Part 1

Describe what is the primary problem you try to solve

The primary problem that needs to be solved is comprises of the below steps

1. Identify the type of file
2. Create the object of that particular class as per the file type
3. Follow the strategy pattern to use the object of a specific type of file and perform the read and write operations
4. The read operations involves extracting parameters from the corresponding input file
5. The fileWrite function writes the data into the file.

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

The secondary problem that is being attempted to be solved here is type of card detection.

1. Read the input file
2. Send the extracted parameters from the input file to a CardDetectorFactory class to identify the type of card and create the corresponding object implementing Factory Method Design Pattern.
3. The card number and corresponding type of card is obtained the object created by the Factory method.
4. The entire CreditCard object is moved into the List, which is further used for writing into the file

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

The design patterns used are as under.

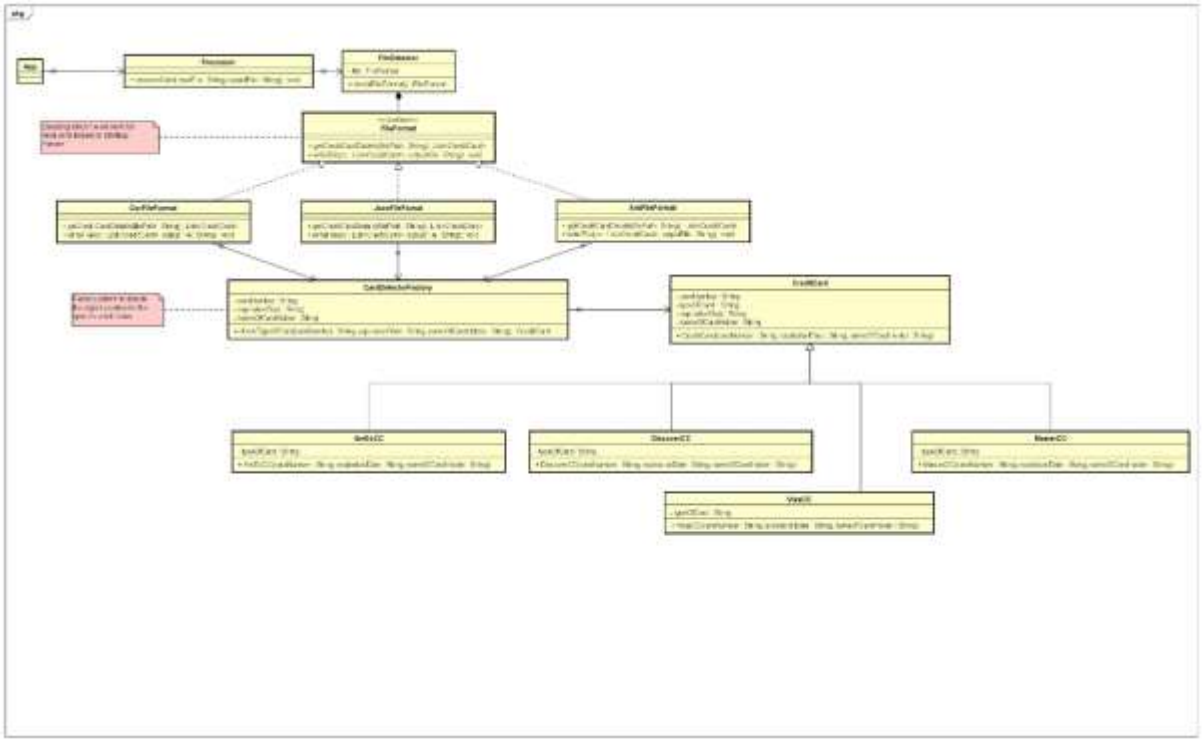
1. Strategy Pattern
   This pattern is utilised when a specific file type is identified to perform read and write operations

   Participants
   - **Strategy**: FileFormat.java interface declaring the functions for read and write operations
   - **Concrete Strategy**: CsvFileFormat.java, JsonFileFormat.java, XmlFileFormat.java implements FileFormat interface and defines the body of the read and write operations
   - **Context**: FileFactory.java is the context which maintains a reference to the strategy object

2. Factory Pattern
   This pattern is used to identify the type of card and create the object corresponding to that class of which the card is detected to belong.

Participants

- **Concrete Factory:** CardDetectorFactory.java is the factory class that identifies the type of card and creates the corresponding object of that specific class**.**
- **Concrete Product:** AmExCC.java, DiscoverCC.java, MasterCC.java, VisaCC.java are the concrete product whose objects are created by the Factory class.
- **Client:** Processor.java that only invokes the read and write operation methods.



Class Diagram showing the design pattern

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

The consequences of using the above showcased design patterns are as follows:

1. The strategy pattern has helped in fixing the operations for any type of file. If the new file type is included, a new Concrete Strategy implementation class would be written. There won't be any change to other concrete.
2. The factory method pattern helps in identifying the type of card, so with the addition of any new type of card, only changed would be made is in the CreditCardDetector.java file with some add-on check conditions and a separate class extending from the parent class CreditCard.java would be written. Thus, existing classes won't be affected.
3. It has helped in maintaining a code and also made it easy to follow the design pattern and understand the flow of the code.