Afterpay Touch Coding Exercise

Introduction

Welcome to the Afterpay Touch Coding Exercise.

We'd like to get a feel for how you approach problems, how you think, and how you design your code. Please complete the exercise below in Java or Kotlin and send us your working and tested solution.

Please do not share the exercise on social media (GitHub, Twitter etc).

Thank you and have fun!

Sharing your exercise with us

We would like to keep the solutions private. There are two options for you to submit your code:

1. If you are placing your solution on Github please **create a private repository**. The recruiter will give you an email address or Github username for you to share with when you're done.
2. If you decide to send us a copy of your code please compress everything **including the `.git` folder if you decide to use a source control like Github**.

Exercise

Consider the following credit card fraud detection algorithm:

A credit card transaction comprises the following elements.

- hashed credit card number
- timestamp - of format `year-month-dayThour:minute:second`
- amount - of format `dollars.cents`

Transactions are to be received in a file as a comma separated string of elements, one per line, eg:

```
10d7ce2f43e35fa57d1bbf8b1e2, 2014-04-29T13:15:54, 10.00
```

A credit card will be identified as fraudulent if the sum of amounts for a unique hashed credit card number over a 24-hour sliding window period exceeds the price threshold.

Write a command line application which takes a price threshold argument and a filename, eg:

```
your-app 150.00 filename.csv
```

The file passed to your app will contain a sequence of transactions in chronological order.

Your app should print out the hashed credit card numbers that have been identified as fraudulent.

Guidance

- Feel free to create any additional classes you need to support the design of your solution.
- Although not required using a build tool is recommended.
- We expect to see tests that prove your code works.
- We are looking for pragmatic, testable, and maintainable code. If in doubt, refer to the KISS principle.

Motivation

> Any fool can write code that a computer can understand. Good programmers write code that humans can understand.
>
> - Martin Fowler