CMPE 202 Software Systems Engineering

Individual Project

Name: Mahek Virani

SJSU ID:017446936

Deliverables:

Describe what is the primary problem you try to solve.

· Read credit card records from a file, validate the card number, determine the card issuer (Visa, MasterCard, American Express, Discover), and create instances of the corresponding credit card class.

Describe what are the secondary problems you try to solve.

Extensibility for Additional Card Types: The system should be designed to easily accommodate new credit card types.

Validation of Card Numbers: Ensure that card numbers do not exceed 19 digits and adhere to issuer-specific formats.

Describe what design pattern(s) you use how

1. Iterator Pattern:

• For a method of sequentially accessing a collection object's elements, use the iterator pattern.

• Each kind of file has several records, hence this pattern aids in processing each record by iterating through each record.

2. Factory Method (Creational Pattern):

- Usage: Implement a CreditCardFactory class with a method createCreditCard that takes the credit card number and other details as parameters. This method will determine the type of credit card and instantiate the appropriate subclass.

3. **Strategy Pattern (Behavioral Pattern):**

- Usage: Define a strategy interface for validating credit card numbers. Implement different strategies for each credit card type
- Due to the variety of file types, specific objects are created for each type and appropriate methods are employed depending on the input file.
- To support various file formats, I employed a strategy design pattern.
- CsvFileparser, Jsonfileparser, and Xmlfileparser are the three interfaces I've developed. A file parser is added to all of these. As a result of the input, the behavior modifies

Describe the consequences of using this/these patterns:

1. Iterator pattern:

· When working with collections, enables you to work at a higher level of abstraction.Disadvantages**:** Iterator is too much to handle for small applications, Iterator is too extravagant.

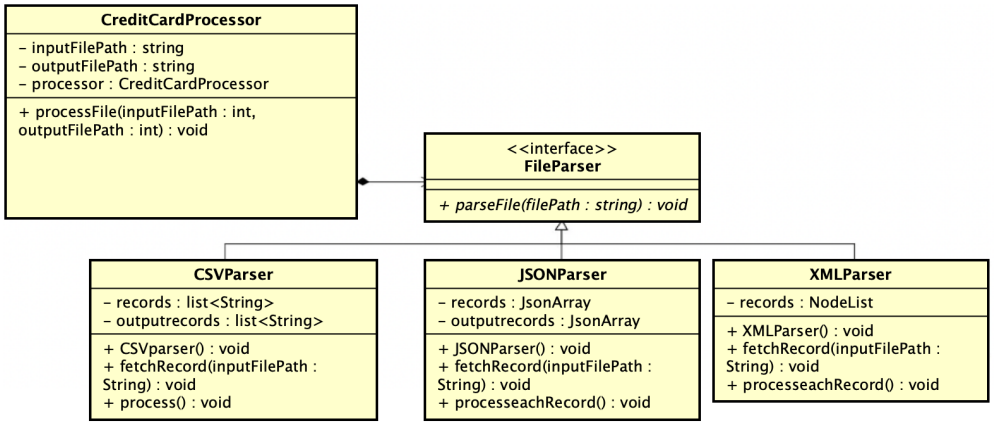
2. Strategy Pattern:

This pattern allows for flexible validation logic that can be swapped at runtime. It's easy to modify or add new validation strategies without altering the client code.

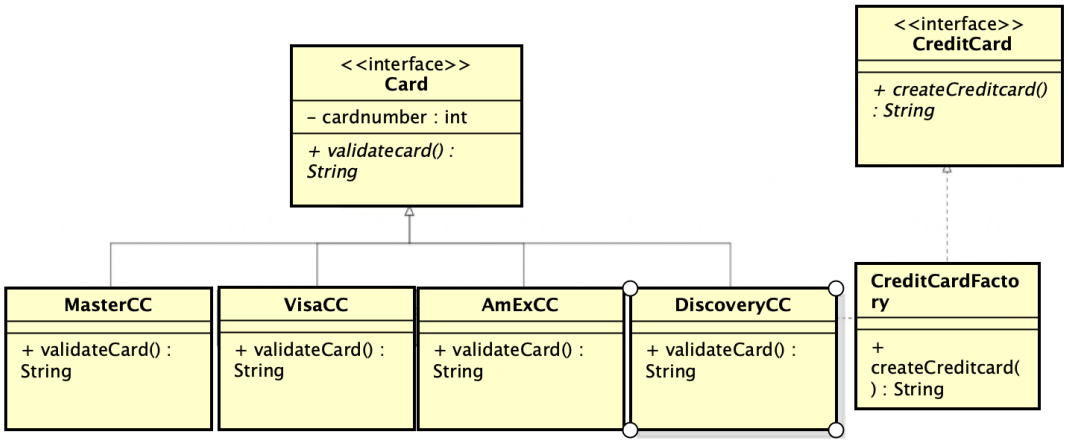
3. Factory Pattern:

This pattern encapsulates the creation logic, making it easy to add support for new card types in the future. The main code does not need to change when new credit card classes are added.

Strategy Pattern



Factory Design Pattern



Git link :https://github.com/gopinathsjsu/individual-project-mahekvirani-sjsu