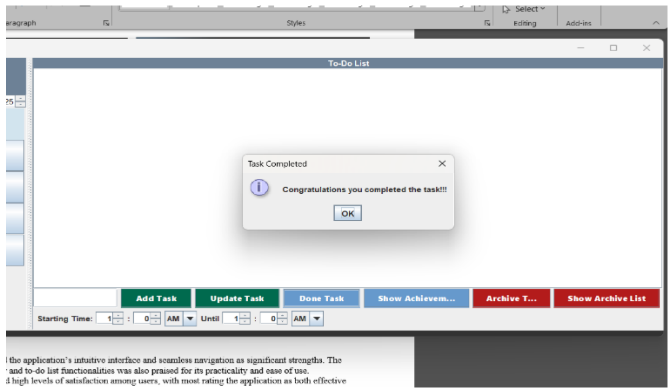
Update Task

New Task Name:

New Day: 0

Starting Time: 1 : 0 AM

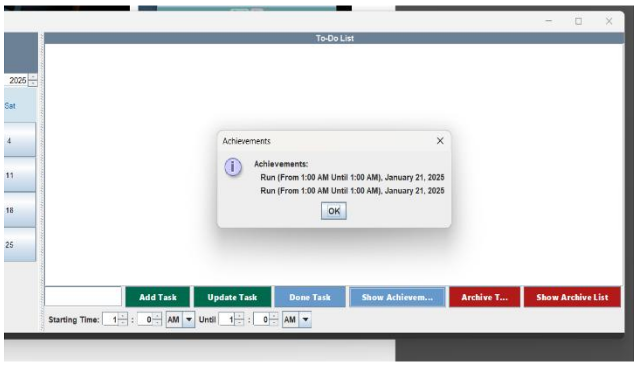
Ending Time: 1 : 0 AM


To-Do List

Task Completed

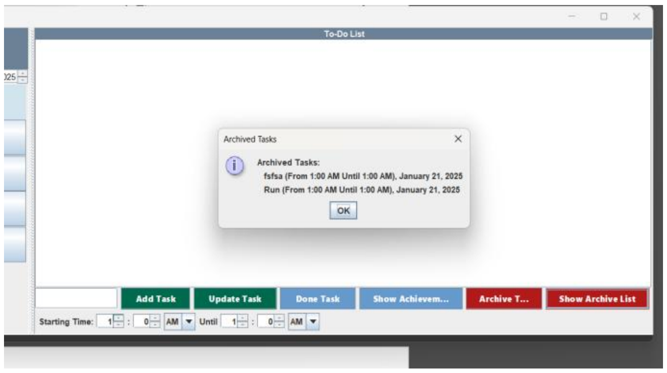
Congratulations you completed the task!!!

the application's intuitive interface and seamless navigation as significant strengths. The and to-do list functionalities was also praised for its practicality and ease of use. d high levels of satisfaction among users, with most rating the application as both effective


To-Do List

Achievements

Achievements:
Run (From 1:00 AM Until 1:00 AM), January 21, 2025
Run (From 1:00 AM Until 1:00 AM), January 21, 2025


To-Do List

Archived Tasks

Archived Tasks:
fsfsa (From 1:00 AM Until 1:00 AM), January 21, 2025
Run (From 1:00 AM Until 1:00 AM), January 21, 2025

Evaluation Findings

The application's smooth navigation and user-friendly UI were cited by users as key advantages. The practicality and use of the calendar and to-do list integration were also commended. According to evaluation data, most users rated the program as both effective and user-friendly, indicating high levels of user satisfaction.

IV. Discussion

Interpret Findings

The results show that EasySched successfully tackles the difficulties related to time and task management. For customers looking for an accessible scheduling tool, its user-friendly design and integration of key functions, like calendar and to-do list modules, offer a solid answer. The system's emphasis on modularity ensures its ongoing relevance and usage by enabling future improvements.

Comparison

Compared to manual scheduling techniques like paper-based organizers, EasySched has a number of advantages. It is a more effective way to manage activities and plans because of its interactive features, automated tracking, and configurable settings, which cut down on the time and effort needed for planning.

Strengths and Limitations

EasySched's robust task-tracking features, flexible design, and user-friendly interface are among its advantages. However, mobility is limited by its desktop-only application, and multi-device access is restricted by the absence of cloud synchronization. These drawbacks point out areas that can be enhanced in subsequent versions.

Recommendations

Future iterations of EasySchedule might include Google Calendar connectivity, customized themes and layouts, and notifications for impending chores to improve its usefulness. These features would increase the application's appeal to a larger audience and enhance the user experience.

V. Conclusion

Summarize Key Points

A Java-based personal scheduler called EasySched was created in response to the demand for effective task management solutions. The application effectively plans daily and monthly tasks by fusing an easy-to-use UI with built-in calendar and to-do list features. Scalability and adaptability for upcoming improvements are guaranteed by its modular architecture.

Closing Statement

EasySched's creation emphasizes how crucial user-centric design is for productivity applications. It advances the more general objective of increasing productivity and encouraging improved time management techniques by offering an easily accessible and efficient scheduling solution.

VI. SOURCE CODE

```
package Packs;
import com.toedter.calendar.JCalendar;


import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.*;
import java.awt.*;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.File;
import java.util.Calendar;


class CalendarFunc extends EasySchedFunc {
   private JPanel panel;
   private JCalendar calendar;
   private int selectedDay;
   private int selectedMonth;
   private int selectedYear;
   private CalendarListener listener;


   public CalendarFunc() {
          Calendar today = Calendar.getInstance();
          this.selectedDay = today.get(Calendar.DAY_OF_MONTH);
          this.selectedMonth = today.get(Calendar.MONTH);
          this.selectedYear = today.get(Calendar.YEAR);


   }


    @Override
   public void initialize() {
      panel = new JPanel(new BorderLayout());
      panel = new JPanel();
      panel.setBackground(new Color(109, 129, 150));
      panel.setLayout(new BorderLayout());

      calendar = new JCalendar();
      calendar.setPreferredSize(new Dimension(100, 100));
      calendar.setBackground(new Color(230, 241, 242));
      calendar.setForeground(Color.BLACK);

      calendar.addPropertyChangeListener(new PropertyChangeListener() {
        @Override
        public void propertyChange(PropertyChangeEvent evt) {
          if (evt.getPropertyName().equals("calendar")) {
            Calendar selectedCalendar = calendar.getCalendar();
            selectedDay = selectedCalendar.get(Calendar.DAY_OF_MONTH);
            selectedMonth = selectedCalendar.get(Calendar.MONTH);
            selectedYear = selectedCalendar.get(Calendar.YEAR);

            if (listener != null) {
              listener.onDaySelected(selectedDay);
              try {
                Clip clip = AudioSystem.getClip();
                clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\Speech
Misrecognition.wav")));
                clip.start();
              } catch (Exception ex) {
                ex.printStackTrace();
              }

            }
          }
        }
```

```
        });

        JLabel selectDayLabel = new JLabel("Select Date", JLabel.CENTER);
        selectDayLabel.setFont(new Font("Arial", Font.BOLD, 16));
        selectDayLabel.setForeground(Color.WHITE);
        selectDayLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));

        panel.add(selectDayLabel, BorderLayout.NORTH);
        panel.add(calendar, BorderLayout.CENTER);



    }

    public int getSelectedDay() {
        return selectedDay;
    }

    public void setCalendarListener(CalendarListener listener) {
        this.listener = listener;
    }

    public interface CalendarListener {
        void onDaySelected(int day);
    }

    public JPanel getPanel() {
        return panel;
    }

    public int getSelectedMonth() {
        return selectedMonth;
    }
    public int getSelectedYear() {
        return selectedYear;
    }

}

package Packs;

import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.*;
import java.util.List;
import java.util.ArrayList;

class ToDoListFunc extends EasySchedFunc {
    private JPanel panel;
    private DefaultListModel<String> listModel;
    private JList<String> list;
    private JTextField input;
    private JButton addButton = new JButton("Add Task");
    private JButton updateButton = new JButton("Update Task");
    private Map<Integer, Map<Integer, List<Task>>> tasksByMonth;
    private CalendarFunc calendarComponent;
    private List<Task> archivedTasks = new ArrayList<>();
    private JButton archiveButton = new JButton("Archive Task");
    private JButton showArchiveButton = new JButton("Show Archive List");
    private JFrame frame;
    private DefaultListModel<String> taskListModel;
    private JList<String> taskList;

    private ArrayList<String> archives;
    private List<String> achievements = new ArrayList<>();




    public ToDoListFunc(CalendarFunc calendarComponent) {
        this.calendarComponent = calendarComponent;
        this.tasksByMonth = new HashMap<>();
        frame = new JFrame("To-Do List");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 400);




        this.calendarComponent.setCalendarListener(new CalendarFunc.CalendarListener() {
            @Override
            public void onDaySelected(int day) {
                refreshTasksForSelectedMonth();
            }
        });
    }


    @Override
```

```
public void initialize() {
    panel = new JPanel(new BorderLayout());
    listModel = new DefaultListModel<>();
    list = new JList<>(listModel);
    input = new JTextField();
    addButton = new JButton("Add Task");
    updateButton = new JButton("Update Task");
    showArchiveButton = new JButton("Show Archive List");
    taskListModel = new DefaultListModel<>();
    taskList = new JList<>(taskListModel);
    achievements = new ArrayList<>();
    archives = new ArrayList<>();



    JButton addButton = new JButton("Add Task");
    addButton.setBackground(new Color(0, 106, 78));
    addButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
    addButton.setForeground(Color.WHITE);
    addButton.setFocusPainted(false);
    addButton.setPreferredSize(new Dimension(100, 30));

    JButton updateButton = new JButton("Update Task");
    updateButton.setBackground(new Color(0, 106, 78));
    updateButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
    updateButton.setForeground(Color.WHITE);
    updateButton.setFocusPainted(false);
    updateButton.setPreferredSize(new Dimension(120, 30));
```

```
panel = new JPanel();
panel.setBackground(new Color(109, 129, 150));
panel.setLayout(new BorderLayout());

JScrollPane scrollPane = new JScrollPane(taskList);
JButton doneButton = new JButton("Done Task");
doneButton.setBackground(new Color(102, 153, 204));
doneButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
doneButton.setForeground(Color.WHITE);
doneButton.setFocusPainted(false);
doneButton.setPreferredSize(new Dimension(110, 30));

JButton showAchievementsButton = new JButton("Show Achievement List");
showAchievementsButton.setBackground(new Color(102, 153, 204));
showAchievementsButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
showAchievementsButton.setForeground(Color.WHITE);
showAchievementsButton.setFocusPainted(false);
showAchievementsButton.setPreferredSize(new Dimension(150, 30));

JButton archiveButton = new JButton("Archive Task");
archiveButton.setBackground(new Color(179, 27, 27));
archiveButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
archiveButton.setForeground(Color.WHITE);
archiveButton.setFocusPainted(false);
archiveButton.setPreferredSize(new Dimension(110, 30));

JButton showArchiveButton = new JButton("Show Archive List");
showArchiveButton.setBackground(new Color(179, 27, 27));
showArchiveButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
showArchiveButton.setForeground(Color.WHITE);
showArchiveButton.setFocusPainted(false);
showArchiveButton.setPreferredSize(new Dimension(150, 30));
```

```
JTextArea toDoListTextArea = new JTextArea();
toDoListTextArea.setBackground(new Color(109, 129, 150));
toDoListTextArea.setForeground(Color.WHITE);
toDoListTextArea.setLineWrap(true);
toDoListTextArea.setWrapStyleWord(true);
toDoListTextArea.setBorder(BorderFactory.createEmptyBorder(60, 60, 60, 60));



SpinnerNumberModel startHourModel = new SpinnerNumberModel(1, 1, 12, 1);
JSpinner startHourSpinner = new JSpinner(startHourModel);

SpinnerNumberModel startMinuteModel = new SpinnerNumberModel(0, 0, 59, 1);
JSpinner startMinuteSpinner = new JSpinner(startMinuteModel);

JComboBox<String> startAmPmSelector = new JComboBox<>(new String[]{"AM", "PM"});

SpinnerNumberModel endHourModel = new SpinnerNumberModel(1, 1, 12, 1);
JSpinner endHourSpinner = new JSpinner(endHourModel);

SpinnerNumberModel endMinuteModel = new SpinnerNumberModel(0, 0, 59, 1);
JSpinner endMinuteSpinner = new JSpinner(endMinuteModel);

JComboBox<String> endAmPmSelector = new JComboBox<>(new String[]{"AM", "PM"});



JComboBox<String> monthSelector = new JComboBox<>(new String[]{"January", "February",
"March", "April", "May", "June",
                    "July", "August", "September", "October", "November",
"December"});
```

```
JTextField dayField = new JTextField(2);
JTextField yearField = new JTextField(4);

JPanel timePanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
timePanel.add(new JLabel("Starting Time:"));
timePanel.add(startHourSpinner);
timePanel.add(new JLabel(":"));
timePanel.add(startMinuteSpinner);
timePanel.add(startAmPmSelector);

timePanel.add(new JLabel("Until"));

timePanel.add(endHourSpinner);
timePanel.add(new JLabel(":"));
timePanel.add(endMinuteSpinner);
timePanel.add(endAmPmSelector);

JPanel datePanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
datePanel.add(new JLabel("Month:"));
datePanel.add(monthSelector);
datePanel.add(new JLabel("Day:"));
datePanel.add(dayField);
datePanel.add(new JLabel("Year:"));
datePanel.add(yearField);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 5, 0));
buttonPanel.add(addButton);
buttonPanel.add(updateButton);
buttonPanel.add(doneButton);
buttonPanel.add(showAchievementsButton);
buttonPanel.add(archiveButton);
buttonPanel.add(showArchiveButton);
```

```
frame.add(scrollPane, BorderLayout.CENTER);


frame.add(buttonPanel, BorderLayout.SOUTH);
panel.add(buttonPanel, BorderLayout.NORTH);




JPanel inputPanel = new JPanel(new BorderLayout());
inputPanel.add(input, BorderLayout.CENTER);
inputPanel.add(buttonPanel, BorderLayout.EAST);
frame.add(inputPanel, BorderLayout.NORTH);

JPanel southPanel = new JPanel(new BorderLayout());
southPanel.add(inputPanel, BorderLayout.NORTH);
southPanel.add(timePanel, BorderLayout.SOUTH);



JLabel toDoListLabel = new JLabel("To-Do List", JLabel.CENTER);
toDoListLabel.setForeground(Color.WHITE);
panel.add(toDoListLabel, BorderLayout.NORTH);
panel.add(new JScrollPane(list), BorderLayout.CENTER);
panel.add(southPanel, BorderLayout.SOUTH);




addButton.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    int selectedMonth = calendarComponent.getSelectedMonth();
    int selectedDay = calendarComponent.getSelectedDay();
    int selectedYear = calendarComponent.getSelectedYear();
```

```
        if (selectedMonth == -1 || selectedDay == -1 || selectedYear == -1) {
            JOptionPane.showMessageDialog(panel, "Please select a date first!", "No Date Selected",
JOptionPane.WARNING_MESSAGE);
            return;
        }

        String taskName = input.getText().trim();
        if (taskName.isEmpty()) {
            JOptionPane.showMessageDialog(panel, "Task name cannot be empty!", "Empty Task",
JOptionPane.WARNING_MESSAGE);
            return;
        }

        int startHour = (int) startHourSpinner.getValue();
        int startMinute = (int) startMinuteSpinner.getValue();
        String startAmPm = (String) startAmPmSelector.getSelectedItem();

        int endHour = (int) endHourSpinner.getValue();
        int endMinute = (int) endMinuteSpinner.getValue();
        String endAmPm = (String) endAmPmSelector.getSelectedItem();

        String startTime = String.format("%d:%02d %s", startHour, startMinute, startAmPm);
        String endTime = String.format("%d:%02d %s", endHour, endMinute, endAmPm);

        String monthName = getMonthName(selectedMonth);

        String fullTask = String.format("   %s (From %s Until %s), %s %d, %d", taskName, startTime,
endTime, monthName, selectedDay, selectedYear);

        tasksByMonth.putIfAbsent(selectedYear, new HashMap<>());
        tasksByMonth.get(selectedYear).putIfAbsent(selectedMonth, new ArrayList<>());
        tasksByMonth.get(selectedYear).get(selectedMonth).add(new Task(fullTask));
```

```
            refreshTasksForSelectedMonth();
            try {
                Clip clip = AudioSystem.getClip();
                clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\notify.wav")));
                clip.start();
            } catch (Exception ex) {
                ex.printStackTrace();
            }

            input.setText("");
        }
    });

    updateButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedYear = calendarComponent.getSelectedYear();
            int selectedMonth = calendarComponent.getSelectedMonth();
            int selectedIndex = list.getSelectedIndex();
            try {
                Clip clip = AudioSystem.getClip();
                clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\Windows
Unlock.wav")));
                clip.start();
            } catch (Exception ex) {
                ex.printStackTrace();
            }

            if (selectedIndex == -1) {
                JOptionPane.showMessageDialog(panel, "Please select a task to update!", "No Task Selected",
JOptionPane.WARNING_MESSAGE);
                return;
            }

            Task selectedTask = tasksByMonth.get(selectedYear).get(selectedMonth).get(selectedIndex);
            if (selectedTask == null) {
                JOptionPane.showMessageDialog(panel, "Selected task is null!", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }

            int response = JOptionPane.showConfirmDialog(panel, "Do you want to update this task?",
"Update Task", JOptionPane.YES_NO_OPTION);
            if (response == JOptionPane.YES_OPTION) {
                JTextField dayField = new JTextField(2);
                dayField.setText(String.valueOf(selectedTask.getDay()));

                JTextField taskNameField = new JTextField(20);
                taskNameField.setText("");

                JSpinner startHourSpinner = new JSpinner(new SpinnerNumberModel(1, 1, 12, 1));
                JSpinner startMinuteSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 59, 1));
                JComboBox<String> startAmPmSelector = new JComboBox<>(new String[]{"AM", "PM"});

                JSpinner endHourSpinner = new JSpinner(new SpinnerNumberModel(1, 1, 12, 1));
                JSpinner endMinuteSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 59, 1));
                JComboBox<String> endAmPmSelector = new JComboBox<>(new String[]{"AM", "PM"});

                JPanel updatePanel = new JPanel(new GridLayout(0, 2, 5, 2));
                updatePanel.add(new JLabel("New Task Name:"));
                updatePanel.add(taskNameField);
                updatePanel.add(new JLabel(" "));
                updatePanel.add(new JLabel(" "));
                updatePanel.add(new JLabel("New Day:"));
                updatePanel.add(dayField);
                updatePanel.add(new JLabel(" "));
                updatePanel.add(new JLabel(" "));

                updatePanel.add(new JLabel("Starting Time:"));
                updatePanel.add(new JPanel(new GridLayout(1, 3, 5, 0)) {{
                    add(startHourSpinner);
                    add(startMinuteSpinner);
                    add(startAmPmSelector);
                }});

                updatePanel.add(new JLabel("Ending Time:"));
                updatePanel.add(new JPanel(new GridLayout(1, 3, 5, 0)) {{
                    add(endHourSpinner);
                    add(endMinuteSpinner);
                    add(endAmPmSelector);
                }});

                int option = JOptionPane.showConfirmDialog(panel, updatePanel, "Update Task",
JOptionPane.OK_CANCEL_OPTION);
                if (option == JOptionPane.OK_OPTION) {
                    String updatedTaskName = taskNameField.getText().trim();
                    String updatedDay = dayField.getText().trim();

                    if (updatedTaskName.isEmpty() || updatedDay.isEmpty()) {
                        JOptionPane.showMessageDialog(panel, "Please enter valid task details!", "Invalid Input",
JOptionPane.WARNING_MESSAGE);
                        return;
                    }

                    int newDay = Integer.parseInt(updatedDay);
                    int startHour = (int) startHourSpinner.getValue();
                    int startMinute = (int) startMinuteSpinner.getValue();
                    String startAmPm = (String) startAmPmSelector.getSelectedItem();

                    int endHour = (int) endHourSpinner.getValue();
                    int endMinute = (int) endMinuteSpinner.getValue();
                    String endAmPm = (String) endAmPmSelector.getSelectedItem();

                    String startTime = String.format("%d:%02d %s", startHour, startMinute, startAmPm);
                    String endTime = String.format("%d:%02d %s", endHour, endMinute, endAmPm);

                    String updatedTask = String.format("   %s (From %s Until %s), %s %d, %d",
                        updatedTaskName, startTime, endTime, getMonthName(selectedMonth), newDay,
selectedYear);

                    tasksByMonth.get(selectedYear).get(selectedMonth).get(selectedIndex).setName(updatedTask);
                    refreshTasksForSelectedMonth();
                }
            }
        }
});

doneButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedIndex = list.getSelectedIndex();
            if (selectedIndex == -1) {
                JOptionPane.showMessageDialog(panel, "Please select a task to mark as done!", "No Task
Selected", JOptionPane.WARNING_MESSAGE);
                return;
            }
```

```
            int selectedYear = calendarComponent.getSelectedYear();
            int selectedMonth = calendarComponent.getSelectedMonth();


            Task taskToMarkDone = tasksByMonth.get(selectedYear).get(selectedMonth).get(selectedIndex);


            achievements.add(taskToMarkDone.getName());


            System.out.println("Achievements: " + achievements);


            tasksByMonth.get(selectedYear).get(selectedMonth).remove(taskToMarkDone);


            refreshTasksForSelectedMonth();


            try {
                Clip clip = AudioSystem.getClip();
                clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\tada.wav")));
                clip.start();
            } catch (Exception ex) {
                ex.printStackTrace();
            }


            JOptionPane.showMessageDialog(panel,"Congratulations you completed the task!!!", "Task
Completed", JOptionPane.INFORMATION_MESSAGE);
```

```
            }
        });



        showAchievementsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                StringBuilder achievementsText = new StringBuilder("Achievements:\n");
                try {
                    Clip clip = AudioSystem.getClip();
                    clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\Windows
Unlock.wav")));
                    clip.start();
                 } catch (Exception ex) {
                    ex.printStackTrace();
                 }

                if (achievements.isEmpty()) {
                    JOptionPane.showMessageDialog(panel, "No achievements yet.", "Achievements",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                      for (String achievement : achievements) {
                     achievementsText.append(achievement).append("\n");
                      }



                    JOptionPane.showMessageDialog(panel, achievementsText.toString(), "Achievements",
JOptionPane.INFORMATION_MESSAGE);


                }
            }
        });





        archiveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int selectedIndex = list.getSelectedIndex();
                if (selectedIndex == -1) {
                    JOptionPane.showMessageDialog(panel, "Please select a task to archive!", "No Task Selected",
JOptionPane.WARNING_MESSAGE);
                    return;
                }

                int selectedYear = calendarComponent.getSelectedYear();
                int selectedMonth = calendarComponent.getSelectedMonth();


                Task taskToArchive = tasksByMonth.get(selectedYear).get(selectedMonth).get(selectedIndex);
                archivedTasks.add(taskToArchive);


                tasksByMonth.get(selectedYear).get(selectedMonth).remove(taskToArchive);


                refreshTasksForSelectedMonth();

                try {
                    Clip clip = AudioSystem.getClip();
                    clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\chimes.wav")));
                    clip.start();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
                JOptionPane.showMessageDialog(panel,"Task has been moved to Archive List!", "Task
Completed", JOptionPane.INFORMATION_MESSAGE);
            }


        });


        showArchiveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {


                StringBuilder archivedTaskNames = new StringBuilder("Archived Tasks:\n");


                if (archivedTasks.isEmpty()) {
                    archivedTaskNames.append("No archived tasks available.");
                } else {
                    for (Task archivedTask : archivedTasks) {
                        archivedTaskNames.append(archivedTask.getName()).append("\n");
                    }
                }


                try {
                    Clip clip = AudioSystem.getClip();
                    clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\Windows
Unlock.wav")));
                    clip.start();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }


                JOptionPane.showMessageDialog(panel, archivedTaskNames.toString(), "Archived Tasks",
JOptionPane.INFORMATION_MESSAGE);

            }





        });







      }
      private void showUpdateDialog(Task taskToUpdate) {


         JPanel updatePanel = new JPanel();
         updatePanel.setLayout(new BoxLayout(updatePanel, BoxLayout.Y_AXIS));



         JTextField taskNameField = new JTextField(taskToUpdate.getName(), 20);
         updatePanel.add(new JLabel("Task Name:"));
         updatePanel.add(taskNameField);



         SpinnerNumberModel startHourModel = new SpinnerNumberModel(1, 1, 12, 1);
         JSpinner startHourSpinner = new JSpinner(startHourModel);
         SpinnerNumberModel startMinuteModel = new SpinnerNumberModel(0, 0, 59, 1);
         JSpinner startMinuteSpinner = new JSpinner(startMinuteModel);
         JComboBox<String> startAmPmSelector = new JComboBox<>(new String[]{"AM", "PM"});

         SpinnerNumberModel endHourModel = new SpinnerNumberModel(1, 1, 12, 1);
```

```
        JSpinner endHourSpinner = new JSpinner(endHourModel);
        SpinnerNumberModel endMinuteModel = new SpinnerNumberModel(0, 0, 59, 1);
        JSpinner endMinuteSpinner = new JSpinner(endMinuteModel);
        JComboBox<String> endAmPmSelector = new JComboBox<>(new String[]{"AM", "PM"});

        updatePanel.add(new JLabel("Starting Time:"));
        updatePanel.add(startHourSpinner);
        updatePanel.add(new JLabel(":"));
        updatePanel.add(startMinuteSpinner);
        updatePanel.add(startAmPmSelector);
        updatePanel.add(new JLabel("Until:"));
        updatePanel.add(endHourSpinner);
        updatePanel.add(new JLabel(":"));
        updatePanel.add(endMinuteSpinner);
        updatePanel.add(endAmPmSelector);



        JComboBox<String> monthSelector = new JComboBox<>(new String[]{"January", "February",
"March", "April", "May", "June",
                                                        "July", "August", "September", "October", "November",
"December"});
        JTextField dayField = new JTextField(2);
        JTextField yearField = new JTextField(4);

        updatePanel.add(new JLabel("Month:"));
        updatePanel.add(monthSelector);
        updatePanel.add(new JLabel("Day:"));
        updatePanel.add(dayField);
        updatePanel.add(new JLabel("Year:"));
        updatePanel.add(yearField);
```

```
JButton saveButton = new JButton("Save Changes");
updatePanel.add(saveButton);



JDialog updateDialog = new JDialog();
updateDialog.setTitle("Update Task");
updateDialog.setSize(400, 400);
updateDialog.setModal(true);
updateDialog.setLocationRelativeTo(panel);

updateDialog.getContentPane().add(updatePanel);



updateDialog.setVisible(true);



saveButton.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {

      String updatedTaskName = taskNameField.getText().trim();
      int updatedStartHour = (int) startHourSpinner.getValue();
      int updatedStartMinute = (int) startMinuteSpinner.getValue();
      String updatedStartAmPm = (String) startAmPmSelector.getSelectedItem();
      int updatedEndHour = (int) endHourSpinner.getValue();
      int updatedEndMinute = (int) endMinuteSpinner.getValue();
      String updatedEndAmPm = (String) endAmPmSelector.getSelectedItem();
      String updatedMonth = (String) monthSelector.getSelectedItem();
      int updatedDay;
      int updatedYear;

      try {
```

```
                updatedDay = Integer.parseInt(dayField.getText().trim());
                updatedYear = Integer.parseInt(yearField.getText().trim());
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(updateDialog, "Invalid date entered!", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }


            String startTime = String.format("%d:%02d %s", updatedStartHour, updatedStartMinute,
updatedStartAmPm);
            String endTime = String.format("%d:%02d %s", updatedEndHour, updatedEndMinute,
updatedEndAmPm);


            String updatedTaskDetails = String.format("   %s (From %s Until %s), %s %d, %d",
                updatedTaskName, startTime, endTime, updatedMonth, updatedDay, updatedYear);
            taskToUpdate.setName(updatedTaskDetails);


            int updatedMonthIndex = monthSelector.getSelectedIndex();
            int selectedYear = calendarComponent.getSelectedYear();
            int selectedMonth = calendarComponent.getSelectedMonth();
            tasksByMonth.get(selectedYear).get(selectedMonth).remove(taskToUpdate);
            tasksByMonth.putIfAbsent(updatedYear, new HashMap<>());
            tasksByMonth.get(updatedYear).putIfAbsent(updatedMonthIndex, new ArrayList<>());
            tasksByMonth.get(updatedYear).get(updatedMonthIndex).add(taskToUpdate);


            refreshTasksForSelectedMonth();
            updateDialog.dispose();
        }
    });
}
public JPanel getPanel() {
    return panel;
}

private void refreshTasksForSelectedMonth() {

        List<Task> tasks = new ArrayList<>();
    listModel.clear();
    DefaultListModel<String> model = new DefaultListModel<>();
    for (Task task : tasks) {
        model.addElement(task.getName());
    }
    taskList.setModel(model);

    int selectedMonth = calendarComponent.getSelectedMonth();
    int selectedYear = calendarComponent.getSelectedYear();

    if (tasksByMonth.containsKey(selectedYear) &&
tasksByMonth.get(selectedYear).containsKey(selectedMonth)) {
        for (Task task : tasksByMonth.get(selectedYear).get(selectedMonth)) {
            listModel.addElement(task.getName());
        }

    }
    System.out.println("Archived Tasks:");
    for (Task archivedTask : archivedTasks) {
        System.out.println(archivedTask.getName());
    }
}

private String getMonthName(int month) {
    String[] monthNames = new String[]{"January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"};

    return monthNames[month];

}


}

package Packs;


abstract class EasySchedFunc {

public abstract void initialize();

}


class Task {

private String name;

private String time;

private int day;




public Task(String name) {

    this.name = name;

}


public String getName() {

    return name;

}


public void setName(String name) {

    this.name = name;

}


public String getTime() {
    return time;
}

public void setTime(String time) {
    this.time = time;
}

public int getDay() {
    return this.day;
}
public void setDay(int day) {
    this.day = day;
}

}

package Packs;

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

public class EasySchedMain {
private JFrame frame;

public EasySchedMain() {
    frame = new JFrame("EasySchedMain - Schedule Manager");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(1520, 480);
    frame.setLayout(new BorderLayout());
    frame.setLocationRelativeTo(null);

    frame.getContentPane().setBackground(new Color(50, 50, 50));
    CalendarFunc calendarComponent = new CalendarFunc();
    calendarComponent.initialize();

    ToDoListFunc toDoListComponent = new ToDoListFunc(calendarComponent);
    toDoListComponent.initialize();



    JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
calendarComponent.getPanel(), toDoListComponent.getPanel());
    splitPane.setDividerLocation(600);

    JPanel controlPanel = new JPanel();
    controlPanel.setBackground(new Color(50, 50, 50));
    controlPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));



    frame.add(splitPane, BorderLayout.CENTER);
    frame.add(controlPanel, BorderLayout.SOUTH);

    frame.setVisible(true);
}

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

package Packs;
```

```
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.border.EmptyBorder;
import javax.swing.border.BevelBorder;
import javax.swing.border.LineBorder;
import javax.swing.border.SoftBevelBorder;

class HomeFrame {
  private JFrame frame;

  public HomeFrame() {
    frame = new JFrame("Welcome to Your Personal Scheduler!");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800, 600);
    frame.getContentPane().setLayout(new BorderLayout());
    frame.setLocationRelativeTo(null);



    frame.getContentPane().setBackground(new Color(109, 129, 150));



    JPanel solidColorPanel = new JPanel();
    solidColorPanel.setBackground(new Color(183, 205, 215));
    solidColorPanel.setLayout(new BorderLayout());
    frame.getContentPane().add(solidColorPanel, BorderLayout.CENTER);
```

```
        JLabel iconLabel = new JLabel();
        iconLabel.setBackground(new Color(104, 168, 165));
        iconLabel.setHorizontalAlignment(SwingConstants.CENTER);
        try {

            ImageIcon originalIcon = new ImageIcon("D:\\Cat with Clock Cartoon Logo (5).png");



            int iconWidth = 450;
            int iconHeight = 450;
            Image highQualityIcon = originalIcon.getImage()
                .getScaledInstance(iconWidth, iconHeight, Image.SCALE_SMOOTH);

            iconLabel.setIcon(new ImageIcon(highQualityIcon));
        } catch (Exception e) {
            System.err.println("Icon could not be loaded: " + e.getMessage());
        }
        solidColorPanel.add(iconLabel, BorderLayout.NORTH);



        String name = showCustomInputDialog();

        JLabel welcomeLabel = new JLabel("Welcome, " + name + " to your Personal Scheduler!!",
SwingConstants.CENTER);
        welcomeLabel.setFont(new Font("Agency FB", Font.BOLD, 21));
        welcomeLabel.setForeground(new Color(255, 255, 255));
        welcomeLabel.setOpaque(false);
        solidColorPanel.add(welcomeLabel, BorderLayout.CENTER);



        Timer fadeInTimer = new Timer(70, new ActionListener() {
            float opacity = 0.0f;


            @Override
            public void actionPerformed(ActionEvent e) {
                opacity += 0.05f;
                if (opacity >= 1.0f) {
                    opacity = 1.0f;
                    ((Timer) e.getSource()).stop();
                }
                welcomeLabel.setForeground(new Color(84, 99, 115, Math.round(opacity * 255)));


            }
        });
        fadeInTimer.start();


        JPanel buttonPanel = new JPanel();
        buttonPanel.setForeground(new Color(255, 255, 255));
        buttonPanel.setBorder(new SoftBevelBorder(BevelBorder.LOWERED, new Color(157, 189, 191), new
Color(173, 200, 201), new Color(173, 200, 201), new Color(173, 200, 201)));
        buttonPanel.setBackground(new Color(101, 156, 163));
        buttonPanel.setLayout(new FlowLayout());


        JButton proceedButton = new JButton("Proceed") {
            @Override
            protected void paintComponent(Graphics g) {
                Graphics2D g2 = (Graphics2D) g;
                g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2.setColor(getBackground());
                g2.fillRoundRect(0, 0, getWidth(), getHeight(), 20, 20);
                super.paintComponent(g);
            }
            @Override
            protected void paintBorder(Graphics g) {
                Graphics2D g2 = (Graphics2D) g;
                g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2.setColor(getForeground());
                g2.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, 20, 20);
            }
        };
        proceedButton.setOpaque(false);
        proceedButton.setContentAreaFilled(false);
        proceedButton.setBackground(new Color(169, 197, 214));
        proceedButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
        proceedButton.setForeground(new Color(0, 166, 83));
        proceedButton.setFocusPainted(false);
        proceedButton.addActionListener(e -> {
            frame.dispose();
            SwingUtilities.invokeLater(() -> new EasySchedMain());
            try {
                Clip clip = AudioSystem.getClip();
                clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\Windows Print
complete.wav")));
                clip.start();
            } catch (Exception ex) {
                ex.printStackTrace();
            }


        });


        JButton exitButton = new JButton("Exit") {
            @Override
```

```
            protected void paintComponent(Graphics g) {
                Graphics2D g2 = (Graphics2D) g;
                g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2.setColor(getBackground());
                g2.fillRoundRect(0, 0, getWidth(), getHeight(), 20, 20);
                super.paintComponent(g);


            }



            @Override
            protected void paintBorder(Graphics g) {
                Graphics2D g2 = (Graphics2D) g;
                g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2.setColor(getForeground());
                g2.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, 20, 20);
            }
        };
        exitButton.setOpaque(false);
        exitButton.setContentAreaFilled(false);
        exitButton.setBackground(new Color(169, 197, 214));
        exitButton.setFont(new Font("Segoe UI Black", Font.BOLD, 12));
        exitButton.setForeground(new Color(255, 85, 85));
        exitButton.setFocusPainted(false);
        exitButton.addActionListener(e -> System.exit(0));



        buttonPanel.add(proceedButton);
        buttonPanel.add(exitButton);
        solidColorPanel.add(buttonPanel, BorderLayout.SOUTH);
        frame.setVisible(true);
        try {
            Clip clip = AudioSystem.getClip();
            clip.open(AudioSystem.getAudioInputStream(new File("C:\\Windows\\Media\\Ring06.wav")));
            clip.start();
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }



    private String showCustomInputDialog() {

        JPanel panel = new JPanel();
        panel.setBackground(new Color(230, 241, 242));

        JLabel label = new JLabel("Enter your name:");
        label.setFont(new Font("Agency FB", Font.BOLD, 16));
        label.setForeground(new Color(70, 64, 69));
        label.setHorizontalAlignment(SwingConstants.CENTER);

        JTextField textField = new JTextField(20);
        textField.setFont(new Font("Agency FB", Font.PLAIN, 16));
        textField.setForeground(new Color(70, 64, 69));
        textField.setBackground(new Color(255, 255, 255));
        textField.setPreferredSize(new Dimension(250, 30));

        panel.setLayout(new BorderLayout());
        panel.add(label, BorderLayout.NORTH);
        panel.add(textField, BorderLayout.CENTER);



        String name = "";
        while (name.trim().isEmpty()) {
            int option = JOptionPane.showOptionDialog(frame, panel, "Personal Scheduler",
                    JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE,
                    null, new Object[] { "OK" }, "OK");

            if (option == JOptionPane.OK_OPTION) {
                name = textField.getText();

                if (name.trim().isEmpty()) {
                    JOptionPane.showMessageDialog(frame, "Name is required. Please enter a valid name.", "Input
Error", JOptionPane.ERROR_MESSAGE);
                }
            } else {
                break;
            }
        }

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

References

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7th ed.).
Deitel, P. J., & Deitel, H. M. (2019). *Java: How to program* (11th ed.). Pearson.
Savitch, W. (2018). *Absolute Java* (6th ed.). Pearson.

-

Republic of the Philippines
CAVITE STATE UNIVERSITY
Bacoor City Campus
SHIV, Molino VI, City of Bacoor

DEPARTMENT OF COMPUTER STUDIES
FINAL LABORATORY EXAMINATION IN
DCIT 50: OBJECT ORIENTED PROGRAMMING
1st Semester, A.Y. 2021-2022

INTENDED LEARNING OUTCOMES:

After the completion of the unit, students will be able to:

- Develop an application that demonstrates the functions of Object-Oriented Programming with swing components

LABORATORY EXAM: ACTIVITY COMPILATION USING SWING COMPONENTS

DIRECTION:

- Brainstorm a project title that provide solution to real life problem
- Using swing components, create a menu that compiles your classes as one program. Note that you must convert the other console programs in a window application and that the user must be able to choose any program to execute using your menu.
- Attach your project file on google classroom together with this activity sheet.

rubrics for grading

Category	Description
Program Execution (25pts)	Program executes correctly with no syntax or runtime errors.
Correctness (25pts)	Program produces correct answers or appropriate results for all inputs tested.
Completeness (25pts)	Program shows evidence of excellent case analysis, and all possible cases are handled appropriately.
Logic (25pts)	Program logic is correct, with no known boundary errors, and no redundant or contradictory conditions.