CMPE 202 – Individual Project

Student Name: Abhilash Tayade
SJSU ID: 016730129

Describe what is the primary problem you try to solve.

Answer:

The primary problem is to design a system that can read a record from different type of files, validate the credit card number, and create an instance of the appropriate credit card class.

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

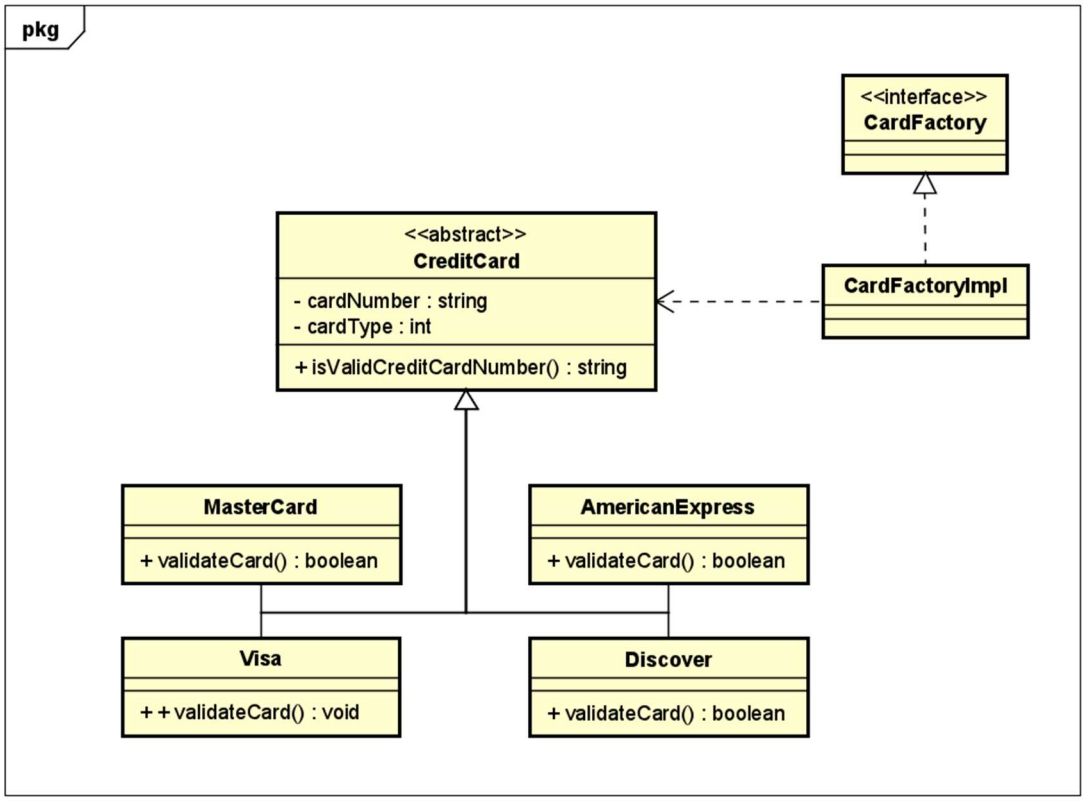
Answer:

1. Determining the card issuer based on the credit card number.
2. Handling invalid credit card numbers.

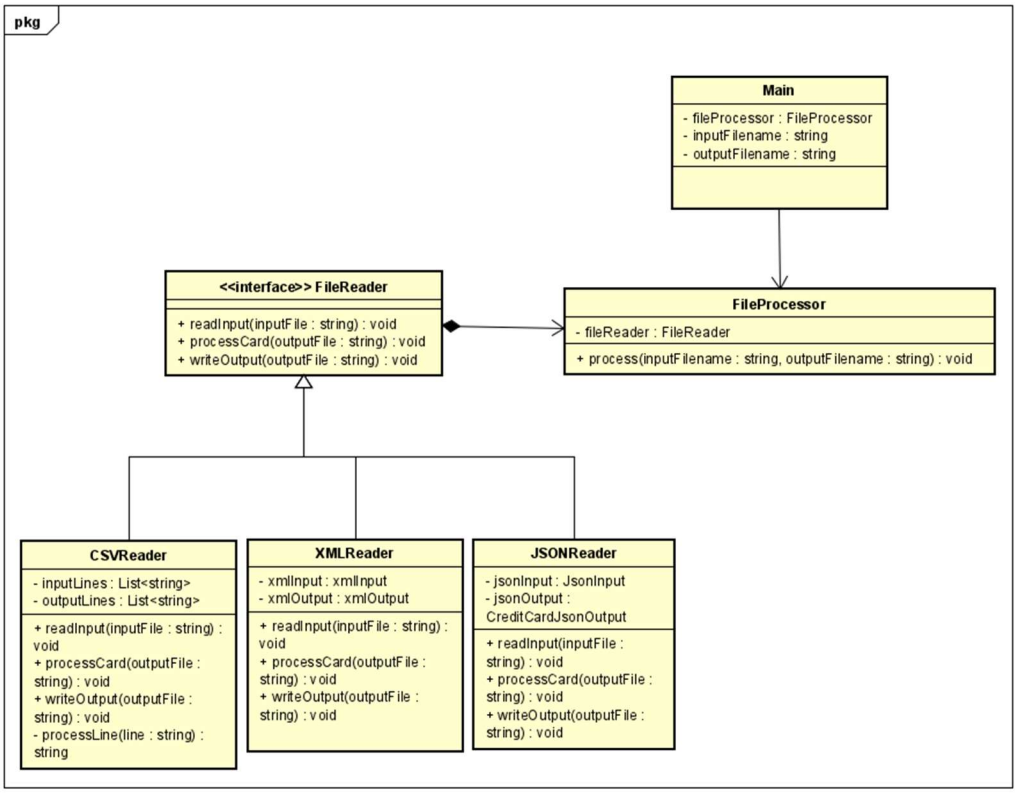
Describe what design pattern(s) you use how (use plain text and diagrams).

Answer:

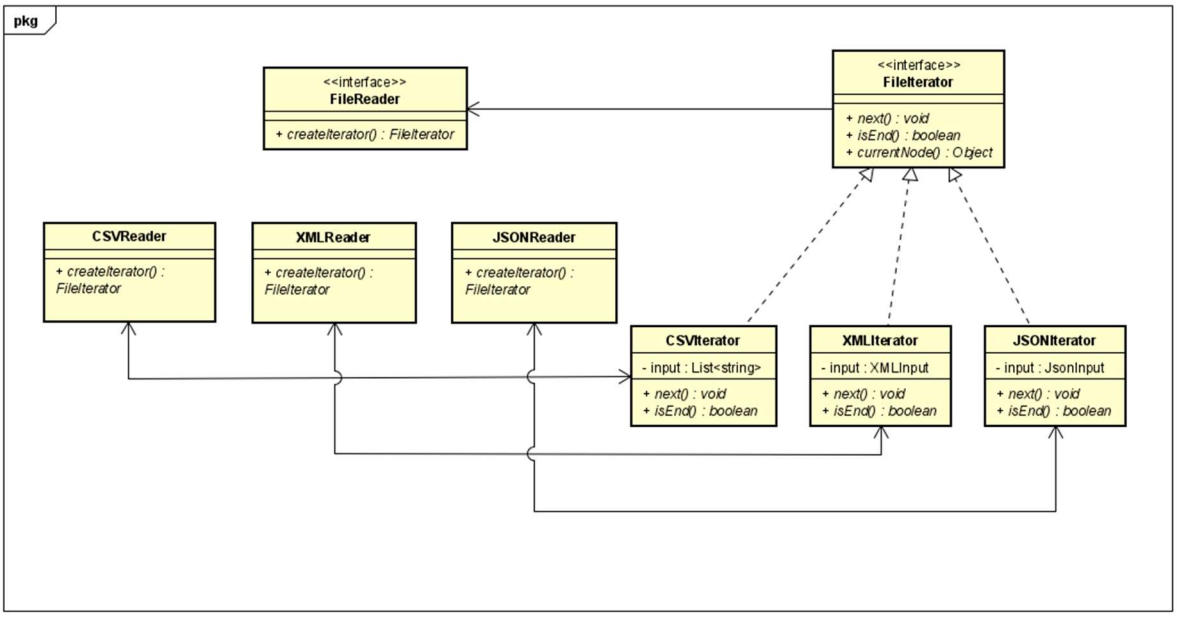
1. **Factory Pattern:** Because we have a superclass CreditCard and various subclasses such as VisaCC, MasterCC, AmExCC, and so on, the factory pattern will allow us to shift the burden of class instantiation from the client application to the factory class. We design a CardFactory that will generate card objects based on the input.



2. **Strategy Pattern:** We will use the strategy pattern to develop a family of algorithms to parse and read various file formats. The requirement here is to vary the algorithm used to read and process the file dependent on the file type. We will construct an interface FileReader with algorithms for reading, processing, and writing data to a file. This interface will be implemented by file readers of various types.



3. **Iterator Pattern:** We will use the iterator design to address the need to efficiently traverse across different types of files. We will have a FileIterator interface that will be implemented by a class with many overloaded constructors (one for each file type).



Describe the consequences of using this/these pattern(s).

1. Factory Pattern:
    a. The Factory Method separated credit card construction code from the code that actually uses the card. Therefore, it's easier to extend the card construction code independently from the rest of the code.
    b. You can introduce new variants of cards without breaking existing client code.
2. Strategy Pattern:
    a. We can now swap algorithms used at runtime.
    b. Isolated the implementation details of an file reading algorithm from the code that uses it.
    c. We can introduce new strategies without having to change the FileProcessor.
3. Iterator Pattern:
    a. The iterator encapsulates the details of working with a file with complex structure.
    b. We can iterate new types of files and pass the iterator to existing code without breaking anything.