**Expense Tracker**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

class Expense {

private String description;

private double amount;

private String category;

private Date date;

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

this.description = description;

this.amount = amount;

this.category = category;

this.date = new Date();

}

// Getters and setters

}

class ExpenseTracker {

private List<Expense> expenses;

public ExpenseTracker() {

expenses = new ArrayList<>();

}

public void addExpense(Expense expense) {

expenses.add(expense);

}

public List<Expense> getExpenses() {

return expenses;

}

public double getTotalExpenseForCategory(String category) {

double totalExpense = 0;

for (Expense expense : expenses) {

if (expense.getCategory().equalsIgnoreCase(category)) {

totalExpense += expense.getAmount();

}

}

return totalExpense;

}

}

public class ExpenseTrackerApp {

private static ExpenseTracker tracker;

private static JTextField descriptionField, amountField, categoryField;

private static JTextArea expenseListArea;

private static JComboBox<String> categoryDropdown;

private static SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

public static void main(String[] args) {

tracker = new ExpenseTracker();

JFrame frame = new JFrame("Expense Tracker");

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

frame.setLayout(new BorderLayout());

JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridLayout(4, 2));

inputPanel.add(new JLabel("Description:"));

descriptionField = new JTextField();

inputPanel.add(descriptionField);

inputPanel.add(new JLabel("Amount:"));

amountField = new JTextField();

inputPanel.add(amountField);

inputPanel.add(new JLabel("Category:"));

categoryField = new JTextField();

inputPanel.add(categoryField);

JButton addButton = new JButton("Add Expense");

inputPanel.add(addButton);

frame.add(inputPanel, BorderLayout.NORTH);

expenseListArea = new JTextArea(20, 50);

expenseListArea.setEditable(false);

frame.add(new JScrollPane(expenseListArea), BorderLayout.CENTER);

categoryDropdown = new JComboBox<>();

categoryDropdown.addItem("All");

frame.add(categoryDropdown, BorderLayout.SOUTH);

loadExpensesFromFile();

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String description = descriptionField.getText();

String amountStr = amountField.getText();

String category = categoryField.getText();

// Input validation

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

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

return;

}

try {

double amount = Double.parseDouble(amountStr);

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

tracker.addExpense(expense);

updateExpenseList();

saveExpensesToFile();

descriptionField.setText("");

amountField.setText("");

categoryField.setText("");

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Invalid amount. Please enter a valid number.");

}

}

});

categoryDropdown.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String selectedCategory = (String) categoryDropdown.getSelectedItem();

updateExpenseList(selectedCategory);

}

});

frame.pack();

frame.setVisible(true);

}

private static void updateExpenseList() {

updateExpenseList("All");

}

private static void updateExpenseList(String categoryFilter) {

StringBuilder sb = new StringBuilder();

for (Expense expense : tracker.getExpenses()) {

if (categoryFilter.equals("All") || expense.getCategory().equalsIgnoreCase(categoryFilter)) {

sb.append(dateFormat.format(expense.getDate())).append(" - ")

.append(expense.getDescription()).append(" - $")

.append(expense.getAmount()).append("\n");

}

}

expenseListArea.setText(sb.toString());

if (!categoryFilter.equals("All")) {

double totalExpense = tracker.getTotalExpenseForCategory(categoryFilter);

JOptionPane.showMessageDialog(null, "Total Expense for " + categoryFilter + ": $" + totalExpense);

}

}

private static void saveExpensesToFile() {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("expenses.dat"))) {

oos.writeObject(tracker.getExpenses());

} catch (IOException e) {

e.printStackTrace();

}

}

private static void loadExpensesFromFile() {

File file = new File("expenses.dat");

if (file.exists()) {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {

List<Expense> loadedExpenses = (List<Expense>) ois.readObject();

tracker.getExpenses().addAll(loadedExpenses);

updateExpenseList();

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

}