BARLA ANILKUNAR

Java Full Stack _Coding Assessment_15's report

Submitted on Dec 24 2023 13:11:51 IST

141.7 (94.5%)
scored out of 150

Completed
in the assignment

3
problems attempted out of 3

2.9 / 5
avg. code quality score

Severe Violation
flagged by DoSelect proctoring engine

Test time analysis

53m 45s
time taken for completion

Dec 15 2023 19:54:10 IST
test invite time

Dec 24 2023 12:18:06 IST
test start time

Dec 24 2023 13:11:51 IST
test end time

Performance summary

2
solutions accepted

1
solution partially accepted

Proctor analysis

0
browser used

0
navigation violation

2
webcam violations

0 min
no test window violation

Webcam Violation - flagged by DoSelect Proctoring Engine due to below reasons

Total Frames Captured : 0

Frames with Matching Faces	0

Frames with Multiple Faces	0
Frames with Different Face	0
Frames with No Face	0

Total Frames Missing : 3225

Webcam not detected	0
Test-taker closing the tab	0
Network Issues	0
Other factors*	3225

Total Webcam Violations : 2

Set of 10 back-to-back Suspicious Frames**	0
Set of 10 back-to-back Missing frames	1
Suspicious Frames**/Missing Frames detected in more than 10% of test duration	1

* Missing frames due to other factors such as test-taker's system issues etc

** Suspicious frames includes Multiple Faces, Different Faces and No Face

Identity Image

Solutions

Problem Name	Problem Type	Status	Score
Average of weekly expense	Database	ACCEPTED	**50.0** / 50
Paper Wasp	Coding	PARTIALLY ACCEPTED	**41.7** / 50
Exception in Age	Coding	ACCEPTED	**50.0** / 50

Technology used

Java

MySQL

Additional Information

Question	Response
Enrollment Number	EBEON0923842377
Batch Code (Eg : 2022-XXXX)	2023-10433

Detailed Report

Problem 1 : Average of weekly expense

DATABASE | SCORE: **50**

Environment Specifications & Instructions

- **Type of Database: MySQL**
- **Database Name to be used: DB_**WeekExpense

Existing Information

- **Table Descriptions :** tbl_WeekExpense description as below:

Column Name	Data Type
WeekNumber	varchar(20)
WeekDayName	varchar(50)
Expense	decimal(18, 0)

Problem Statement

Construct a query to display the average of weekly expense "**Avg_Expense**" of week number 5 "**Week05**".

Sample Input

WeekNumber	WeekDayName	Expense
Week05	Monday	20.00
Week05	Tuesday	60.00
Week05	Wednesday	20.00
Week05	Thurusday	42.00
Week05	Friday	10.00
Week05	Saturday	15.00
Week05	Sunday	8.00
Week04	Monday	29.00
Week04	Tuesday	17.00
Week04	Wednesday	42.00
Week04	Thurusday	11.00
Week04	Friday	43.00
Week04	Saturday	10.00
Week04	Sunday	15.00
Week03	Monday	10.00
Week03	Tuesday	32.00
Week03	Wednesday	35.00
Week03	Thurusday	19.00

Sample Output

Avg_Expense
25.00

Note:

The sample output given is not the expected output. It is just given for ease of understanding and clarification.

Solution

ACCEPTED SCORE: **50.0** / 50

MySQL

```mysql
use DB_WeekExpense;
/*
 * Enter your query below.
 * Please append a semicolon ";" at the end of the query
 */
 Select avg(Expense)as Avg_Expense from tbl_WeekExpense where WeekNumber='week05'
```

Evaluation Details

Testcase #1 *(weight:1)*

Status	Passed
Execution time	0.00s
CPU	0s
Memory	6MB
Description	Testcase passed! The solution's output matches the expected output.

Problem 2 : Paper Wasp

CODING | SCORE: **50**

Your task here is to implement a **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields, and methods unless mentioned otherwise.

Specifications:

```
class definitions:
 class Insect:
  data members:
    String insectName;
    int insectWeight;
     visibility: private

   Insect(String insectName, int insectWeight): constructor with public
visibility
    Define getters and setters with public visibility
    toString(): has been implemented for you

 class Insecticides:
  method definition:
   mapInsectstName(List<Insect> list):
    return type: List<String>
    visibility: public

   getWeight(List<Insect> list):
    return type: List<Insect>
    visibility: public
```

Task:

class Insect:

- define class **Insect** according to the above specifications

class **Insecticides:**

Implement the below method for this class:

- **List<String>mapInsectsName(List<Insect> list):** fetch and return the Insect name from the list
- **List<Insect>getWeight(List<Insect> list):** filter the weight from the list greater than **40** and less than equal to **100**, put it into a list and return the desired list

> *Refer sample output for clarity*

Sample Input

```
 Insecticides i = new Insecticides();
  List<Insect> list = new ArrayList<Insect>();
         list.add(new Insect("Ant", 45));
         list.add(new Insect("Cockroach", 65));
         list.add(new Insect("bee", 99));
         list.add(new Insect("paper wasp", 11));
-----------------------------------------------------------
```

```
  i.mapInsectstName(list)
  i.getWeight(list)
```

Sample Output

```
[Ant, Cockroach, bee, paper wasp]
------------------------------------------------------------
[Insect{insectName='Ant', insectWeight=45}, Insect{insectName='Cockroach',
insectWeight=65}, Insect{insectName='bee', insectWeight=99}]
```

NOTE

- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

Solution

PARTIALLY ACCEPTED | SCORE: **41.7** / 50

Code Quality Analysis

Many quality violations

Quality score: 1.8

Deep Code Analysis Results

Straightforward approach

No cyclomatic constructs detected.

Modular code

Sufficient reusable components found.

Extensible implementation

The code is easy to extend.

Java 8

```
1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6
7  class Insect {
8    //Code Here..
9    private String insectName;
10   private int insectWeight;
11
12   public Insect (String insectName,int insectWeight)
13   {
14       this.insectWeight=insectWeight;
15       this.insectName=insectName;
16   }
17   public String getInsectName()
18 {
19     return insectName;
```

```
}
public int getInsectWeight()
{
    return insectWeight;
}
      @Override
    public String toString() {
        return "Insect{" +
                "insectName='" + insectName + '\'' +
                ", insectWeight=" + insectWeight +
                '}';
    }
}

class Insecticides {
 //Code Here..
 public List<String>
 mapInsectstName(List<Insect>list){
     List<String>names=new ArrayList<>();
     for(Insect insect:list){
         names.add(insect.getInsectName());
     }
     return names;
 }

 public List<Insect>
 getWeight(List<Insect> list){

  List<Insect> filteredList=new ArrayList<>();
 for(Insect insect:list){
         if(insect.getInsectWeight()>40 && insect.getInsectWeight()<=100){
             filteredList.add(insect);
         }
     }
     return filteredList;
 }
 }

public class Source {
        public static void main(String args[] ) throws Exception {
                /* Enter your code here. Read input from STDIN. Print output to STDOUT */
                Insecticides i=new Insecticides();
                List<Insect> list =new ArrayList<>();
                list.add(new Insect("Ant",45));
                list.add(new Insect("Cockroach",65));
                list.add(new Insect("bee",99));
                list.add(new Insect("paper wasp",11));
                System.out.println(i.mapInsectstName(list));

                System.out.println(i.getWeight(list));

        }
}
```

Evaluation Details

Test_Insecticides *(weight:1)*

Status	Passed
Execution time	3.36s
CPU	0s
Memory	1MB
Description	Testcase passed!

Test_getWeight2 *(weight:1)*

Status	Passed
Execution time	3.13s
CPU	0s
Memory	1MB
Description	Testcase passed!

Test_Insect *(weight:1)*

Status	Failed
Execution time	3.24s
CPU	0s
Memory	1MB
Description	Testcase failed.

Evaluation logs

```
Exception in thread "main" java.lang.AssertionError: expected:<11> but was:<13>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:645)
at org.junit.Assert.assertEquals(Assert.java:631)
at eval.main(eval.java:13)
```

Test_getWeight1 *(weight:1)*

Status	Passed
Execution time	3.03s
CPU	0s
Memory	1MB

Description	Testcase passed!

Test_mapInsectstName2 *(weight:1)*

Status	Passed
Execution time	2.98s
CPU	0s
Memory	1MB
Description	Testcase passed!

Test_mapInsectstName1 *(weight:1)*

Status	Passed
Execution time	2.83s
CPU	0s
Memory	1MB
Description	Testcase passed!

Sample_TC *(sample)*

Status	Passed
Execution time	3.60s
CPU	0s
Memory	1MB
Description	Testcase passed!

Problem 3 : Exception in Age

CODING | SCORE: **50**

Write a java program to validate the age of a person and display proper message by using user defined exception. Age of a person should be above 15.

Your task here is to implement a **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications

```
class definitions:
  class MyException: Define exception
  class Source:
    method definitons:
      checkAge(int age): throw a user defined exception if age is smaller than
15
        visibility: public
```

Task

- Define **MyException** class
- Create a class **Source** and implement the below given method
- **String checkAge(int age)**: throw a user defined exception if age is smaller than 15

Sample Input

```
22
```

Sample Output

```
valid
```

NOTE:

- The above **Sample Input** and **Sample Output** are only for demonstration purposes and will be obtained if you implement the **main()** method with all method calls accordingly.
- Upon implementation of **main()** method, you can use the **RUN CODE** button to pass the **Sample Input** as input data in the method calls and arrive at the **Sample Output**.

Solution

ACCEPTED | SCORE: **50.0** / 50

Code Quality Analysis

Minor quality violations

Quality score: 3.9

Deep Code Analysis Results

Straightforward approach

No cyclomatic constructs detected.

Modular code

Sufficient reusable components found.

Low extensibility

Some extensible features detected.

Java 8

```
class MyException extends Exception{
    MyException(String message){
        super(message);
    }
}
public class Source{
    public String checkAge(int age)throws MyException{
        if(age <=15){
            throw new MyException("Age should be above 15");
        }
        else{
            return "valid";
        }
    }
    public static void main(String[] args){

        try{
            Source source =new Source();
             int age = 22;
            // checkAge(age);
            System.out.println(source.checkAge(age));
        }
        catch(MyException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Evaluation Details

Test_Methods_Source *(weight:1)*

Status	Passed
Execution time	3.51s
CPU	0s
Memory	1MB
Description	Testcase passed!

Sample_TC *(sample)*

Status	Passed
Execution time	3.67s

CPU	0s
Memory	1MB
Description	Testcase passed!

Test_Valid *(weight:1)*

Status	Passed
Execution time	3.45s
CPU	0s
Memory	1MB
Description	Testcase passed!

Test_Methods_MyException *(weight:1)*

Status	Passed
Execution time	3.86s
CPU	0s
Memory	1MB
Description	Testcase passed!