**Title**: Credit Card Processing System

1. **Primary Problem**:

The primary problem addressed in this system is the determination of the credit card type based on the given card number. This involves identifying whether a card number corresponds to a Visa, MasterCard, American Express, or Discover card, based on predefined criteria such as the starting digits and the length of the card number.

2. **Secondary Problems**:

**File Processing**: Reading and writing different file formats (CSV, XML, JSON) to handle credit card data.

**Data Representation**: Creating appropriate data structures to represent and manage credit card information.

3. **Design Patterns Used**:

a. **Factory Pattern (Creational Pattern)**:

**Usage**: CreditCardFactory is used to encapsulate the logic for determining the type of a credit card. It provides a static method getCreditCardType that returns the type based on the card number.

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

**Usage**: FileProcessor utilizes the Strategy pattern through FileFormatParser and FileFormatWriter interfaces, allowing for flexible file processing strategies (CSV, XML, JSON).

4. **Consequences of Using These Patterns**:

**Factory Pattern**:

**Pros**: Encapsulates card type determination logic, making the system easy to modify and extend.

Reduces coupling by not requiring clients to know the specifics of card type determination.

**Cons**: Can lead to complex code structure if many card types with different validation criteria are added.

**Strategy Pattern**:

**Pros**: Provides flexibility to change file processing strategies without modifying the client code.

Promotes the open/closed principle by allowing new file formats to be added without changing existing code.

**Cons**: The strategy interface can become large or complex if there are too many different file formats to support.