CEVIANT

Senior Engineer * Code Challenge

1. Transactions

We would like to have a RESTful API for our statistics. The main use case for the API is to calculate real time statistics for the last 60 seconds of transactions. The API needs the following endpoints:

- `POST /transactions` – called every time a transaction is made.
- `GET /statistics` – returns the statistic based on the transactions of the last 60 seconds.
- `DELETE /transactions` – deletes all transactions.

You can complete the challenge using any IDE of your choice. Please use a GitHub private repository and follow the instructions on each section.
Make sure you push your changes to the master branch and test your solution before submitting.

Specs

`POST /transactions`

This endpoint is called to create a new transaction. It MUST execute in constant time and memory (O(1)).

Body:

```
{
   "amount":"12.3343",
   "timestamp":"2018-07-17T09:59:51.312Z"
}
```

Where:

`amount` – transaction amount; a string of arbitrary length that is parsable as a BigDecimal timestamp – transaction time in the ISO 8601 format `YYYY-MM-DDThh:mm:ss.sssZ` in the UTC timezone (this is not the current timestamp)

Returns: Empty body with one of the following:

- 201 – in case of success
- 204 – if the transaction is older than 60 seconds
- 400 – if the JSON is invalid
- 422 – if any of the fields are not parsable or the transaction date is in the future

`GET /statistics`

This endpoint returns the statistics based on the transactions that happened in the last 60 seconds. It MUST execute in constant time and memory (O(1)).

Returns:

```
{
    "sum":"1000.00",
    "avg":"100.53",
    "max":"200000.49",
    "min":"50.23",
    "count":10
}
```

Where:

- sum – a BigDecimal specifying the total sum of transaction value in the last 60 seconds
- avg – a BigDecimal specifying the average amount of transaction value in the last 60 seconds
- max – a BigDecimal specifying single highest transaction value in the last 60 seconds
- min – a BigDecimal specifying single lowest transaction value in the last 60 seconds
- count – a long specifying the total number of transactions that happened in the last 60 seconds

All BigDecimal values always contain exactly two decimal places and use \`HALF_ROUND_UP\` rounding. eg: 10.345 is returned as 10.35, 10.8 is returned as 10.80

`DELETE /transactions`

This endpoint causes all existing transactions to be deleted The endpoint should accept an empty request body and return a 204 status code.

Requirements

These are the additional requirements for the solution:

You are free to choose any JVM language to complete the challenge in, but your application has to run in Maven.

- The API has to be thread safe with concurrent requests.
- The `POST /transactions` and `GET /statistics` endpoints MUST execute in constant time and memory i.e. O(1). Scheduled cleanup is not sufficient
- The solution has to work without a database (this also applies to in-memory databases). Unit tests are mandatory.
- In addition to passing the tests, the solution must be at a quality level that you would be comfortable enough to put in production.

2. Algorithm

Given a binary tree, find the sum of all the leaf nodes.

Examples:

Input :

```
        1
      /   \
     2     3
    / \   / \
   4   5 6   7
          \
           8
```

Output :

```
Sum = 4 + 5 + 8 + 7 = 24
```