Individual Project CMPE202 - Harika Kolipaka

GitHub Classroom: https://github.com/gopinathsjsu/individual-project-harikakolipaka22k.git

Student ID: 017422912

Overview

This project focuses on the validation and identification of credit card types from various data sources. The primary goal is to process credit card data, ensuring the validity of credit card numbers and correctly categorizing them into types such as Visa, MasterCard, American Express, and Discover.

Objective

1. **Primary Objective**: Develop a system to validate credit card numbers and identify the card issuer from a given dataset.
2. **Secondary Objective**: Implement a design that is adaptable and extensible for future additions of credit card classes and types.

Design Patterns

To achieve these objectives, the following design patterns are employed:

Strategy Design Pattern

The Strategy Pattern is a behavioral design pattern that enables an object to change its behavior at runtime. It encapsulates algorithms in separate classes following a common interface strategy.

Implementation Details:

- **Interfaces**:
  - `Reader`: Handles reading of different file formats.
  - `Writer`: Manages writing data to various file formats.
  - `CreditCardHandler`: Determines the type of credit card.
- This pattern allows the application to adapt its processing logic depending on the file format and credit card type, enhancing flexibility and maintainability.

Chain of Responsibility Design Pattern

This pattern creates a chain of receiver objects for a request. This pattern decouples sender and receiver of a request based on the type of request.

Implementation Details:

- **Main Credit Card Handler**: Acts as a central point for processing files, delegating the validation task to specific credit card handlers (Visa, MasterCard, etc.).

- Each handler in the chain is responsible for certain validation criteria, creating a scalable and robust validation mechanism.

Consequences of the Design Choices

Chain of Responsibility

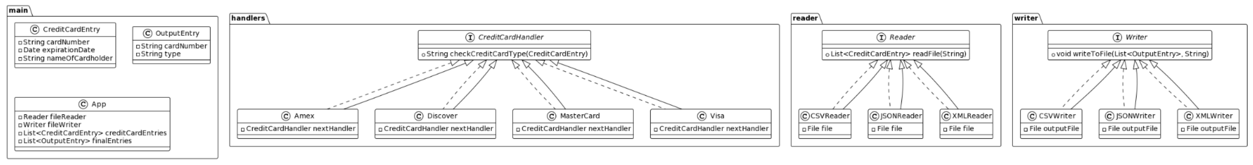
- **Advantages**:
  - Promotes decoupling of request processing.
  - Enhances flexibility in changing the order and adding new handlers.
- **Disadvantages**:
  - Complexity in debugging and tracking the processing path.
  - Possibility of requests falling through without proper handling.

Strategy Design Pattern

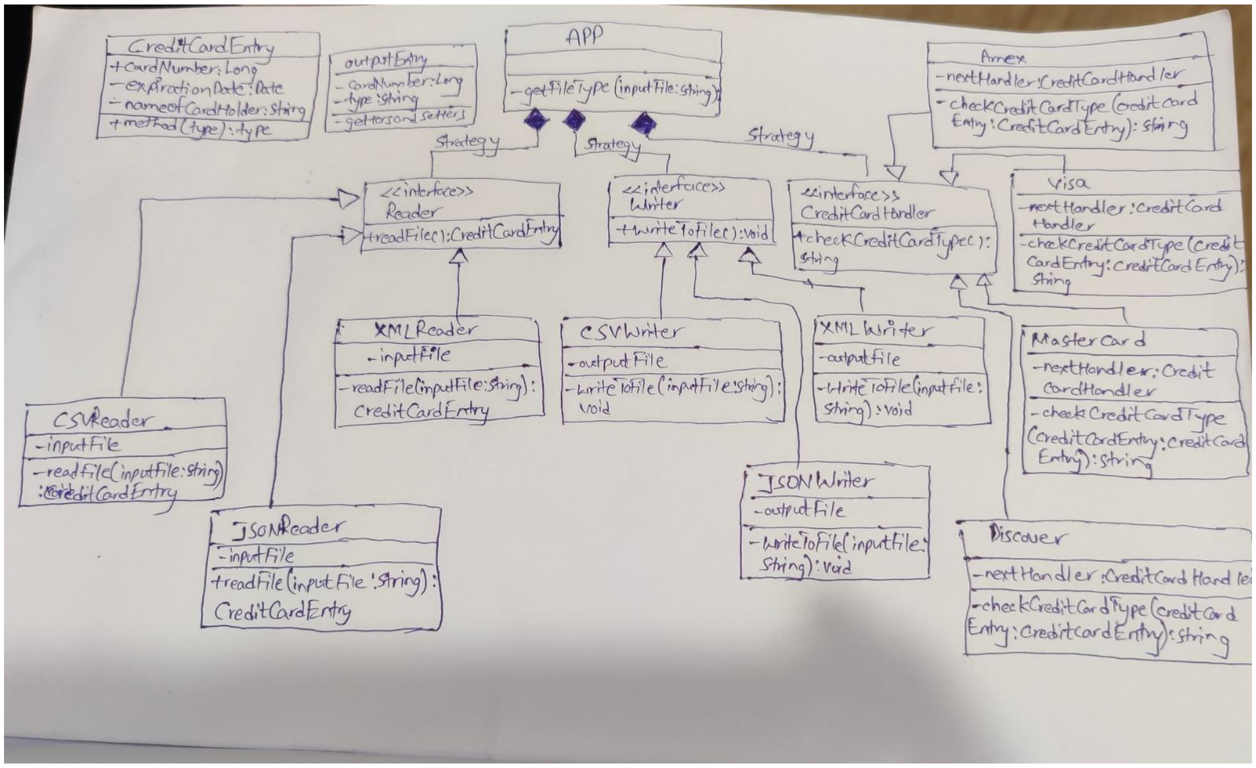
- **Advantages**:
  - Increases flexibility in implementing various algorithms.
  - Promotes code re-usability and modular architecture.
- **Disadvantages**:
  - Requires clients to be aware of different strategies.
  - Can lead to a proliferation of strategy classes.

Class Diagram

Class Diagram:



Class Diagram 2:


CreditCardEntry
+CardNumber: Long
-expirationDate: Date
-nameofCardHolder: String
+method(type): type
outputEntry
-CardNumber: Long
-type: String
-getters and setters
APP
-getFileType(inputFile: String)
Strategy
Strategy
Strategy
<<interface>> Reader
+readFile(): CreditCardEntry
<<interface>> Writer
+writeToFile(): void
<<interface>> CreditCardHandler
+checkCreditCardType(): String
Amex
-nextHandler: CreditCardHandler
-checkCreditCardType(creditCardEntry: CreditCardEntry): String
Visa
-nextHandler: CreditCardHandler
-checkCreditCardType(creditCardEntry: CreditCardEntry): String
XMLReader
-inputFile
-readFile(inputFile: String): CreditCardEntry
CSVWriter
-outputFile
-writeToFile(inputFile: String): void
XMLWriter
-outputFile
-WriteToFile(inputFile: String): void
MasterCard
-nextHandler: CreditCardHandler
-checkCreditCardType(CreditCardEntry: CreditCardEntry): String
CSVReader
-inputFile
-readFile(inputFile: String): CreditCardEntry
JSONReader
-inputFile
+readFile(inputFile: String): CreditCardEntry
JSONWriter
-outputFile
-WriteToFile(inputFile: String): void
Discover
-nextHandler: CreditCardHandler
-checkCreditCardType(creditCardEntry: CreditCardEntry): String

Usage Instructions

Running the Project

1. Launch `App.java`.
2. Input the paths for the input and output files, including their extensions.

JUnit Testing

- Execute the provided JUnit test cases to validate the functionality.