Bajaj Health Programming Challenge (Qualifier 1)

Problem Statement 1

You need to create a Java application packaged as a JAR file that accepts two parameters: PRN Number, and path to a json file. The second parameter will accept the **absolute** location of a JSON file. The application should traverse the JSON file to any depth and pick the value for the first instance of the key "destination". Once the key is found, stop processing further.

The application will produce one output:

MD5 Hash of the string concatenated value of the following: PRN Number, and the value of the key “destination” in the JSON file and a random string.

Generate an MD5 hash of the concatenated PRN number, the value associated with the key "destination" and a random 8-character alphanumeric string. The output should have both the hashed value and the random string appended with the “;” character in the format <hashed value>;<random string>. A new random salt string should be generated every time the application is invoked.

MD5_HASH(PRN Number + Destination Value + Random String);Random String

Note: Keep your PRN number in all small case and without spaces.

Eg. If your PRN is 240350000046, and the file is in directory /path/to/file/test.json, the jar file will be called like:

java -jar test.jar 240350000046 path/to/file/test.json

The output should be the result of:

Md5(24035000040 + <destination value> + <random string>);<random string>

Steps to Implement

1. Parse Command-Line Arguments: Read the PRN Number, and file location.
2. Read and Parse JSON File: Use a JSON parsing library to read the JSON file.
3. Traverse JSON: Implement a function to traverse the JSON and pick the value associated with the first instance of the key "destination".
4. Generate a random alphanumeric string of size 8 characters.
5. Generate the hash of the following string after concatenating the
    a. PRN Number,
    b. the value associated with the key "destination" in the json,
    c. the random string

6. Append the same random string with a “;” after the hash value.
7. The format of the output will be **hash;random string**.
8. Generate a jar file of the application and upload it in github with the source code, and add the links to the submission form (https://forms.office.com/r/W8txqGtNQY)

Example

Given the following JSON file **example.json**:

```
{
 "key1": 10,
 "key2": {
  "key3": "value3",
  "destination": "finalValue"
 },
 "key4": [1, 2, {"destination": "anotherValue"}]
}
```

Running the application with:

```
Java -jar DestinationHashGenerator.jar 12345 example.json
```

Output:

```
c719733517bfbdad3b8504213a7f3c96;5f4dcc3b
```

Explanation:

The first instance of the key "destination" has the value "finalValue" and the PRN number is “12345”, the random string generated is “5f4dcc3b”.

It then generates an MD5 hash of the concatenated string "12345finalValue5f4dcc3b" which is c719733517bfbdad3b8504213a7f3c96

The output is the generated hash and the random string in the format <**hash>;<random string>** which is c719733517bfbdad3b8504213a7f3c96;5f4dcc3b

Note: After completing the code, generate the JAR file for the application and submit your response with appropriate details in the given form https://forms.office.com/r/W8txqGtNQY. Make sure the correct github repository link with the code is shared as well. The jar file should be uploaded in the same repository and the downloadable public raw url to the file should be added to the form response. Example jar url on github: (https://github.com/test/testrepo/raw/main/test.jar)

Problem Statement 2

A developer in your team has developed an API to create an account for a new user. You need to validate this API in as many different angles as possible.

Here are the API details

API endpoint - https://bfhldevapigw.healthrx.co.in/automation-campus/create/user

API curl –

```
curl --location 'https://bfhldevapigw.healthrx.co.in/automation-campus/create/user' \
--header 'roll-number: 1' \
--header 'Content-Type: application/json' \
--data-raw '{
  "firstName": "Test",
  "lastName": "Test",
  "phoneNumber": 9999999999,
  "emailId": "test.test@test.com"
}'
```

Headers -

roll-number (required field)

Body -

firstName (required field)

lastName (required field)

phoneNumber (required field, phoneNumber cannot be duplicate between multiple users)

emailId (required field, emailId cannot be duplicate between multiple users)

You can use any language/ tool to implement test cases. Students implementing most test cases will get selected for the next round.

Please note – **if you don't send your roll number in the header you will not get credits for that test case.**

HINTS:

You can use postman to get started with API testing. Feel free to use online curl testing tools as well.

Couple of example test cases to get you started -

1. Try creating accounts for different users with same phone number, you should get a bad request response code 400
2. If you won't send roll number you will get unauthorized response with error code 401

Similarly try implementing as many test cases as you can.