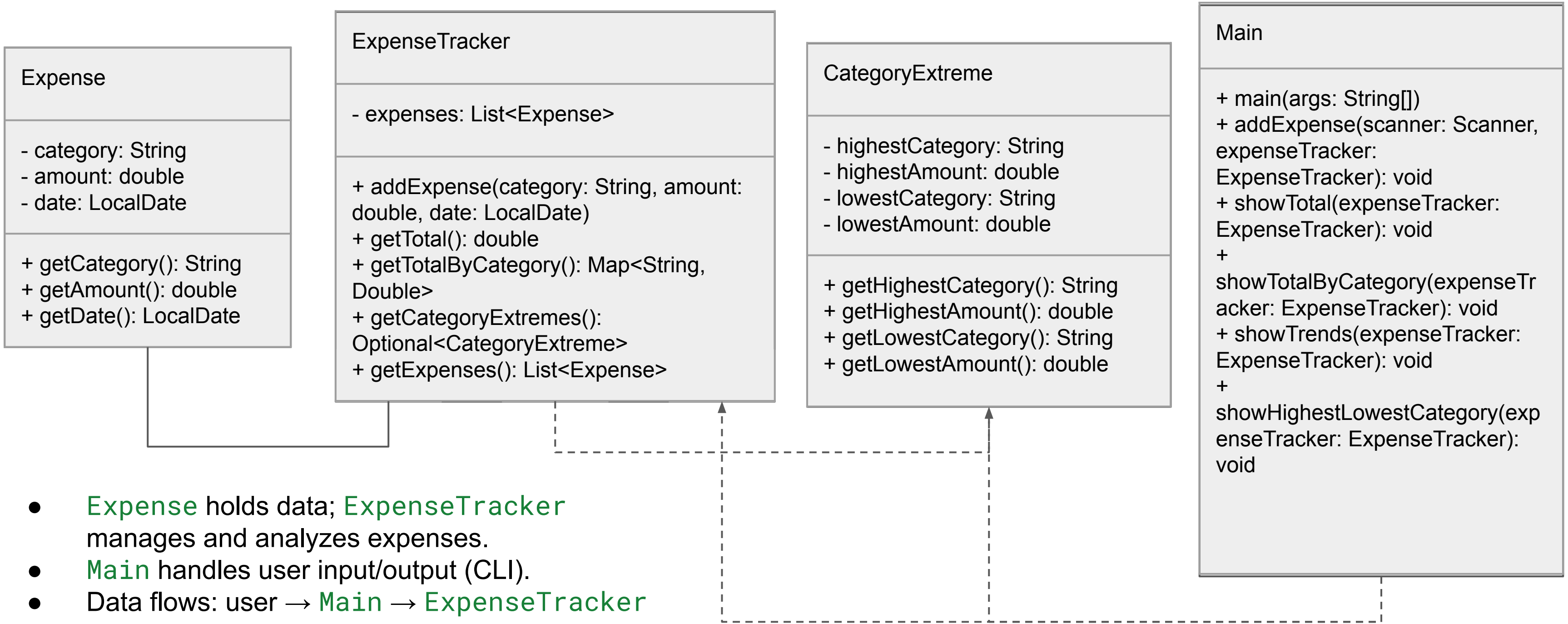
Java Expense Tracker Application

Presented By

Rovia Simmons

12/14/2025

Approach and Design



Expense

- category: String
- amount: double
- date: LocalDate

+ getCategory(): String
+ getAmount(): double
+ getDate(): LocalDate

ExpenseTracker

- expenses: List<Expense>

+ addExpense(category: String, amount: double, date: LocalDate)
+ getTotal(): double
+ getTotalByCategory(): Map<String, Double>
+ getCategoryExtremes(): Optional<CategoryExtreme>
+ getExpenses(): List<Expense>

CategoryExtreme

- highestCategory: String
- highestAmount: double
- lowestCategory: String
- lowestAmount: double

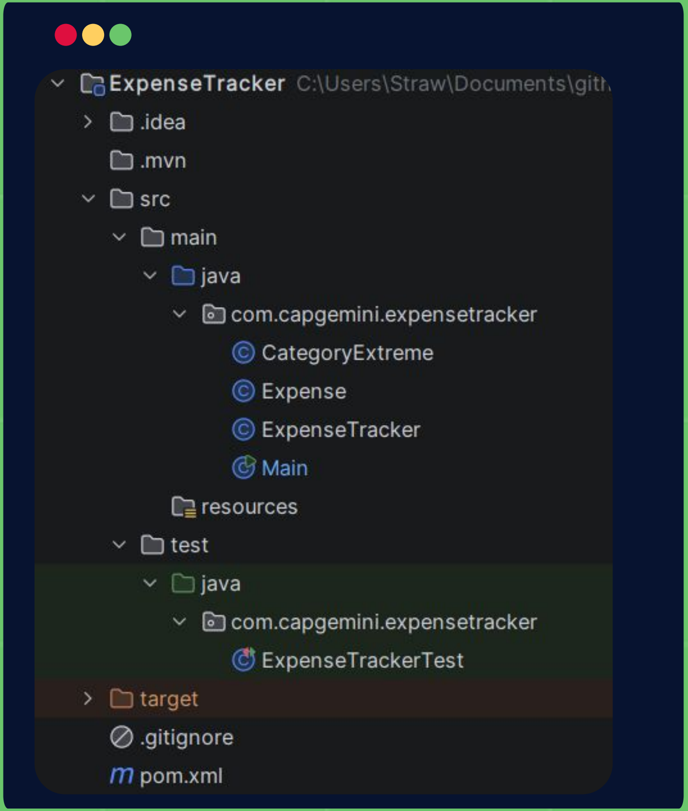
+ getHighestCategory(): String
+ getHighestAmount(): double
+ getLowestCategory(): String
+ getLowestAmount(): double

Main

+ main(args: String[])
+ addExpense(scanner: Scanner, expenseTracker: ExpenseTracker): void
+ showTotal(expenseTracker: ExpenseTracker): void
+ showTotalByCategory(expenseTracker: ExpenseTracker): void
+ showTrends(expenseTracker: ExpenseTracker): void
+ showHighestLowestCategory(expenseTracker: ExpenseTracker): void

- `Expense` holds data; `ExpenseTracker` manages and analyzes expenses.
- `Main` handles user input/output (CLI).
- Data flows: user → `Main` → `ExpenseTracker` → results.

Key Files and Folders



```
⌄ ExpenseTracker  C:\Users\Straw\Documents\git
  › .idea
    .mvn
  ⌄ src
    ⌄ main
      ⌄ java
        ⌄ com.capgemini.expensetracker
            CategoryExtreme
            Expense
            ExpenseTracker
            Main
        resources
    ⌄ test
      ⌄ java
        ⌄ com.capgemini.expensetracker
            ExpenseTrackerTest
  › target
    .gitignore
    pom.xml
```

Project structure showing main source code and test files

This project uses Maven structure.

Main Files:

- `Main.java`: The application's entry point.
- `ExpenseTracker.java`: Contains the core logic for expense handling.
- `Expense.java`: The data model for an individual expense.

Test Files: `ExpenseTrackerTest.java`: Contains the unit tests to verify the functionality.

Process to Run, Test, and Verify

Run the application

Locate the folder with the POM file in the command line.

```
mvn clean compile
mvn exec:java -Dexec.mainClass="com.capgemini.expensetracker.Main"
```

Run the tests and verify ExpenseTracker works

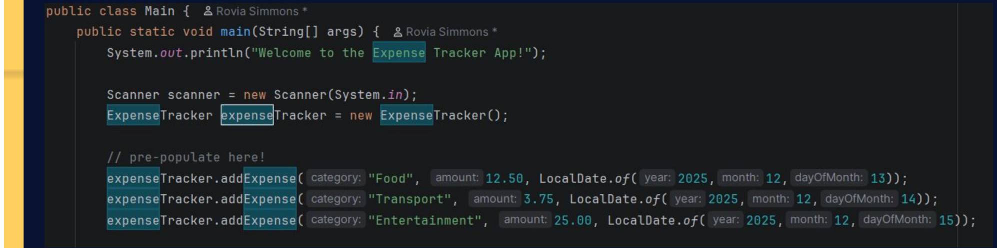
```
mvn clean compile
Mvn test
```

Add sample/seed data

Enter expenses via CLI prompts

Or pre-populate in Main.java:

Add sample/seed data



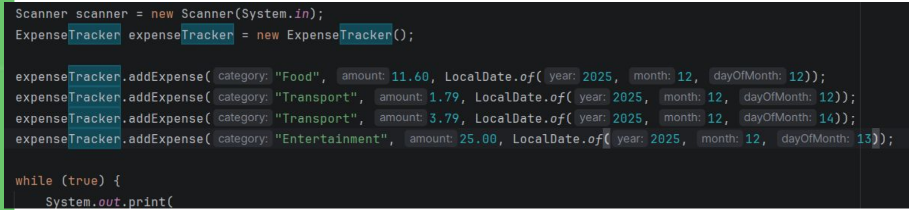
```
public class Main {  Rovia Simmons *
    public static void main(String[] args) {  Rovia Simmons *
        System.out.println("Welcome to the Expense Tracker App!");

        Scanner scanner = new Scanner(System.in);
        ExpenseTracker expenseTracker = new ExpenseTracker();

        // pre-populate here!
        expenseTracker.addExpense( category: "Food",  amount: 12.50, LocalDate.of( year: 2025, month: 12, dayOfMonth: 13));
        expenseTracker.addExpense( category: "Transport",  amount: 3.75, LocalDate.of( year: 2025, month: 12, dayOfMonth: 14));
        expenseTracker.addExpense( category: "Entertainment",  amount: 25.00, LocalDate.of( year: 2025, month: 12, dayOfMonth: 15));
```

Test Data and Seed Data

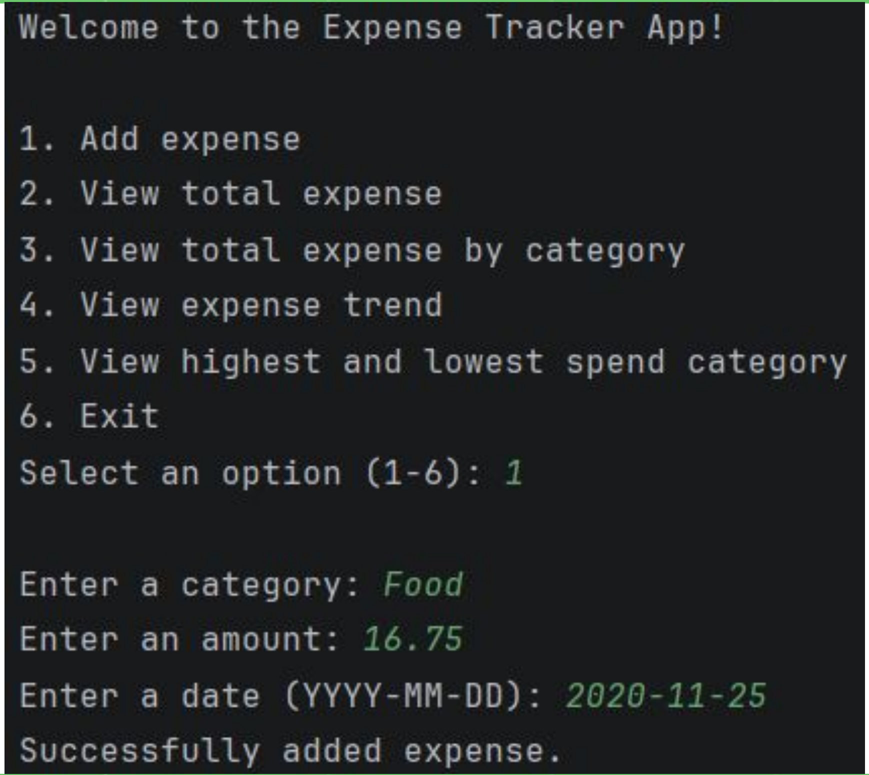
Initial seed for testing:

Food 11.60 2025-12-12
Transport 1.79 2025-12-12
Transport 3.79 2025-12-14
Entertainment 25.00 2025-12-13

Pre-populated Main.java:

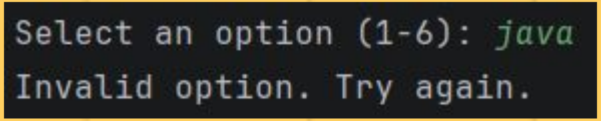


```
Scanner scanner = new Scanner(System.in);
ExpenseTracker expenseTracker = new ExpenseTracker();

expenseTracker.addExpense( category: "Food",  amount: 11.60, LocalDate.of( year: 2025,  month: 12,  dayOfMonth: 12));
expenseTracker.addExpense( category: "Transport",  amount: 1.79, LocalDate.of( year: 2025,  month: 12,  dayOfMonth: 12));
expenseTracker.addExpense( category: "Transport",  amount: 3.79, LocalDate.of( year: 2025,  month: 12,  dayOfMonth: 14));
expenseTracker.addExpense( category: "Entertainment",  amount: 25.00, LocalDate.of( year: 2025,  month: 12,  dayOfMonth: 13));

while (true) {
    System.out.print(
```

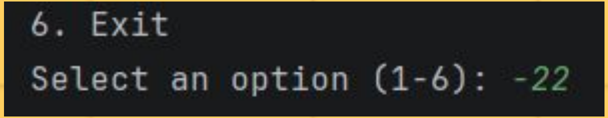


```
Welcome to the Expense Tracker App!

1. Add expense
2. View total expense
3. View total expense by category
4. View expense trend
5. View highest and lowest spend category
6. Exit
Select an option (1-6): 1

Enter a category: Food
Enter an amount: 16.75
Enter a date (YYYY-MM-DD): 2020-11-25
Successfully added expense.
```

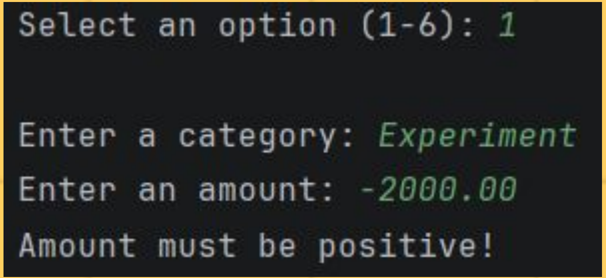
Test Data and Seed Data (Cont’d)



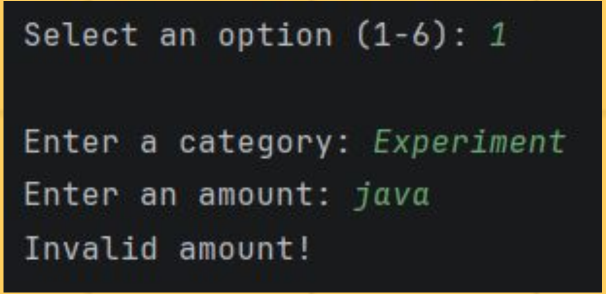
```
Select an option (1-6): java
Invalid option. Try again.
```



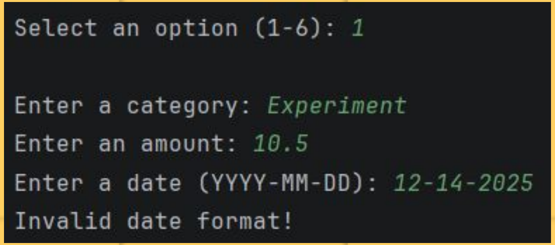
```
 6. Exit
 Select an option (1-6): -22
```



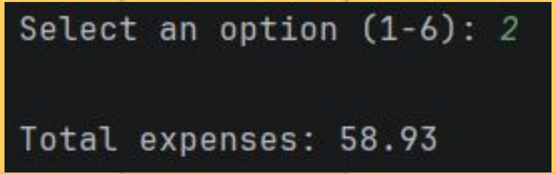
```
Select an option (1-6): 1


Enter a category: Experiment
Enter an amount: -2000.00
Amount must be positive!
```



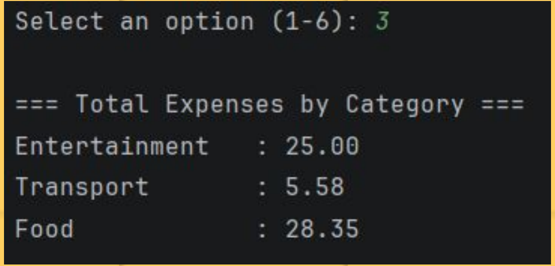
```
Select an option (1-6): 1


Enter a category: Experiment
Enter an amount: java
Invalid amount!
```



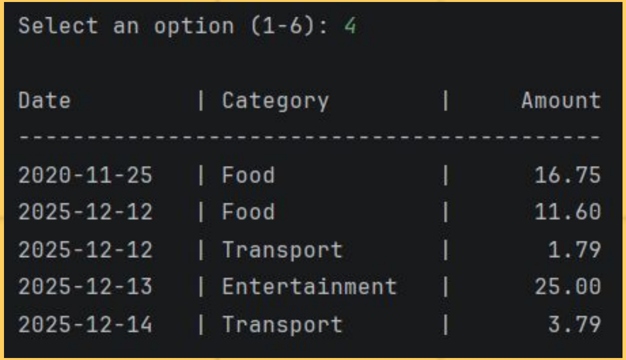
```
Select an option (1-6): 1

Enter a category: Experiment
Enter an amount: 10.5
Enter a date (YYYY-MM-DD): 12-14-2025
Invalid date format!
```



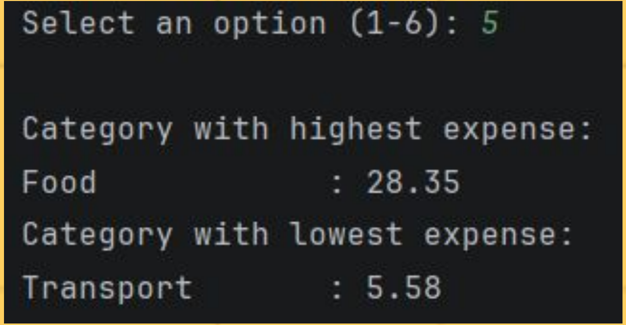
```
Select an option (1-6): 2


Total expenses: 58.93
```



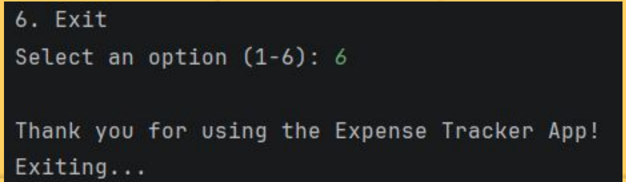
```
Select an option (1-6): 3


=== Total Expenses by Category ===
Entertainment    : 25.00
Transport        : 5.58
Food             : 28.35
```



```
Select an option (1-6): 4


Date          | Category          |      Amount
-----------------------------------------------
2020-11-25    | Food              |       16.75
2025-12-12    | Food              |       11.60
2025-12-12    | Transport         |        1.79
2025-12-13    | Entertainment     |       25.00
2025-12-14    | Transport         |        3.79
```



```
Select an option (1-6): 5


Category with highest expense:
Food                : 28.35
Category with lowest expense:
Transport           : 5.58
```



```
6. Exit
Select an option (1-6): 6


Thank you for using the Expense Tracker App!
Exiting...
```