Overview:

The Expense Tracker project is a simple console-based Java application designed to record, view, and analyze daily expenses.

using object-oriented programming (OOP) concepts, classes, objects, and methods, also using Java utilities like ArrayList and Scanner.

Methodology :

We divided the project in two Classes for clarity and to follow Object Oriented Programming.

* ExpenseTracker.java

⇒ This part contains :

→ Logic and operations to perform desired output.

→ The main method which is the entry point of program execution.

→ Gets expenses ➡️ Stores in an ArrayList ➡️ Shows the list of added expenses ➡️ Shows total expenses.

* Expense.java

⇒ This part of code provides :

→ Data representation for Expenses

→ Stores data fields

→ Getter methods to access internal data.

→ toString() method to print an Expense object.

Implementation :

⇒Tools and Environment -

1. Programming language : Java
2. IDE : Apache Netbeans

⇒ Created a project in apache netbeans named ExpenseTracker.

⇒ In the main class, I designed the desired output interface first.

⇒ Expense class to design a single expense.

⇒ ExpenseTracker main method to help users to interact and manage the list of expenses.

Code Implementation :

⇒ Expense.java(data model)

…

*public class Expense {*

*String description;*

*double amount;*

*String date;*

… → declare a public class named Expense, so that we can access the class from anywhere.

→ declare variables with type and name to make objects of class Expense.

…

*public Expense ( double amount, String description, String date){*

*this.amount = amount;*

*this.description = description;*

*this.date = date;*

}

… → Constructor to initialize new objects.

→ Access the value of parameters to the new objects.

…

public double getAmount(){

return amount;

}

public String getDescription(){

return description;

}

public String getDate(){

return date;

}

… → Getter methods to access private data.

→ getAmount returns the amount of Expense.

→ getDescription returns the description of Expense.

→ getDate returns the date of Expense.

…

@Override

public String toString() {

return date + " | " + description + " | $" + amount;

// toString() method to display expense details

}

…→ Overriding a superclass method, accessible from anywhere in the program.

⇒ ExpenseTracker.java

…

public class ExpenseTracker {

static ArrayList<Expense> myExpenses = new ArrayList<>();

…→ Declare the main class.

→ A list of arrays that can grow, stores the value of the Expense object added by the user, into myExpenses.

…

public static void main(String[] args) {

System.out.println("ExpenseTracker Started");

Scanner scanner = new Scanner(System.in);

boolean running = true;

while (running) {

System.out.println("\n>>>>> Expense Tracker <<<<<");

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

System.out.println("2. View Expenses");

System.out.println("3. Show Total Expense");

System.out.println("4. Exit")

…→ Main method to execute the program.

→ Creates objects to read user input.

→ Print program console view after execution.

…

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

int choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

addExpense(scanner);

break;

case 2:

viewExpenses();

break;

case 3:

showTotalExpense();

break;

case 4:

running = false;

break;

default:

System.out.println("Invalid option!");

}

}

scanner.close();

}

… → Take user input as a choice.

→ Controls which operation to run based on user choise.

→ If you choose case 1, the program calls addExpense method and after the task of method is done break statement stops the execution before case 2 runs, and returns to main menu until the user choose case 4.

…

private static void addExpense(Scanner scanner) {

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

String description = scanner.nextLine();

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

double amount = scanner.nextDouble();

scanner.nextLine(); // consume newline

System.out.print("Enter date (YYYY-MM-DD): ");

String date = scanner.nextLine();

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

expenses.add(expense);

System.out.println("Expense added successfully!");

}

…→ Method to ask users to add a description,amount and expense date.

→ Created an object of Expense, store the value of created object Expense to anExpense.

…

private static void viewExpenses() {

if (expenses.isEmpty()) {

System.out.println("No expenses recorded.");

} else {

for (Expense expense : expenses) {

System.out.println(expense);

}

}

}

…→ Method to view all Expenses stored in anExpense read from users.

…

private static void viewExpenses() {

if (expenses.isEmpty()) {

System.out.println("No expenses recorded.");

} else {

for (Expense expense : expenses) {

System.out.println(expense);

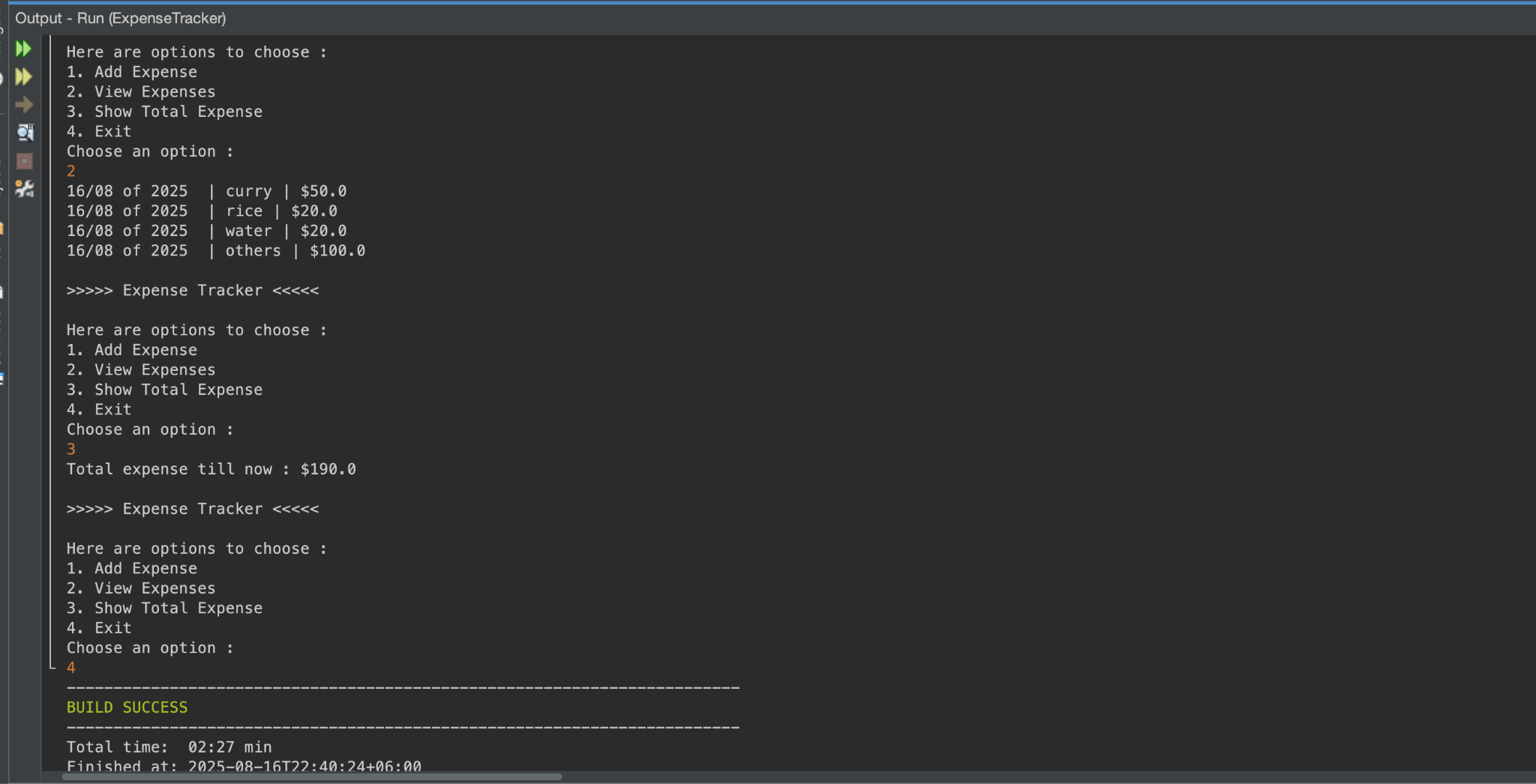
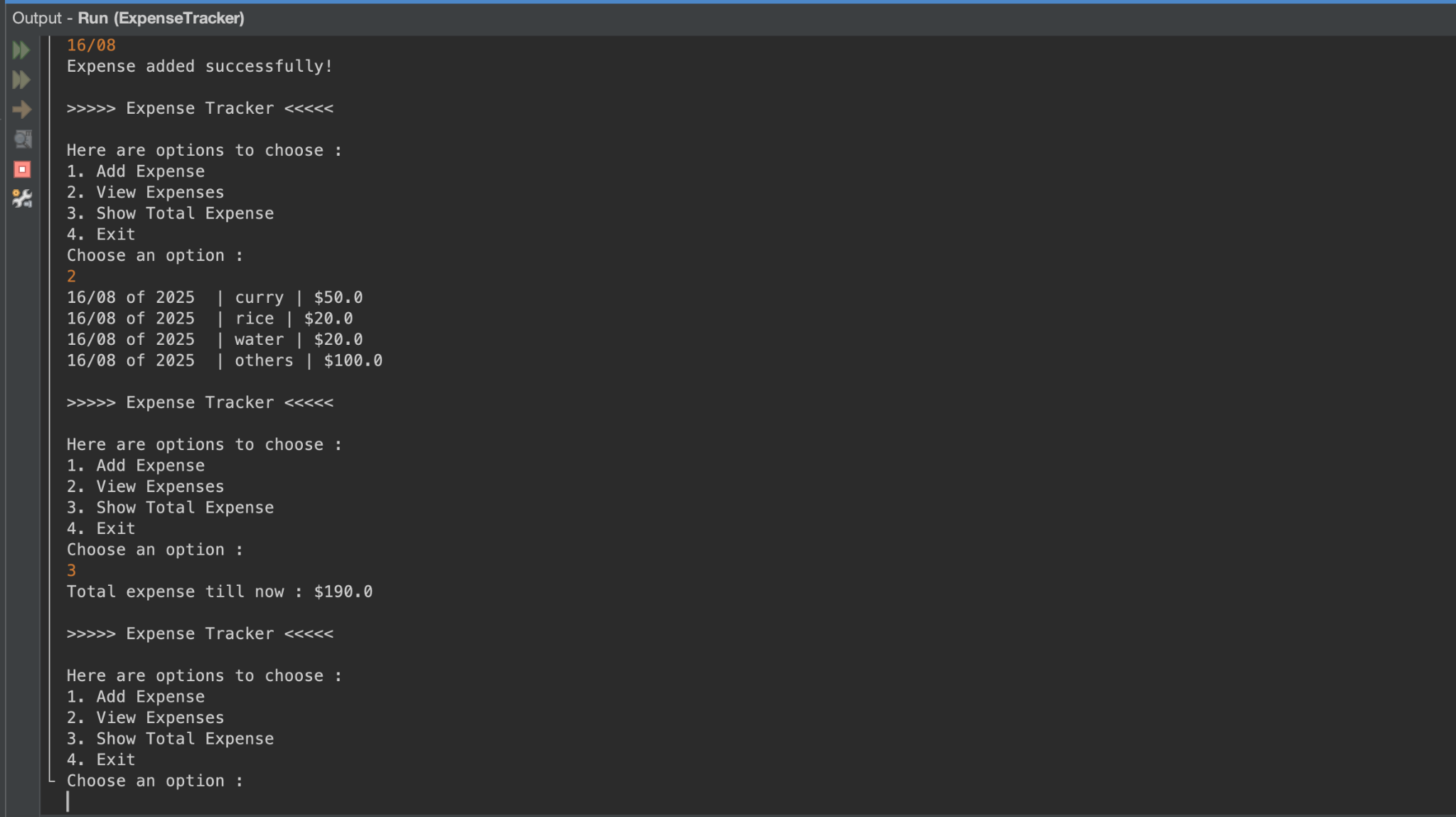
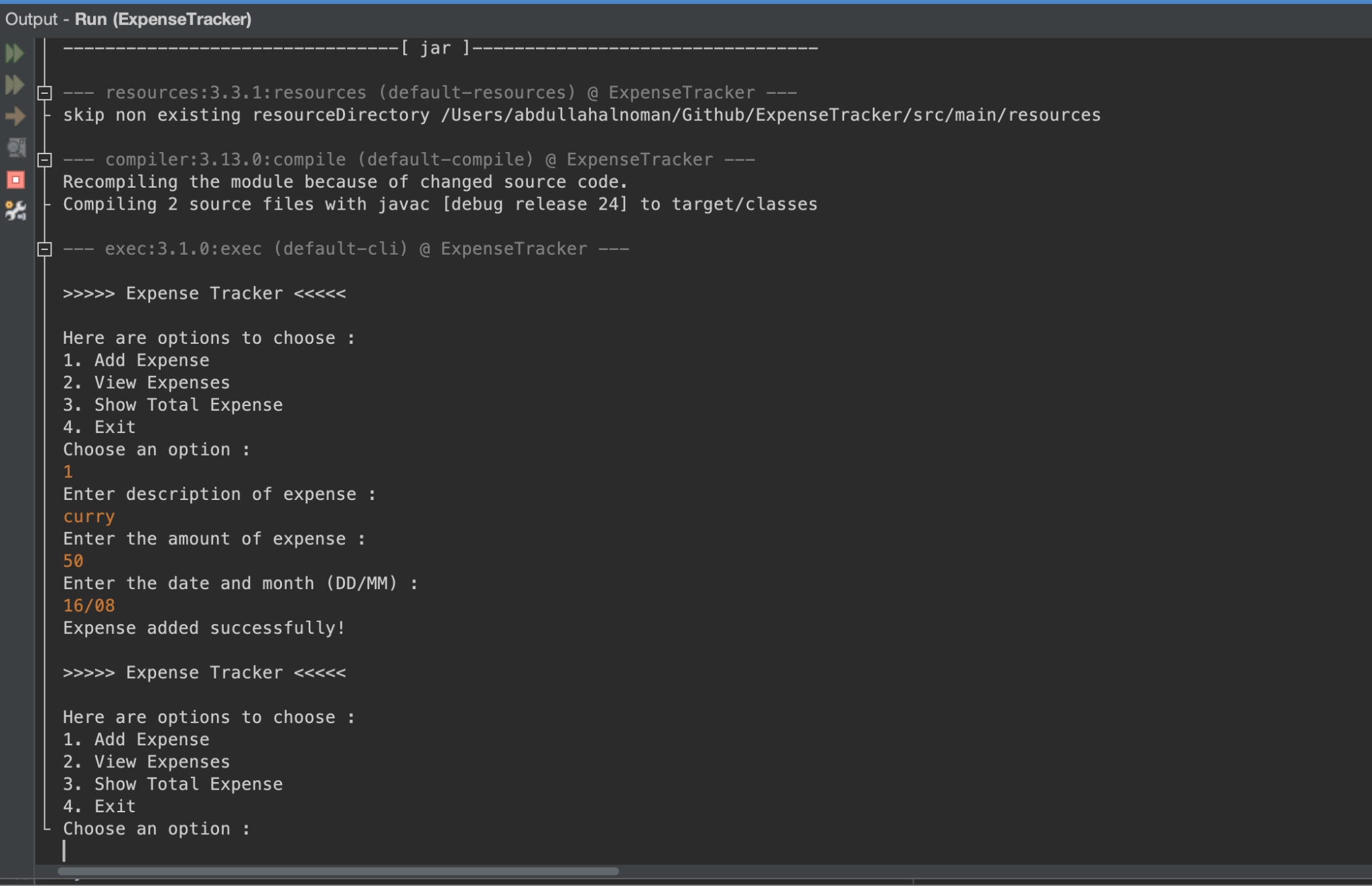
}

}

}

…→Method to loop through each expense that was added by the user and add them after that show the total expense amount of the user.

Test and Result :



Conclusion / Future work :

⇒ Store expenses with real time and date with java package Date.

⇒ Store expenses list into a file using java file I/O.