Individual Project

Part 1

1. Describe what is the primary problem you try to solve.

Firstly the program should correctly identify the type of credit card based on the card number. Different cards have different numbering system and the program should be able to clearly distinguish the card.

Once the type of credit card has been identified, I will have to create an instance (object) corresponding to the class (VisaCC, MasterCC, etc).

2. Describe what are the secondary problems you try to solve (if there are any).

First of all, I will have to extract the data present in the CSV file (or other input files in Part 2) correctly. I will have to make a program that can correctly interpret the input file to extract relevant details.

Also, the design should be flexible enough so that we can include new credit card types or different input file formats in the future.

3. Describe what design patterns you will use and how (use plain text and diagrams).

In part-1, I face two problems - one is to figure how I can figure out the type of credit card a specific record in about and the other is how I will create the appropriate objects.

Therefore, I have to pick one creational design pattern to create an instant (object) and a behavioral design pattern to determine the type of the object.

For the creational design pattern, I have decided to use Factory Pattern and for the behavioral pattern, the Chain of Responsibility Pattern.

How I will use these patterns can be illustrated by the class diagram (class_diagram_part1) in the repository.

4. Describe the consequences of using this/these patterns.

There are various positive and negative consequences of using Factory Pattern and Chain of Responsibility pattern. They are as follows.

Positive consequences

I. Both design patterns promote decoupling which improves the paintability and reusability of the code.
II. The chain of responsibility will allow me to add or remove handlers without altering the code of existing handlers.
III. The factory pattern makes it easier to create and change the way objects are created.
IV. It will be easier to introduce new types of credit cards by adding new factories and handlers without changing the code.
V. The handlers and factories can be reused across the program
VI. Chain of responsibility gives me control over how a request is processed through the chain.

Negative Consequences

I. The complexity of writing the program is increased due to incorporating multiple design patterns
II. If the chain in chain of responsibility is too long and the best handler is at the end of the chain, this can lead to slower performance.

Part 2

I have used the adapter design pattern to parse different input file formats (json, xml, csv). I have also changed the program to produce a file with the same extension as the input file formats and pass the output in the produced file. I have included the class diagram in my GitHub repository. The diagram consists of the classes and design patterns I used for both Part-1 and Part-2.

Part 3

I have used IntelliJ as the IDE. My java version is 19.0.1 and I had to use org.json as a dependency to parse JSON files. The Java code and unit tests are in the Individual-Project-Code folder. The java files that contain the unit tests start with the prefix ‘Test’.