
# EXPENSE CALCULATOR PROJECT

## FINAL REPORT

CSE228: DATA STRUCTURES

SUBMITTED BY: SUDAM LOKESHWAR
ROLL NUMBER:72
SECTION : K22UG
REG.NO: 12210661

MENTOR:AMAN KUMAR
MENTOR ID: 63642

DATE OF SUBMISSION : 23|10|2023

# ACKNOWLEDGEMENT

The project is related to the Expenses Calculations named as “Expense Calculator System”, this report is intended to provide a summary of the progress of the expense calculator project. The scope of this report is limited to the basic functionality of the expense calculator, which includes the ability to add and display expenses and calculate the total amount of expenses.

I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful completion of this data structure project on the expense calculator.

First and foremost, I am deeply thankful to my project supervisor Mr. Aman Kumar sir for their invaluable guidance and unwavering support throughout the project. Their expertise and insights have been instrumental in shaping this project and enhancing my understanding of data structures.

[Sudam Lokeshwar]
[Lovely Professional University]
Date: 23|10|23

# Table of Contents

# 1. Introduction

### 1.1 Background

In an era of increasing financial awareness and the need for responsible budgeting, managing personal expenses is paramount. To help individuals gain a clearer understanding of their spending habits, we present an Expense Calculator, a user-friendly tool that empowers users to keep track of their daily expenses and calculate their total spending. This application serves as a valuable financial companion, allowing users to maintain, organize, and evaluate their financial data effortlessly.

The Expense Calculator is designed to provide a simple yet effective solution for anyone looking to monitor their expenditures and maintain better control over their finances. This versatile tool leverages the power of technology to streamline the process of expense management and offers a clear and concise breakdown of total spending.

### 1.2 Purpose of the Report

The purpose of this report is to introduce the Expense Calculator, a tool for managing daily expenses and calculating total spending. It outlines the application's key functionalities, provides development guidance, and emphasizes the importance of responsible financial management. The report aims to inform and educate readers about the benefits of using the Expense Calculator, encouraging better financial awareness and budgeting practices.

### 1.3 Structure of the Report

The structure of this report is organized into several sections, each of which contributes to a holistic understanding of the "Expense Calculator" project. These sections include an introduction, objectives and scope, methodology, a summary of key features, a conclusion, a bibliography, and annexure. Each section is tailored to offer insights into various facets of the project's development and functionality.

# 2. Objectives of the project

### 2.1 **Objectives of the Expense Calculator Project**

The objectives of the "Expense Calculator" project are aimed at creating a practical and user-friendly tool for managing daily expenses and promoting financial awareness. The primary objectives include:

1)To design a calculator that accurately calculates and displays the total spending, helping users gain insights into their financial habits.
2)User-Friendly Interface: To ensure a user-friendly interface that simplifies the process of inputting daily expenses and provides clear, accessible results.
3)Promote Financial Responsibility: To promote financial awareness and responsibility, enabling users to understand their spending patterns and make informed decisions about their finances.

# 3. Scope of the Calculator

The scope of the Expense Calculator project encompasses managing daily expenses, real-time calculation of total spending, providing a user-friendly interface, customization options, data security, and an educational component to encourage financial responsibility and awareness.

### 3.1. **Application Tools**

The development of the "Expense Calculator" project was made possible through the use of specific tools and technologies. These tools include:

The Java programming language, chosen for its portability and flexibility.
An Integrated Development Environment (IDE) [Specify the IDE used] for code development and debugging.
Version control system (e.g., Git) for collaborative development and code versioning.
A text editor for creating documentation and textual content related to the project.

The combination of these tools contributed to a streamlined development process and facilitated the creation of a functional expense calculator.

# 4. Methodology

Methodology for developing an expense calculator

**4.1 Requirements gathering**

The first step is to gather requirements for the expense calculator. This includes understanding what the calculator should be able to do and what features are important to users. Some key requirements include:

The ability to input daily expenses

The ability to calculate total spending

The ability to manage financial data

The ability to display total expenses in a clear and concise way

**4.2 Design**

Once the requirements have been gathered, the next step is to design the expense calculator. This includes deciding on the overall structure of the calculator, the user interface, and the algorithms that will be used to calculate total spending.

One way to design the expense calculator is to use an object-oriented approach. This would involve creating a class to represent the expense calculator and classes to represent the different types of data that the calculator will store, such as daily expenses and total spending.

The algorithms used to calculate total spending should be accurate and efficient. One common approach is to use a simple accumulator to add up the daily expenses.

**4.3 Implementation**

Once the design is complete, the next step is to implement the expense calculator. This involves writing code to implement the classes and algorithms that were designed in the previous step.

If you are using an object-oriented approach, you will need to write code to create the expense calculator class and the classes to represent the different types of data that the calculator will store. We will also need to write code to implement the algorithms for calculating total spending.

# 5. Source Code

```java
package k22ug;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Scanner;

public class ExpenseCalculator {

    private HashMap<String, Double> expenses;

    public ExpenseCalculator() {
        this.expenses = new HashMap<>();
    }

    public void addExpense(String date, double amount) {
        if (isValidDateFormat(date)) {
            this.expenses.put(date, this.expenses.getOrDefault(date, 0.0) + amount);
        } else {
            System.out.println("Invalid date format. Please use YYYY-MM-DD.");
        }
    }

    public void modifyExpense(String date, double newAmount) {
        if (isValidDateFormat(date)) {
            if (this.expenses.containsKey(date)) {
                this.expenses.put(date, newAmount);
                System.out.println("Expense modified successfully.");
            } else {
                System.out.println("Expense not found for the given date.");
            }
        } else {
            System.out.println("Invalid date format. Please use YYYY-MM-DD.");
        }
    }

```

```
37
38      public void deleteExpense(String date) {
39          if (isValidDateFormat(date)) {
40              if (this.expenses.containsKey(date)) {
41                  this.expenses.remove(date);
42                  System.out.println("Expense deleted successfully.");
43              } else {
44                  System.out.println("Expense not found for the given date.");
45              }
46          } else {
47              System.out.println("Invalid date format. Please use YYYY-MM-DD.");
48          }
49      }
50
51      public double calculateTotalExpenses() {
52          return this.expenses.values().stream().mapToDouble(Double::doubleValue).sum();
53      }
54
55      public void displayExpenses() {
56          for (String date : this.expenses.keySet()) {
57              System.out.println(date + ": $" + this.expenses.get(date));
58          }
59          System.out.println("Total Expenses: $" + calculateTotalExpenses());
60      }
61
62      private boolean isValidDateFormat(String date) {
63          SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
64          dateFormat.setLenient(false); // To make the parsing strict
65
66          try {
67              Date parsedDate = dateFormat.parse(date);
68              return parsedDate != null;
69          } catch (ParseException e) {
70              return false;
71          }
72      }
73
74      public static void main(String[] args) {
75          Scanner scanner = new Scanner(System.in);
76          ExpenseCalculator calculator = new ExpenseCalculator();
```

```
77
78          while (true) {
79              System.out.println("\nExpense Calculator");
80              System.out.println("1. Add an expense");
81              System.out.println("2. Modify an expense");
82              System.out.println("3. Delete an expense");
83              System.out.println("4. Display expenses");
84              System.out.println("5. Exit");
85              System.out.print("Enter your choice: ");
86
87              int choice = scanner.nextInt();
88              scanner.nextLine(); // Consume newline left-over
89
90              switch (choice) {
91                  case 1:
92                      System.out.print("Enter the date (YYYY-MM-DD): ");
93                      String addDate = scanner.nextLine();
94                      System.out.print("Enter the amount: ");
95                      double addAmount = scanner.nextDouble();
96                      calculator.addExpense(addDate, addAmount);
97                      break;
98                  case 2:
99                      System.out.print("Enter the date to modify (YYYY-MM-DD): ");
100                     String modifyDate = scanner.nextLine();
101                     System.out.print("Enter the new amount: ");
102                     double newAmount = scanner.nextDouble();
103                     calculator.modifyExpense(modifyDate, newAmount);
104                     break;
105                 case 3:
106                     System.out.print("Enter the date to delete (YYYY-MM-DD): ");
107                     String deleteDate = scanner.nextLine();
108                     calculator.deleteExpense(deleteDate);
109                     break;
```

```
                case 4:
                    calculator.displayExpenses();
                    break;
                case 5:
                    System.out.println("Goodbye!");
                    return;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

Here's an explanation of each part of the code:
This Java code defines a simple program called ExpenseCalculator that allows a user to manage and calculate their expenses. It uses a HashMap to store expenses by date and provides a menu-driven interface for adding expenses, displaying expenses, and exiting the program.

Here's a breakdown of the code:

Import Statements:
import java.util.HashMap;: This imports the HashMap class from the java.util package.
import java.util.Scanner;: This imports the Scanner class from the java.util package, which is used for user input.

ExpenseCalculator Class:
The ExpenseCalculator class is the main class of the program.
It has a private field expenses, which is a HashMap that stores expenses with dates as keys and amounts as values.

Constructor:
The constructor ExpenseCalculator() initializes the expenses HashMap when an object of the class is created.

addExpense Method:
The addExpense method is used to add an expense to the expenses HashMap.
It takes two parameters: date (as a string in the format "YYYY-MM-DD") and amount (a double representing the expense amount).
If an expense with the given date already exists, it adds the new amount to the existing value. If not, it creates a new entry.

calculateTotalExpenses Method:
The calculateTotalExpenses method calculates and returns the total expenses by summing up all the values (amounts) in the expenses HashMap.

displayExpenses Method:
The displayExpenses method is used to print all expenses along with their dates and the total expenses to the console.

Modify Expense:
The modifyExpense method allows users to modify an existing expense by specifying a date and a new amount.

Delete Expense:
The deleteExpense method allows users to delete an existing expense by specifying a date.

main Method:
The main method is the entry point of the program.
It creates a Scanner object for user input and an ExpenseCalculator object.
It runs a loop that displays a menu with three options:
1)Add an expense
2)Display expense
3)Modify expense
4)Delete expense
5)Exit
It reads the user's choice and performs actions accordingly:
For option 1, it prompts the user to enter a date and an amount, and then it adds the expense using the addExpense method.
For option 2, it displays the expenses using the displayExpenses method.
For option 3, it modify the expenses using the ModifyExpenses method.
For option 4, it delete the expenses using the DeleteExpenses method.
For option 5, it exits the program.
If the user enters an invalid choice, it informs them.
The loop continues until the user chooses to exit.

The program provides a simple interface for users to keep track of their expenses and see a summary of their total expenses.

# 6. Summary

The "Expense Calculator" project is a Java program that helps users manage and calculate their expenses. It includes the following key features:

Expense Tracking: Users can input expenses along with their corresponding dates (in the format "YYYY-MM-DD") and amounts. The program stores this information in a HashMap.

Total Expense Calculation: The program can calculate and display the total expenses by summing up all the stored expense amounts.

User-Friendly Menu: The program provides a menu-driven interface for users with three options: adding an expense, displaying expenses, and exiting the program.

Data Storage: Expenses are stored in a HashMap, allowing for efficient retrieval and updates.

Input Validation: The program handles user input, ensuring that invalid choices are rejected, and it also consumes any leftover newline characters.

Interactive Loop: The program operates within a loop that continues until the user chooses to exit, making it easy for users to manage their expenses over multiple sessions.

# 7.TEST CASES

## Sample test case 1 :

Expense Calculator

1. Add an expense

2. Modify an expense

3. Delete an expense

4. Display expenses

5. Exit

Enter your choice: 1

Enter the date (YYYY-MM-DD): 2023-11-07

Enter the amount: 553

Expense Calculator

1. Add an expense

2. Modify an expense

3. Delete an expense

4. Display expenses

5. Exit

Enter your choice: 1

Enter the date (YYYY-MM-DD): 2023-05-19

Enter the amount: 2424

Expense Calculator

1. Add an expense

2. Modify an expense

3. Delete an expense

4. Display expenses

5. Exit

Enter your choice: 4

2023-05-19: $2424.0

2023-11-07: $553.0

Total Expenses: $2977.0

## Sample Test Case 2:

Expense Calculator

1. Add an expense
2. Modify an expense
3. Delete an expense
4. Display expenses
5. Exit

Enter your choice: 2

Enter the date to modify (YYYY-MM-DD): 2023-11-07

Enter the new amount: 255

Expense modified successfully.

Expense Calculator

1. Add an expense
2. Modify an expense
3. Delete an expense
4. Display expenses
5. Exit

Enter your choice: 4

2023-05-19: $2424.0

2023-11-07: $255.0

Total Expenses: $2679.0

Expense Calculator

1. Add an expense
2. Modify an expense
3. Delete an expense
4. Display expenses
5. Exit

Enter your choice: 5

Goodbye!

# 8.CONCLUSION

In conclusion, the "Expense Calculator" project offers a practical and user-friendly solution for managing and tracking personal expenses. It provides a convenient way for users to input, store, and calculate their expenses, all within a straightforward menu-driven interface. This project can be a valuable tool for individuals looking to maintain a record of their financial transactions, as it efficiently organizes and displays expenses, ultimately helping users make informed financial decisions.