**DAILY EXPENSE TRACKER**

**Category class**

**package** com.expensetracker;

**public** **enum** Category {

***FOOD***, ***TRANSPORT***, ***ENTERTAINMENT***, ***UTILITIES***, ***HEALTHCARE***, ***EDUCATION***, ***GROCERIES***, ***RENT***, ***TRAVEL***;

}

**Expense class**

**package** com.expensetracker;

**public** **class** Expense {

**private** String description;

**private** **double** amount;

**private** Category category;

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

**this**.description = description;

**this**.amount = amount;

**this**.category = category;

}

**public** String getDescription() {

**return** description;

}

**public** **double** getAmount() {

**return** amount;

}

**public** Category getCategory() {

**return** category;

}

@Override

**public** String toString() {

**return** String.*format*("Description: %s, Amount: %.2f, Category: %s", description, amount, category);

}

}

**ExpenseTracker class**

package com.expensetracker;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

public class ExpenseTracker {

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

public void addExpense(String description, double amount, Category category) {

expenses.add(new Expense(description, amount, category));

}

public void addMultipleExpenses(List<String> descriptions, List<Double> amounts, Category category) {

if (descriptions.size() != amounts.size()) {

throw new IllegalArgumentException("Descriptions and amounts lists must be of the same size.");

}

for (int i = 0; i < descriptions.size(); i++) {

addExpense(descriptions.get(i), amounts.get(i), category);

}

}

public List<Expense> getExpenses() {

return new ArrayList<>(expenses);

}

public List<Expense> getExpensesByCategory(Category category) {

return expenses.stream()

.filter(expense -> expense.getCategory() == category)

.collect(Collectors.toList());

}

public double getTotalExpenses() {

return expenses.stream()

.mapToDouble(Expense::getAmount)

.sum();

}

public double getTotalExpensesByCategory(Category category) {

return getExpensesByCategory(category).stream()

.mapToDouble(Expense::getAmount)

.sum();

}

}

**ExpenseTrackerGUI class**

package com.expensetracker;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class ExpenseTrackerGUI extends JFrame {

private ExpenseTracker tracker;

private JTextArea expenseTextArea;

private JTextField descriptionField;

private JTextField amountField;

private JTextField categoryField;

private JLabel totalLabel;

public ExpenseTrackerGUI() {

tracker = new ExpenseTracker();

setTitle("Expense Tracker");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 600, 500);

getContentPane().setLayout(null);

JLabel lblDescription = new JLabel("Description:");

lblDescription.setBounds(10, 11, 80, 14);

getContentPane().add(lblDescription);

descriptionField = new JTextField();

descriptionField.setBounds(100, 8, 150, 20);

getContentPane().add(descriptionField);

descriptionField.setColumns(10);

JLabel lblAmount = new JLabel("Amount:");

lblAmount.setBounds(10, 42, 80, 14);

getContentPane().add(lblAmount);

amountField = new JTextField();

amountField.setBounds(100, 39, 150, 20);

getContentPane().add(amountField);

amountField.setColumns(10);

JLabel lblCategory = new JLabel("Category:");

lblCategory.setBounds(10, 73, 80, 14);

getContentPane().add(lblCategory);

categoryField = new JTextField();

categoryField.setBounds(100, 70, 150, 20);

getContentPane().add(categoryField);

categoryField.setColumns(10);

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

btnAddExpense.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addExpense();

}

});

btnAddExpense.setBounds(10, 104, 150, 23);

getContentPane().add(btnAddExpense);

JButton btnViewExpenses = new JButton("View Expenses");

btnViewExpenses.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

viewExpenses();

}

});

btnViewExpenses.setBounds(180, 104, 150, 23);

getContentPane().add(btnViewExpenses);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 138, 564, 250);

getContentPane().add(scrollPane);

expenseTextArea = new JTextArea();

expenseTextArea.setEditable(false);

scrollPane.setViewportView(expenseTextArea);

totalLabel = new JLabel("Total Expenses: 0.00");

totalLabel.setBounds(10, 400, 200, 14);

getContentPane().add(totalLabel);

}

private void addExpense() {

try {

String description = descriptionField.getText();

double amount = Double.parseDouble(amountField.getText());

Category category = Category.valueOf(categoryField.getText().toUpperCase());

tracker.addExpense(description, amount, category);

descriptionField.setText("");

amountField.setText("");

categoryField.setText("");

showAlert("Expense Added", "Expense has been successfully added.");

} catch (Exception e) {

showAlert("Invalid Input", "Please enter valid data.");

}

}

private void viewExpenses() {

expenseTextArea.setText("");

for (Category category : Category.values()) {

expenseTextArea.append(category.name() + ":\n");

for (Expense expense : tracker.getExpensesByCategory(category)) {

expenseTextArea.append(" " + expense.toString() + "\n");

}

}

totalLabel.setText("Total Expenses: " + tracker.getTotalExpenses());

}

private void showAlert(String title, String message) {

JOptionPane.showMessageDialog(this, message, title, JOptionPane.INFORMATION\_MESSAGE);

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

ExpenseTrackerGUI frame = new ExpenseTrackerGUI();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

}

**Main class**

package com.expensetracker;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

ExpenseTracker tracker = new ExpenseTracker();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Add Expense");

System.out.println("2. Add Multiple Expenses");

System.out.println("3. View Expenses");

System.out.println("4. View Expenses by Category");

System.out.println("5. View Summary");

System.out.println("6. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter description: ");

String description = scanner.nextLine();

System.out.print("Enter amount: ");

double amount = scanner.nextDouble();

scanner.nextLine(); // Consume newline

System.out.print("Enter category (FOOD, TRANSPORT, ENTERTAINMENT, UTILITIES, HEALTHCARE, EDUCATION, GROCERIES, RENT, TRAVEL): ");

String categoryStr = scanner.nextLine().toUpperCase();

Category category = Category.valueOf(categoryStr);

tracker.addExpense(description, amount, category);

break;

case 2:

List<String> descriptions = new ArrayList<>();

List<Double> amounts = new ArrayList<>();

System.out.print("Enter category (FOOD, TRANSPORT, ENTERTAINMENT, UTILITIES, HEALTHCARE, EDUCATION, GROCERIES, RENT, TRAVEL): ");

String multiCategoryStr = scanner.nextLine().toUpperCase();

Category multiCategory = Category.valueOf(multiCategoryStr);

while (true) {

System.out.print("Enter description (or type 'done' to finish): ");

String desc = scanner.nextLine();

if (desc.equalsIgnoreCase("done")) {

break;

}

System.out.print("Enter amount: ");

double amt = scanner.nextDouble();

scanner.nextLine(); // Consume newline

descriptions.add(desc);

amounts.add(amt);

}

tracker.addMultipleExpenses(descriptions, amounts, multiCategory);

break;

case 3:

tracker.getExpenses().forEach(System.out::println);

break;

case 4:

System.out.print("Enter category: ");

String cat = scanner.nextLine().toUpperCase();

tracker.getExpensesByCategory(Category.valueOf(cat)).forEach(System.out::println);

break;

case 5:

System.out.println("Total Expenses: " + tracker.getTotalExpenses());

System.out.println("Enter category for summary (or leave blank for all): ");

String catSummary = scanner.nextLine().toUpperCase();

if (!catSummary.isEmpty()) {

System.out.println("Total Expenses for " + catSummary + ": " + tracker.getTotalExpensesByCategory(Category.valueOf(catSummary)));

}

break;

case 6:

scanner.close();

return;

default:

System.out.println("Invalid option. Try again.");

}

}

}

}