You can view this report online at : https://www.hackerrank.com/x/tests/1526477/candidates/49967984/report

Full Name:	Abdullah Ismail
Email:	abdullah.ansari4@wipro.com
Test Name:	**Back-End Developer (Spring Boot) Test Abdullah**
Taken On:	7 Feb 2023 11:47:38 IST
Time Taken:	103 min 42 sec/ 105 min
Work Experience:	< 1 years
Contact Number:	+917905111709
Linkedin:	https://www.linkedin.com/in/abdullah-ansari-3557841a4/
Invited by:	Kavitha
Invited on:	6 Feb 2023 21:57:03 IST
Skills Score:	Java (Basic) 50/50
	Problem Solving (Basic) 50/50
	REST API (Intermediate) 0/50
	Spring Boot (Basic) 30/50
Tags Score:	Back-End Development 0/50
	Easy 130/200
	Filtering 30/50
	Interviewer Guidelines 50/50
	JSON 0/50
	Java 80/100
	Loops 50/50
	OOPS 50/50
	Operators 50/50
	Problem Solving 50/100
	REST API 0/50
	Sorting 30/50
	Spring Boot 30/50

65%

130/200

scored in **Back-End Developer (Spring Boot) Test Abdullah** in 103 min 42 sec on 7 Feb 2023 11:47:38 IST

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

Question Description	Time Taken	Score	Status

Q1	Counting Closed Paths › Coding	9 min 34 sec	50/ 50	✓
Q2	Spring Boot: Filter Microservice › Back-end Developer	1 hour 15 min 52 sec	30/ 50	✓
Q3	Nutrition Chain › Coding	14 min 1 sec	50/ 50	!
Q4	REST API: Counting Movies › Coding	2 min 35 sec	0/ 50	✗

QUESTION 1

✓ Correct Answer

Score 50

Counting Closed Paths › Coding

Easy · Operators · Loops · Problem Solving

Interviewer Guidelines

QUESTION DESCRIPTION

Some numbers are formed with closed paths. The digits *0, 4, 6* and *9* each have *1* closed path, and *8* has *2.* None of the other numbers is formed with a closed path. Given a *number*, determine the total number of *closed paths* in all of its digits combined.

Example

number = 649578

The digits with closed paths are *6, 4, 9* and *8.* The total number of closed paths is *1 + 1 + 1 + 2 = 5.*

Function Description

Complete the function *closedPaths* in the editor below.

closedPaths has the following parameter(s):

int number: an integer

Returns:

int: the number of *closed paths* in *number*

Constraints

- $1 \le number \le 10^9$

▼ Input Format For Custom Testing

Input from stdin will be processed as follows and passed to the function:

The only line of input contains a single integer, *number*.

▼ Sample Case 0

Sample Input

```
STDIN      Function
-----      ---------
630     →  number = 630
```

Sample Output

```
2
```

Explanation

Sum the *closed paths* count for each digit, *6, 3* and *0*. Return *1 + 0 + 1 = 2.*

▼ Sample Case 1

Sample Input

```
STDIN     Function
-----     ---------
1288   →  number = 1288
```

Sample Output

```
4
```

Explanation

Sum the *closed paths* count for each digit, *1, 2, 8, 8*. Return *0 + 0 + 2 + 2 = 4*.

INTERVIEWER GUIDELINES

▼ Solution

Skills: Loops, Problem Solving

Optimal Solution:

Iterate over all the digits of the number, adding 1 to the answer if the current digit is one of (0,4,6,9) or 2 if the current digit is 8.

```
def closedPaths(number):
    # Write your code here
    ans=0
    #Iterating over all digits of the number
    for i in map(int,str(number)):
        if(i in (0,4,6,9)):
            ans+=1
        if(i == 8):
            ans+=2
    return ans
```

▼ Complexity Analysis

Time Complexity - O($\log_{10}$(number))

Space Complexity - O(1)

CANDIDATE ANSWER

Language used: **Java 8**

```
class Result {

    /*
     * Complete the 'closedPaths' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER number as parameter.
     */

    public static int closedPaths(int number) {

        int closed_paths = 0;
        while(number > 0){
            int digit = number%10;
            if(digit==4 || digit==6 || digit==9 || digit==0){
                closed_paths+=1;
                number/=10;
            }
            else if(digit==8){
                closed_paths+=2;
                number/=10;
            }
            else {
                number/=10;
                continue;
            }
        }
        return closed_paths;

    }

}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	1	0.0991 sec	23.4 KB
TestCase 1	Easy	Sample case	✓ Success	1	0.0693 sec	23.3 KB
TestCase 2	Easy	Sample case	✓ Success	1	0.1067 sec	23.4 KB
TestCase 3	Easy	Sample case	✓ Success	6	0.1035 sec	23.3 KB
TestCase 4	Easy	Sample case	✓ Success	3	0.1847 sec	23.2 KB
TestCase 5	Easy	Hidden case	✓ Success	3	0.0608 sec	23.3 KB
TestCase 6	Easy	Hidden case	✓ Success	7	0.074 sec	23.5 KB
TestCase 7	Easy	Hidden case	✓ Success	7	0.0593 sec	23.3 KB
TestCase 8	Easy	Hidden case	✓ Success	7	0.0696 sec	23.5 KB
TestCase 9	Easy	Hidden case	✓ Success	7	0.0664 sec	23.6 KB
TestCase 10	Easy	Hidden case	✓ Success	7	0.0711 sec	23.4 KB

No Comments

QUESTION 2

✓ Correct Answer

Score 30

Spring Boot: Filter Microservice › Back-end Developer

Spring Boot | Java | Filtering | Sorting | Easy

QUESTION DESCRIPTION

Implement REST APIs to perform filter and sort operations on a collection of Products.

Each event is a JSON entry with the following keys:

- `barcode` : the unique id of the product (String)
- `price` : the price of the product (Integer)
- `discount` : the discount % available on the product(Integer)
- `available` : the availability status of the product (0 or 1)

Here is an example of a product JSON object:

```
[
    {
      "barcode": "74001755",
      "item": "Ball Gown",
      "category": "Full Body Outfits",
      "price": 3548,
      "discount": 7,
      "available": 1
    },
    {
      "barcode": "74002423",
      "item": "Shawl",
      "category": "Accessories",
      "price": 758,
      "discount": 12,
      "available": 1
    }
]
```

You are provided with the implementation of the models required for all the APIs. The task is to implement a set of REST services that exposes the endpoints and allows for filtering and sorting the collection of product records in the following ways:

`GET` request to `/filter/price/{initial_range}/{final_range}`:

- returns a collection of all products whose price is between the initial and the final range supplied
- The response code is 200, and the response body is an array of products in the price range provided.
- In case there are no such products return status code 400.

`GET` request to `/sort/price`:

- returns a collection of all products sorted by their pricing
- The response code is 200 and the response body is an array of the product names sorted in ascending order of price.

Complete the given project so that it passes all the test cases when running the provided unit tests.

▼ Example requests and responses

`GET` **request to** `/filter/price/{initial_range}/{final_range}`

The response code is 200, and when converted to JSON, the response body is as follows for filter/750/900:

```
[
  {
    "barCode": "74002423"
  }
]
```

`GET` **request to** `/sort/price`

The response code is 200 and the response body, when converted to JSON, is as follows:

```
[
  {
    "barCode": "74002423"
  },
  {
    "barCode": "74001755"
  }
]
```

INTERVIEWER GUIDELINES

controller/SampleController.java

```
package com.hackerrank.sample.controller;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import com.hackerrank.sample.dto.FilteredProducts;
import com.hackerrank.sample.dto.SortedProducts;

@RestController
public class SampleController {


           final String uri =
"https://jsonmock.hackerrank.com/api/inventory";
           RestTemplate restTemplate = new RestTemplate();
           String result = restTemplate.getForObject(uri, String.class);
           JSONObject root = new JSONObject(result);

           JSONArray data = root.getJSONArray("data");



                @CrossOrigin

@GetMapping("/filter/price/{initial_price}/{final_price}")
                private ResponseEntity< ArrayList<FilteredProducts> >
filtered_books(@PathVariable("initial_price") int init_price ,
@PathVariable("final_price") int final_price)
                {

                        try {

                                        ArrayList<FilteredProducts> books
= new ArrayList<FilteredProducts>();

                                List<JSONObject> list = new ArrayList<>
();
                                for (int i = 0; i < data.length(); i++) {
                                        if
(data.getJSONObject(i).getInt("price") >= init_price &&
data.getJSONObject(i).getInt("price") <= final_price) {
                                                FilteredProducts
filteredProduct = new
FilteredProducts(data.getJSONObject(i).getString("barcode"));

books.add(filteredProduct);
                                        }
                                }

                                if (books.isEmpty()){
                                        throw new Exception();
                                }
                                    return new
ResponseEntity<ArrayList<FilteredProducts>>(books, HttpStatus.OK);



                        }catch(Exception E)
                                {
                System.out.println("Error encountered :
"+E.getMessage());
            return new ResponseEntity<ArrayList<FilteredProducts>>
(HttpStatus.NOT_FOUND);
                                }

                }


                @CrossOrigin
                @GetMapping("/sort/price")
                private ResponseEntity<SortedProducts[]> sorted_books()
                {
                        try {
                                 List<JSONObject> list = new ArrayList<>
();
                                 for (int i = 0; i < data.length(); i++){
                                         list.add(data.getJSONObject(i));
                                 }

                                 list.sort((s1, s2) -> {
                                         try {
                                                 return
Integer.compare(s1.getInt("price"), s2.getInt("price"));
                                         } catch (JSONException e) {
                                                 e.printStackTrace();
                                         }
                                         return 0;
                                 });

                                SortedProducts[] ans=new
SortedProducts[data.length()];

                                for (int i = 0; i < list.size(); i++){
                                        ans[i] = new
SortedProducts(list.get(i).getString("barcode"));
                                }



                            return new ResponseEntity<SortedProducts[]>
(ans, HttpStatus.OK);

                        }catch(Exception E)
                                {
                System.out.println("Error encountered :
"+E.getMessage());
            return new ResponseEntity<SortedProducts[]>
(HttpStatus.NOT_FOUND);
                                }

                }



}
```

CANDIDATE SUBMISSION

TESTCASE	TEST FILE	STATUS	SCORE
FilterPrice1	TEST-com.hackerrank.sample.SampleApplicationTests.xml	✓ Success	10.0 / 10.0
FilterPriceCheck2	TEST-com.hackerrank.sample.SampleApplicationTests.xml	⊗ Failed	0 / 10.0
FilterPriceCheck3	TEST-com.hackerrank.sample.SampleApplicationTests.xml	✓ Success	10.0 / 10.0
FilterPriceCheck4	TEST-com.hackerrank.sample.SampleApplicationTests.xml	⊗ Failed	0 / 10.0
SortCheck	TEST-	⊗ Failed	0 / 10.0

com.hackerrank.sample.SampleA
pplicationTests.xml

View candidate code

Review logs: output log

No Comments

QUESTION 3

Needs Review

Score 50

Nutrition Chain › Coding Java OOPS Easy

QUESTION DESCRIPTION

Nutrition in food can be broken down into proteins, fats, and carbohydrates. Implement the following classes about food and nutrition to complete this challenge:

1. abstract class *Food* having the following properties
   - double *proteins*
   - double *fats*
   - double *carbs*
   - double *tastyScore*
   - void *getMacroNutrients* [Abstract Method]
2. class *Egg* which extends class Food and has the following properties:
   - Constructor to initialize the attributes (proteins, fats, and carbs) in the same order.
   - int *tastyScore* = 7
   - String type = "*non-vegetarian*"
   - void *getMacroNutrients* => prints("An egg has [this.proteins] gms of protein, [this.fats] gms of fats and [this.carbs] gms of carbohydrates.")
3. class *Bread* which extends class Food and has the following properties:
   - Constructor to initialize the attributes (proteins, fats, and carbs) in the same order.
   - int *tastyScore* = 8
   - String type = "*vegetarian*"
   - void *getMacroNutrients* => prints(" A slice of bread has [this.proteins] gms of protein, [this.fats] gms of fats and [this.carbs] gms of carbohydrates.")

Note: You do not have to worry about input handling, code stub does that

▼ Input Format For Custom Testing

The first line contains an integer, *n*, denoting the number of food items

Every food item take input in the next 4 lines where, the first line is the name of the food and the next three lines are method calls *(getType, getTaste and getMacros)* in random order.

▼ Sample Case 0

Sample Input

```
1
Bread
getType
getMacros
getTaste
```

Sample Output

```
Bread is vegetarian
A slice of bread has 4.0 gms of protein, 1.1 gms of fats and 13.8 gms of
carbohydrates.
Taste: 8
```

▼ Sample Case 1

Sample Input

```
1
Egg
getMacros
getTaste
getType
```

Sample Output

```
An egg has 6.3 gms of protein, 5.3 gms of fats and 0.6 gms of
carbohydrates.
Taste: 7
Egg is non-vegetarian
```

CANDIDATE ANSWER

Language used: **Java 8**

```java
abstract class Food{
    double proteins;
    double fats;
    double carbs;
    double tastyScore;
    abstract void getMacroNutrients();
}

class Egg extends Food{
    Egg(double proteins, double fats, double carbs){
        this.proteins = proteins;
        this.fats = fats;
        this.carbs = carbs;
    }
    int tastyScore = 7;
    String type = "non-vegetarian";
    void getMacroNutrients(){
        System.out.println("An egg has "+this.proteins+" gms of protein, "
        +this.fats+" gms of fats and "+this.carbs+" gms of carbohydrates.");
    }
}

class Bread extends Food{
    Bread(double proteins, double fats, double carbs){
        this.proteins = proteins;
        this.fats = fats;
        this.carbs = carbs;
    }
    int tastyScore = 8;
    String type = "vegetarian";
    void getMacroNutrients(){
        System.out.println("A slice of bread has "+this.proteins+" gms of
protein, "
        +this.fats+" gms of fats and "+this.carbs+" gms of carbohydrates.");
    }
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.1123 sec	24.9 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.0918 sec	24.8 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.0995 sec	24.9 KB

Testcase 3	Easy	Hidden case	✓ Success	10	0.127 sec	24.9 KB
Testcase 4	Easy	Hidden case	✓ Success	14	0.1402 sec	24.8 KB
Testcase 5	Easy	Hidden case	✓ Success	14	0.1948 sec	24.8 KB

No Comments

QUESTION 4

Wrong Answer

Score 0

REST API: Counting Movies › Coding

REST API | Back-End Development | Easy | JSON | Problem Solving

QUESTION DESCRIPTION

Write an *HTTP GET* method to retrieve information from a movie database concerning how many movies have a particular string in their title. Given a search term, query *https://jsonmock.hackerrank.com/api/moviesdata/search/?Title=[substr].* The query response will be a JSON object with the following five fields:

- *page*: The current page.
- *per_page*: The maximum number of results per page.
- *total*: The total number of movies having the substring *substr* in their title.
- *total_pages*: The total number of pages which must be queried to get all the results.
- *data*: An array of JSON objects containing movie information where the *Title* field denotes the title of the movie.

The function will return the integer value found in the *total* field in the returned JSON object.

Function Description

Complete the function *getNumberOfMovies* in the editor below.

getNumberOfMovies has the following parameter(s):

str substr: the string to search for in the movie database

Returns

int: the value of the total field in the returned JSON object

Constraints

- $0 < |substr| < 20$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The only line contains the string *substr*.

▼ Sample Case 0

Sample Input 0

```
STDIN      Function
-----      --------
maze   →   substr = 'maze'
```

Sample Output 0

```
37
```

Explanation 0

The value of *substr* is *maze*, so our query is *https://jsonmock.hackerrank.com/api/moviesdata/search/?Title=maze* and the response is:

```
{
  "page": 1,
  "per_page": 10,
  "total": 37,
  "total_pages": 4,
  "data": [
    {
      "Title": "The Maze Runner",
      "Year": 2014,
      "imdbID": "tt1790864"
    },
    {
      "Title": "Maze Runner: The Scorch Trials",
      "Year": 2015,
      "imdbID": "tt4046784"
    },
    {
      "Title": "Into the Grizzly Maze",
      "Year": 2015,
      "imdbID": "tt1694021"
    },
    {
      "Title": "Hercules in the Maze of the Minotaur",
      "Year": 1994,
      "imdbID": "tt0110018"
    },
    {
      "Title": "The Crystal Maze",
      "Year": 1990,
      "imdbID": "tt0098774"
    },
    {
      "Title": "The Maze",
      "Year": 2010,
      "imdbID": "tt1675758"
    },
    {
      "Title": "Maze",
      "Year": 2000,
      "imdbID": "tt0246072"
    },
    {
      "Title": "Iron Maze",
      "Year": 1991,
      "imdbID": "tt0102128"
    },
    {
      "Title": "The Maze",
      "Year": 1953,
      "imdbID": "tt0046057"
    },
    {
      "Title": "Maze Runner: The Burn Trials",
      "Year": 2015,
      "imdbID": "tt4844320"
    }
  ]
}
```

Return the value of the *total* field, 37, as the answer.

CANDIDATE ANSWER

The candidate did not manually submit any code. The last compiled version has been auto-submitted and the score you see below is for the auto-submitted version.

Language used: **Java 8**

```
public class Solution {
    /*
     * Complete the function below.
     */
    static int getNumberOfMovies(String substr) {
        /*
         * Endpoint: "https://jsonmock.hackerrank.com/api/moviesdata/search/?
Title=substr"
         */

    }
```

Result: Compilation Failed

Compile Message

```
                  Solution.java:18: error: missing return statement
    }
    ^
1 error
```

No Comments

PDF generated at: 7 Feb 2023 18:31:04 UTC