Personal Finance & Expense Tracker

05 January 2026 20:38

Project Title: SpendWise CLI

Objective: A command-line application to track income and expenses, categorize transactions, and persist data to a local file.

1. Core Requirements (The OOP Foundation)

You should structure your code using the four pillars of OOP (Encapsulation, Inheritance, Abstraction, and Polymorphism).

- **Abstract Class Transaction:** Contains shared fields like amount, date, and category.
- **Subclasses Income and Expense:** Inherit from Transaction. You can add specific logic, like an Expense having a isEssential boolean.
- **Interface Reportable:** Define a method String formatForReport() that both subclasses implement.
- **Encapsulation:** All fields should be private with appropriate getters and setters.

2. Data Management (Collections)

Instead of simple arrays, use the **Java Collections Framework** to handle data dynamically.

- **ArrayList<Transaction>:** Use this as your primary in-memory storage for the current session.
- **HashSet<String>:** Use this to store unique categories (e.g., "Food", "Rent", "Salary") to prevent duplicates.
- **HashMap<String, Double>:** Use this to generate a summary report (Category Name -> Total Spent).

3. Data Persistence (File Handling)

To ensure data isn't lost when the program closes, implement a FileManager class.

- **Save Data:** On exit, iterate through your ArrayList and write each transaction to a transactions.txt or data.csv file using BufferedWriter.
- **Load Data:** On startup, use BufferedReader or Scanner to parse the file, recreate the objects, and repopulate your ArrayList.

8-Hour Development Timeline

Time	Task	Focus Area
Hour 1	**Setup & Models**	Create UML, Create Transaction, Income, and Expense classes.
Hour 2	**Logic Engine**	Create an AccountManager class to handle the ArrayList.
Hour 3	**CLI Menu**	Build the switch-case loop for User Input (Add, View, Delete).
Hour 4-5	**File Handling**	Implement the Save/Load logic. Handle IOException.
Hour 6	**Collections Logic**	Implement the "View by Category" summary using a Map.
Hour 7	**Validation**	Add try-catch blocks for invalid inputs (e.g., entering text for an amount).
Hour 8	**Testing & Polish**	Final bug fixes and code commenting.

Suggested Class Diagram

Example Snippet (The "Save" Logic)

Java

```
public void saveToFile(List<Transaction> transactions) {
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("data.txt"))) {
        for (Transaction t : transactions) {
            writer.write(t.toFileString()); // Custom method to format data
            writer.newLine();
        }
    } catch (IOException e) {
        System.err.println("Error saving data: " + e.getMessage());
    }
}
```

```
--- SpendWise Menu ---
1. Add Expense
2. View Summary
3. Save and Exit
> 1
Enter amount: 50.00
Enter category: Food
Expense added!
```