Motioncut Internship Java Programming week-3

Expense Tracker

- Objective: Develop a basic expense tracking application in Java.

- Details: This project will allow you to apply your Java programming skills to create a useful application. Here are the key features your Expense Tracker should have:

  - Add Expenses: Users should be able to input their daily expenses. Each expense entry should include a description, amount, and category (e.g., groceries, transportation, entertainment).

   - View Expenses: Users should be able to view a list of their recorded expenses, including details such as the description, amount, and category.

   - Expense Summaries: Implement functionality to provide expense summaries, such as total expenses for a specific category or for a given time period.

  - User-Friendly Interface: Design a user-friendly interface that prompts users to input their expenses and displays summaries clearly.

   - Data Persistence: Ensure that the entered expenses are saved and can be accessed in future sessions.

   - Input Validation: Handle input validation to ensure that users enter valid expense details.

**Code:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class ExpenseTrackerApp {

private List<Expense> expenses = new ArrayList<>();

private Map<String, Double>categorySummaries = new HashMap<>();

private JFrame frame;

private DefaultListModel<Expense>expenseListModel = new DefaultListModel<>();

private JList<Expense>expenseList;

private JTextFielddescriptionField;

private JTextFieldamountField;

private JTextFieldcategoryField;

private JLabelsummaryLabel;

public ExpenseTrackerApp() {

frame = new JFrame("Expense Tracker");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 400);

// Define colors

ColorprimaryColor = new Color(144, 238, 144); // Light Green

ColorbrownColor = new Color(139, 69, 19);

// Set the background color of the frame

frame.getContentPane().setBackground(primaryColor);

// Create and set custom fonts for labels and buttons

Font labelFont = new Font("Arial", Font.BOLD, 16);

Font buttonFont = new Font("Arial", Font.PLAIN, 14);

// Create a panel for input fields

JPanelinputPanel = new JPanel(new GridLayout(4, 2));

inputPanel.setBackground(primaryColor);

// Create and style labels

JLabeldescriptionLabel = new JLabel("Description:");

descriptionLabel.setFont(labelFont);

descriptionLabel.setForeground(brownColor);

JLabelamountLabel = new JLabel("Amount:");

amountLabel.setFont(labelFont);

amountLabel.setForeground(brownColor);

JLabelcategoryLabel = new JLabel("Category:");

categoryLabel.setFont(labelFont);

categoryLabel.setForeground(brownColor);

// Create and style text fields

descriptionField = new JTextField();

amountField = new JTextField();

categoryField = new JTextField();

// Create and style the "Add Expense" button

JButtonaddButton = new JButton("Add Expense");

addButton.setFont(buttonFont);

addButton.setForeground(brownColor);

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addExpense();

}

});

// Create a panel for the "Add Expense" button

JPanelbuttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

buttonPanel.setBackground(primaryColor);

buttonPanel.add(addButton);

// Create and style the summary label

summaryLabel = new JLabel("Category Summary: ");

summaryLabel.setFont(labelFont);

summaryLabel.setForeground(brownColor);

// Create a panel for the summary label

JPanelsummaryPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

summaryPanel.setBackground(primaryColor);

summaryPanel.add(summaryLabel);

// Add the components to the input panel

inputPanel.add(descriptionLabel);

inputPanel.add(descriptionField);

inputPanel.add(amountLabel);

inputPanel.add(amountField);

inputPanel.add(categoryLabel);

inputPanel.add(categoryField);

// Add the input panel, button panel, and summary panel to the frame

frame.add(inputPanel, BorderLayout.NORTH);

frame.add(buttonPanel, BorderLayout.CENTER);

frame.add(summaryPanel, BorderLayout.SOUTH);

frame.setVisible(true);

}

private void addExpense() {

String description = descriptionField.getText();

String amountText = amountField.getText();

String category = categoryField.getText();

if (description.isEmpty() || amountText.isEmpty() || category.isEmpty()) {

JOptionPane.showMessageDialog(frame, "Please fill in all fields.");

return;

}

double amount;

try {

amount = Double.parseDouble(amountText);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Invalid amount format.");

return;

}

Expense expense = new Expense(description, amount, category);

expenses.add(expense);

expenseListModel.addElement(expense);

updateCategorySummary(category, amount);

// Clear input fields

descriptionField.setText("");

amountField.setText("");

categoryField.setText("");

}

private void updateCategorySummary(String category, double amount) {

if (categorySummaries.containsKey(category)) {

double currentTotal = categorySummaries.get(category);

categorySummaries.put(category, currentTotal + amount);

} else {

categorySummaries.put(category, amount);

}

updateSummaryLabel();

}

private void updateSummaryLabel() {

StringBuilder summaryText = new StringBuilder("Category Summary: ");

for (Map.Entry<String, Double>entry :categorySummaries.entrySet()) {

summaryText.append(entry.getKey()).append(" - ₹").append(entry.getValue()).append(" \n ");

}

summaryLabel.setText(summaryText.toString());

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new ExpenseTrackerApp();

}

});

}

private class Expense {

private String description;

private double amount;

private String category;

public Expense(String description, double amount, String category) {

this.description = description;

this.amount = amount;

this.category = category;

}

@Override

public String toString() {

return description + " - ₹" + amount + " (" + category + ")";

}

}

}

**Output:**

