**Basic Credit Card Processing**

This is a **Java project developed with the support of Maven** as a build automation tool to install/import any dependencies that are required. It implements Basic Credit Card Processing.

**Overview of Design Process**

* Divided this project into three main Java modules. They are **MainCPP.java**, **Helper.java** and **Record.java**
* **MainCPP.java** acts as a gateway to this project as it has main method from where JVM starts execution. It has sorting and summary printing aspects of the project.
* **Helper.java** contains all necessary methods such as provisioning for two types of input sources like reading the input file, **processing each transaction** like **adding new accounts**, **validating credit card numbers using Luhn algorithm and process charges and credits on those accounts**.
* **Record.java** acts as a temporary persistence module using **ArrayList** data structure. Credit card numbers for whose Luhn algorithm fails will have **erroreous** status and other will have **active** status.

**Reason for choosing Java as the programming language**

* Aspects like **ArrayList manipulations, Sorting, File IO handling mechanisms** made me to choose Java. Although it can be done in Python and other languages the familiarity with these concepts in Java made me to choose it over other languages.

**Required dependencies**

* **Maven** is required to build and run this project. Download maven and add it's path to the environment variable.
* Unit tests are written using **Junit4 and Mockito** frameworks. Therefore, these are the dependencies that are required to run test cases in this project.

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-all</artifactId>

<version>1.10.19</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

**How to build and compile code:**

* Open the terminal/command prompt in a folder where project is located. After that, Run the command mvn clean install -DskipTests=true to clean the target, compile, build and package the project.

You must see **BUILD SUCCESS** as shown in the screenshot below:

Text

Description automatically generated

**How to run test cases:**

To run unit/acceptance tests, run the maven command mvn -q test which runs all unit test cases and print the success output.

* Input files for Unit tests are in **/src/main/resources** directory of the project.
* This project has 8 Unit tests.

You must see the test results as shown in the screenshot below:

Text

Description automatically generated

**How to run code:**

* This program will accept input from two sources:
  1. **Filename passed as a command line argument**
* In the same terminal window, run the maven command mvn -q exec:java -Dexec.mainClass="com.ccp.MainCCP" -Dexec.args="input.txt". **input.txt** file is in the root folder of this project (CCP/input.txt) and you can edit and pass your own values. It prints status of the accounts with names followed by balance in alphabetical order.

You must see the output as shown in the screenshot below for the **input.txt** file

Text

Description automatically generated with medium confidence

* 1. **Input read from STDIN**
* In the same terminal window, run the command mvn -q exec:java -Dexec.mainClass="com.ccp.MainCCP". Then the program prompts you to enter the input file (say **input.txt** and enter). It prints status of the accounts with names followed by balance in alphabetical order.

You must see the output as shown in the screenshot below:

Text

Description automatically generated