OBJECT ORIENTED PROGRAMMING (JAVA)

CIA 2 PROJECT

TEAM MEMBERS :

- Nighil Natarajan – 23011101085 (AI/DS-B)
- Prashanth Samkumar – 23011100100 (AI/DS-B)
- Yashawanth Krishna Devendran – 23011101166 (AI/DS-B)

TOPIC :

- Creating an application that demonstrates expense tracker and to-do list manages.

OBJECTIVE :

- This Java application serves as a personal productivity tool, combining an expense tracker with a to-do list.
- It allows users to record their expenses, categorize them, and maintain a list of tasks with priority levels.
- All data is saved locally and loaded on startup, so user information persists across sessions.

INDIVIDUAL CONTRIBUTIONS OF EACH TEAM MEMBER:

- **Nighil Natarajan :**
  Created user defined data types to handle expenses and tasks.
- **Prashanth Samkumar :**
  Created the function to retrive and store data in files.
- **Yashawanth Krishna Devendran :**
  Developed the user interface in this application.

SOURCE CODE :

```
/*
   Object-Oriented Programming
      CIA - 2 Assignment


   Team Members:
   - Nighil Natarajan - 23011101085
   - Prashanth Samkumar - 23011100100
   - Yashawanth Krishna Devendran - 23011101166


   Project: Expense Tracker & To-Do List Application
   This Java application serves as a personal productivity tool, combining an expense tracker with a to-do list.
   It allows users to record their expenses, categorize them, and maintain a list of tasks with priority levels.
   All data is saved locally and loaded on startup, so user information persists across sessions.
*/


import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.swing.*;
import javax.swing.border.EmptyBorder;


public class OOPS_CIA2 extends JFrame implements Serializable {
```

```
    private List<Expense> expenses;  // List to store expenses
    private List<Task> tasks;        // List to store tasks
    private JTextArea expenseDisplayArea; // Text area for displaying expenses
    private JTextArea taskDisplayArea;    // Text area for displaying tasks
    private JLabel summaryLabel;     // Label for showing summary at the bottom

    // Constructor for initializing the main application
    public OOPS_CIA2() {
        expenses = loadExpenses();   // Load saved expenses on startup
        tasks = loadTasks();         // Load saved tasks on startup

        setTitle("Expense Tracker & To-Do List");  // Set window title
        setSize(700, 600);           // Set window size
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  // Exit application on close
        setLocationRelativeTo(null); // Center the window on screen

        // Tabbed Pane for managing separate views for Expenses and Tasks
        JTabbedPane tabbedPane = new JTabbedPane();
        tabbedPane.addTab("Expenses", createExpensePanel());  // Add expense panel as a tab
        tabbedPane.addTab("Tasks", createTaskPanel());        // Add task panel as a tab
        add(tabbedPane, BorderLayout.CENTER);  // Add tabbedPane to the center of the frame

        displaySummary();  // Show initial summary

        // Add window listener to save data on window close
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
```

```
            saveData();  // Save data when closing
        }
    });
}


/*
    Yashawanth Krishna Devendran : Developed the user interface in this application.
*/


// Method to create the Expense panel
private JPanel createExpensePanel() {
    JPanel panel = new JPanel(new BorderLayout(10, 10));  // Panel with BorderLayout and padding
    panel.setBorder(new EmptyBorder(10, 10, 10, 10));  // Adds a border for spacing


    // Form panel with grid layout for organizing input fields
    JPanel formPanel = new JPanel(new GridLayout(5, 2, 5, 5));
    JTextField titleField = new JTextField();        // Field for expense title
    JTextField descriptionField = new JTextField();  // Field for expense description
    JTextField amountField = new JTextField();       // Field for expense amount


    JComboBox<String> categoryField = new JComboBox<>(new String[]{"Food", "Travel", "Entertainment", "Other"});
    formPanel.add(new JLabel("Title:"));         // Label for title
    formPanel.add(titleField);                   // Add title input field to form
    formPanel.add(new JLabel("Description:"));   // Label for description
    formPanel.add(descriptionField);             // Add description input field
    formPanel.add(new JLabel("Amount:"));        // Label for amount
    formPanel.add(amountField);                  // Add amount input field
    formPanel.add(new JLabel("Category:"));      // Label for category
```

```
        formPanel.add(categoryField);               // Add category dropdown

        // Button to add expense
        JButton addExpenseButton = new JButton("Add Expense");
        addExpenseButton.setBackground(new Color(70, 130, 180));  // Set button color
        addExpenseButton.setForeground(Color.WHITE);  // Set button text color
        formPanel.add(addExpenseButton);  // Add button to form panel
        panel.add(formPanel, BorderLayout.NORTH);  // Place form panel at top of the
main panel

        // Text area for displaying expenses with font settings and border
        expenseDisplayArea = new JTextArea();
        expenseDisplayArea.setEditable(false);  // Make it read-only
        expenseDisplayArea.setFont(new Font("Monospaced", Font.PLAIN, 14));  // Set
font
        expenseDisplayArea.setBorder(BorderFactory.createTitledBorder("Expenses
List"));  // Add title border
        panel.add(new JScrollPane(expenseDisplayArea), BorderLayout.CENTER);  //
Add scrollable area

        // Action listener for adding an expense
        addExpenseButton.addActionListener(e -> {
            try {
                String title = titleField.getText().trim();  // Get title from input
                String description = descriptionField.getText().trim();  // Get description
                String category = (String) categoryField.getSelectedItem();  // Get category

                // Validate amount field and other fields
                if (title.isEmpty() || description.isEmpty() ||
amountField.getText().trim().isEmpty()) {
                    JOptionPane.showMessageDialog(this, "Please fill in all fields.");
                    return;
```

```
            }

            double amount = Double.parseDouble(amountField.getText().trim());  // Parse amount as double

            Expense expense = new Expense(title, description, amount, category, new Date());  // Create new Expense object
            expenses.add(expense);  // Add to expenses list
            expenseDisplayArea.append("Added Expense: " + expense.getDetails() + "\n");  // Display in text area

            displaySummary();  // Update summary
            titleField.setText("");  // Clear input fields
            descriptionField.setText("");
            amountField.setText("");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid amount. Please enter a valid number.");  // Show error for invalid input
        }
    });

    return panel;  // Return the constructed expense panel
}

// Method to create the Task panel
private JPanel createTaskPanel() {
    JPanel panel = new JPanel(new BorderLayout(10, 10));  // Panel with padding
    panel.setBorder(new EmptyBorder(10, 10, 10, 10));  // Add border padding

    // Form panel for task input fields
    JPanel formPanel = new JPanel(new GridLayout(4, 2, 5, 5));
    JTextField titleField = new JTextField();         // Field for task title
```

```
        JTextField descriptionField = new JTextField();   // Field for task description

        JComboBox<String> priorityField = new JComboBox<>(new String[]{"Low",
"Medium", "High"});
        formPanel.add(new JLabel("Title:"));            // Label for title
        formPanel.add(titleField);                      // Add title input field
        formPanel.add(new JLabel("Description:"));      // Label for description
        formPanel.add(descriptionField);                // Add description input field
        formPanel.add(new JLabel("Priority:"));         // Label for priority
        formPanel.add(priorityField);                   // Add priority dropdown

        JButton addTaskButton = new JButton("Add Task");  // Button for adding task
        addTaskButton.setBackground(new Color(60, 179, 113));  // Set button color
        addTaskButton.setForeground(Color.WHITE);  // Set button text color
        formPanel.add(addTaskButton);  // Add button to form panel
        panel.add(formPanel, BorderLayout.NORTH);  // Place form panel at top of the
main panel

        // Text area for displaying tasks
        taskDisplayArea = new JTextArea();
        taskDisplayArea.setEditable(false);  // Make read-only
        taskDisplayArea.setFont(new Font("Monospaced", Font.PLAIN, 14));  // Set font
        taskDisplayArea.setBorder(BorderFactory.createTitledBorder("Tasks List"));  //
Add border with title
        panel.add(new JScrollPane(taskDisplayArea), BorderLayout.CENTER);  // Add
scroll pane

        // Action listener for adding a task
        addTaskButton.addActionListener(e -> {
            String title = titleField.getText().trim();  // Get title
            String description = descriptionField.getText().trim();  // Get description
            String priority = (String) priorityField.getSelectedItem();  // Get priority
```

```
            if (title.isEmpty() || description.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill in all fields.");
                return;
            }

            Task task = new Task(title, description, new Date(), priority);  // Create new Task
            tasks.add(task);  // Add to tasks list
            taskDisplayArea.append("Added Task: " + task.getDetails() + "\n");  // Display in text area

            titleField.setText("");  // Clear input fields
            descriptionField.setText("");
        });

        return panel;  // Return the constructed task panel
    }


/*
        Prashanth Samkumar : Created the function to retrive and store data in files.
*/

    // Method to display a summary at the bottom of the frame
    private void displaySummary() {
        double totalAmount = expenses.stream().mapToDouble(Expense::getAmount).sum();  // Sum of all expense amounts
        long expenseCount = expenses.size();  // Count of expenses
        long taskCount = tasks.size();  // Count of tasks
```

```
        if (summaryLabel == null) {
            summaryLabel = new JLabel();  // Initialize summary label
            summaryLabel.setBorder(new EmptyBorder(10, 0, 10, 0));  // Add padding
            summaryLabel.setFont(new Font("SansSerif", Font.BOLD, 14));  // Set font
style
            add(summaryLabel, BorderLayout.SOUTH);  // Add label to frame bottom
        }

        // Update summary label text
        summaryLabel.setText(String.format("Total Expenses: %d | Total Amount: %.2f |
Total Tasks: %d",
                expenseCount, totalAmount, taskCount));
    }

    // Save data for both expenses and tasks
    private void saveData() {
        try (ObjectOutputStream oosExpenses = new ObjectOutputStream(new
FileOutputStream("expenses.ser"));
             ObjectOutputStream oosTasks = new ObjectOutputStream(new
FileOutputStream("tasks.ser"))) {
            oosExpenses.writeObject(expenses);  // Save expenses
            oosTasks.writeObject(tasks);        // Save tasks
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    // Load expenses from file
    private List<Expense> loadExpenses() {
```

```
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("expenses.ser"))) {
            return (List<Expense>) ois.readObject();  // Read expenses
        } catch (IOException | ClassNotFoundException e) {
            return new ArrayList<>();  // Return empty list if file not found
        }
    }

    // Load tasks from file
    private List<Task> loadTasks() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("tasks.ser"))) {
            return (List<Task>) ois.readObject();  // Read tasks
        } catch (IOException | ClassNotFoundException e) {
            return new ArrayList<>();  // Return empty list if file not found
        }
    }



/*
        Nighil Natarajan  : Created user defined data types to handle expenses and tasks.
 */

    // Expense class to store expense data
    static class Expense implements Serializable {
        private final String title;
        private final String description;
        private final double amount;
        private final String category;
        private final Date date;
```

```
        public Expense(String title, String description, double amount, String category,
Date date) {
            this.title = title;
            this.description = description;
            this.amount = amount;
            this.category = category;
            this.date = date;
        }

        public double getAmount() {
            return amount;
        }

        public String getDetails() {
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
            return String.format("%s | %s | %.2f | %s | %s", title, description, amount,
category, formatter.format(date));
        }
    }

    // Task class to store task data
    static class Task implements Serializable {
        private final String title;
        private final String description;
        private final Date date;
        private final String priority;

        public Task(String title, String description, Date date, String priority) {
            this.title = title;
            this.description = description;
```

```
            this.date = date;
            this.priority = priority;
        }

        public String getDetails() {
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
            return String.format("%s | %s | %s | %s", title, description, priority,
formatter.format(date));
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new OOPS_CIA2().setVisible(true);
        });
    }
}
```