# Basic Credit Card Processing

-----

Imagine that you're writing software for a credit card provider. Implement a

program that will add new credit card accounts, process charges and credits

against them, and display summary information.

## Guiding Philosophy

We want to see a well-modeled, working solution that shows that you can write code and read directions.

We are NOT

- going to throw any "gotchas" at you or your submission,

- testing for your ability to flesh out edge cases, or

- trying to trick you.

Keep it simple! Please do NOT implement extra features that we don't ask for.

## Requirements:

- We'll run your submission using a terminal (console) either on a Mac or Linux machine.

Develop on any platform you like, but your submission must work on one or both of those systems.

- Your submission must be source code only. Please do not include compilation artifacts or binary dependencies.

- Your program must accept input from two sources:

- a filename passed as a command line argument, like `./myprogram input.txt`;

- input read from STDIN, like `./myprogram < input.txt`.

- The program should read the whole input, which may contain multiple lines.

- Your program must accept the following three input commands passed with space delimited

arguments:

- "Add" will create a new credit card for a given name, card number, and limit

- Card numbers should be validated using Luhn 10

- New cards start with a $0 balance

- "Charge" will increase the balance of the card associated with the provided

name by the amount specified

- Charges that would raise the balance over the limit are ignored as if they

were declined

- Charges against Luhn 10 invalid cards are ignored

- "Credit" will decrease the balance of the card associated with the provided

name by the amount specified

- Credits that would drop the balance below $0 will create a negative balance

- Credits against Luhn 10 invalid cards are ignored

- When all input has been read and processed, a summary should be generated and

written to STDOUT in the format shown in the example below.

- The summary should include the name of each person followed by a colon and

balance.

- The names should be displayed alphabetically.

- Display "error" instead of the balance if the credit card number does not pass

Luhn 10.

## Input Assumptions:

- All input will be valid. For example, you don't need to check for or gracefully handle:

- Illegal characters

- Malformed commands

- Capitalization changes

- Multiple "Add" commands for the same name

- Different names with the same credit card number

- "Charge" or "Credit" commands without a preceding "Add" command

- All input will be space delimited.

- Credit card numbers may vary in length up to 19 characters.

- Credit card numbers will always be numeric.

- Amounts will always be prefixed with "$" and will be in whole dollars (no

decimals).

## Input and Output:

Given the following input:

```

Add Tom 4111111111111111 $1000

Add Lisa 5454545454545454 $3000

Add Quincy 1234567890123456 $2000

Charge Tom $500

Charge Tom $800

Charge Lisa $7

Credit Lisa $100

Credit Quincy $200

```

Your program must produce the following output:

```

Lisa: $-93

Quincy: error

Tom: $500

```

## Implementing and Packaging Your Solution:

Implement your solution in Clojure or Java, keep in mind

we may ask you to explain or extend your code. Please write automated tests

and include them with your submission.

You may implement your own Luhn 10 check, or you may use an external library. We

do not prefer either approach over the other, but your solution will be assessed

on correctness and testing.

In addition to your source code and tests (remember, no binaries), please also include a README that

explains:

- An overview of your design decisions

- How to install any required dependencies (runtimes, frameworks, etc)

- How to build, package or compile your code if applicable

- How to run your code and testsfs

Thank you!

\*\*Note: this information is confidential. It is prohibited to share, post online

or otherwise publicize without Braintree's prior written consent.\*\*