**The Credit Card Problem**

Input file : CSV Files

Sample of input files

|  |  |  |
| --- | --- | --- |
| CardNumber | ExpirationDate | NameOfCardholder |
| 5410000000000000 | 3/20/30 | Alice |
| 4120000000000 | 4/20/30 | Bob |
| 341000000000000 | 5/20/30 | Eve |
| 6010000000000000 | 6/20/30 | Richard |

CSV File will have credit card number, exp date and name . And all the records are ‘,’ separated.

Different types of Credit card exists , VIZ:

These credit card have there specific card specifications which needs to be taken when parsing the information

A screenshot of a cell phone

Description automatically generated

Problem :

We need to identify the type of the file that is given as an input as in future there is a chance of new type of file types to come.

And on the run time decide the subclass of credit card.

Secondary Problem :

There can be a functions in future which might have different functions based on what type of credit card subclass what we are dealing with.

Three Design Patterns for the problem that can be used are:

1. Strategy Pattern: to incorporate different pattern to parse the input file
2. Iterator Pattern: To iterate different input file
3. Factory Pattern: to decide and create credit card object on run time depending on the input record.

1.Strategy Pattern:

A screenshot of a cell phone

Description automatically generated

2.

Iteartor Pattern

A screenshot of a cell phone

Description automatically generated

3.Factory Pattern

A screenshot of a cell phone

Description automatically generated

Consequences of using design pattern:

1. Strategy Pattern will help us build a system which decides the type of behavior required on the type of input and react in a different way during the runtime. Basically it will help us understand what type of file is being given as a input, and strategy pattern will decide what type of behavior/function that needs to be used on the input file
2. Iterator pattern: With every type of different file, we need an iterator to iterate the object, and Iterator pattern will make it easy
3. We have a base object of credit card, and on the run time based on different feature of input provided we need to decide what type of credit card object needs to be created. Factory pattern makes it easier to create object on runtime.