PITTALA ARUN KUMAR

Java Full Stack - Coding Assessment 38's report

Submitted on Jun 24 2023 22:47:33 IST

116.7 (39%)
scored out of 300

Completed
in the assignment

3
problems attempted out of 3

4.1 / 5
avg. code quality score

Severe Violation
flagged by DoSelect proctoring engine

Test time analysis

2h 0m 0s
time taken for completion

Jun 24 2023 20:46:33 IST
test invite time

Jun 24 2023 20:47:33 IST
test start time

Jun 24 2023 22:47:33 IST
test end time

Performance summary

2
solutions partially accepted

1
solution rejected

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 : 5293.0

Webcam not detected	0
Test-taker closing the tab	0
Other factors*	5293.0

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 network, test-taker's system issues etc

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

Solutions

Problem Name	Problem Type	Status	Score
Shop Online	Coding	REJECTED	**0.0** / 100
Create Table	Database	PARTIALLY ACCEPTED	**66.7** / 100
Validations [Lab 10 Ex-3]	Coding	PARTIALLY ACCEPTED	**50.0** / 100

Technology used

Java

MySQL

Additional Information

Question	Response
Enrollment Number	EBEON0223750433
Batch Code (Eg : 2022-XXXX)	2022-8936

Detailed Report

Problem 1 : Shop Online

CODING SCORE: **100**

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 class unless mentioned otherwise.

Specifications:

```
class definitions:
 class Customer:
       data fields:
          int  id
          String name
          double walletBalance
          String address
    method definitions:
        Define a parameterized constructor with public visibility

 class Item:
    data fields:
       int id
       String name
       String companyName
       double price
       boolean isInStock
  method definitions:
    Define a parameterized constructor with public visibility

  class ShoppingWebsite:
   method definition:
       purchaseItem(Item i, Customer c) throws ItemOutOfStockException,
InsufficientBalanceException:
          return type: String
          visibility: public

 class InsufficientBalanceException extends Exception:
   method definition:
      InsufficientBalanceException(String message):
         visibility: public

 class ItemOutOfStockException extends Exception:
  method definition:
     ItemOutOfStockException(String message):
        visibility: public
```

Task:

-Implement class **Customer** according to the above specifications

-Implement class **Item** according to the above specifications

-Class **ShoppingWebsite**

String purchaseItem(Item i, Customer c) throws ItemOutOfStockException, InsufficientBalanceException:

- Throw an **ItemOutOfStockException** when the item is out of stock with the message "item is out of stock".

- Throw an **InsufficientBalanceException** when customer wallet balance is not sufficient(Item price is greater than the wallet balance) with the message "customer wallet balance is not sufficient"**.**
- If no exception found then return "Order Successful".

-class **InsufficientBalanceException**

- define custom exception class **InsufficientBalanceException** by **extending** the **Exception** class.
- define a parameterized constructor with a String argument to pass the message to the super class.

-class **ItemOutOfStockException**

- define custom exception class **ItemOutOfStockException** by **extending** the **Exception** class.
- define a parameterized constructor with a String argument to pass the message to the super class.

Sample Testcase

Input

```
Customer cusDet = new Customer(927392, "Steve" ,5000.0, "USA");
Item itemDet = new Item(27392, "T-Shirt", "US polo", 800, true);
ShoppingWebsite obj = new ShoppingWebsite();
String out = obj.purchaseItem(itemDet, cusDet);
```

output

```
out = "Order Successful"
```

NOTE

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

Solution

REJECTED SCORE: **0.0** / 100

Java 8

```
1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6  class Customer {
7    // Write code from here..
8    int id;
9    String name;
```

```
10    double walletBalance;
11    String address;
12    public Customer(int id,String name,double walletBalance,String address){
13        this.id=id;
14        this.name=name;
15        this.walletBalance=walletBalance;
16        this.address=address;
17    }
18  }
19  class Item {
20    // Write code from here..
21    int id;
22    String name;
23    String company;
24    double price;
25    boolean isInstock;
26    pubic Item(int id,String name,String company,double price,boolean isInstock){
27        this.id=id;
28        this.name=name;
29        this.company=company;
30        this.price=price;
31        this.isInstock=isInstock;
32    }
33  }
34
35  class ShoppingWebsite {
36    // Write code from here
37    public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
38    InsufficientBalanceException{
39        if(!item.isInstock){
40            throw new ItemOutOfStockException("item is out of stock");
41        }
42        if(item.price>customer.walletBalance){
43            throw new InsufficientBalanceException("Customer wallet balance is sufficient");
44        }
45        return "order Successful";
46    }
47  }
48
49
50  class InsufficientBalanceException extends Exception {
51    // Write code from here..
52    public InsufficientBalanceException(String meassage){
53        super(message);
54    }
55  }
56  class ItemOutOfStockException extends Exception{
57    // Write code from here..
58    public ItemOfStockException(String message){
59        super(message);
60    }
61  }
62  public class Source {
63          public static void main(String args[] ) throws Exception {
64                  /* Enter your code here. Read input from STDIN. Print output to STDOUT */
65                  Customer cusDet=new Customer(927392,"Steve",5000);
66          }
67  }
```

Evaluation Details

ValidData_TC *(weight:1)*

Status	Failed
Execution time	1.11s
CPU	0s

Memory	436kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

InvalidData_ItemOutOfStockException *(weight:1)*

Status	Failed
Execution time	1.07s
CPU	0s
Memory	436kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

InvalidData_InsufficientBalanceException *(weight:1)*

Status	Failed
Execution time	1.30s
CPU	0s
Memory	432kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

Sample Testcase *(sample)*

Status	Failed
Execution time	1.15s
CPU	0s
Memory	432kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

ShoppingWebsite_TC *(weight:1)*

Status	Failed
Execution time	1.32s
CPU	0s
Memory	436kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

InsufficientBalanceException_TC *(weight:1)*

Status	Failed
Execution time	1.08s
CPU	0s
Memory	432kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

Customer_TC *(weight:1)*

Status	Failed
Execution time	1.20s
CPU	0s
Memory	436kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

Item_TC *(weight:1)*

Status	Failed
Execution time	1.07s

CPU	0s
Memory	436kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

ItemOutOfStockException_TC *(weight:1)*

Status	Failed
Execution time	1.09s
CPU	0s
Memory	432kB
Description	Testcase failed.

Evaluation logs

```
Source.java:37: error: ';' expected
public String purchaseItem(Item item,Customer customer) throwsOutOfStockException,
^
Source.java:58: error: invalid method declaration; return type required
public ItemOfStockException(String message){
^
2 errors
```

Problem 2 : Create Table

DATABASE | SCORE: **100**

- Create a database named **organization**
- In the database **organization**, create the following table

Table name: **Employee**

Fields:

```
EmployeeId : INT (Primary key)
LastName : NVARCHAR(20) (do not accept NULL value)
FirstName : NVARCHAR(20) (do not accept NULL value)
Title : NVARCHAR(30)
BirthDate : DATETIME
JoiningDate : DATETIME
Address : NVARCHAR(70)
City : NVARCHAR(40)
State : NVARCHAR(40)
Country : NVARCHAR(40)
PostalCode : NVARCHAR(10)
Phone : NVARCHAR(24)
Email : NVARCHAR(60)
```

- Insert a record with the following values

```
EmployeeId : 40211
LastName : Bar
FirstName : Foo
Title : Engineer
BirthDate : 1992-05-10
JoiningDate : 2017-11-21
Address : XYZ
City : Bengaluru
State : Karnataka
Country : India
PostalCode : 560008
Phone : 9999999999
Email : xyz@xyz.com
```

Solution

PARTIALLY ACCEPTED | SCORE: **66.7** / 100

MySQL

```mysql
use DoSelect;
# Write your sql queries here
CREATE DATABASE organization;
USE organization;
 CREATE TABLE Employee{
     EmployeeId INT PRIMARY KEY,
     LastName NVARCHAR(20) NOT NULL,
     FirstName NVARCHAR(20)NOT NULL;
     Title NVARCHAR(30),
     BirthDate DATETIME,
     JoiningDate DATETIME,
```

```
12      Address NVARCHAR(70),
13      City NVARCHAR(40),
14      State NVARCHAR(40),
15      Country NVARCHAR(40),
16      PostalCode NVARCHAR(40),
17      Phone NVARCHAR(24),
18      Email NVARCHAR(60)
19  };
20  --Insert a record into the Employee table
21  Insert INTO
Employee(EmployeeId,LastName,FirstName,Title,Birthdate,JoiningDate,Address,City,State,Country,Posta
Email)
22
VALUES(40211,'Bar','Foo','Engineer','1992-05-10','2017-11-21','XYZ','Bengaluru','karnataka','India'
```

Evaluation Details

Testcase #1 *(weight:1)*

Status	Passed
Execution time	0.00s
CPU	0s
Memory	4MB
Description	Testcase passed!
Annotation	Primary key has been set correctly

Evaluation logs

```
(1049, "Unknown database 'organization'")
```

Testcase #2 *(weight:1)*

Status	Passed
Execution time	0.00s
CPU	0s
Memory	1MB
Description	Testcase passed!
Annotation	Data inserted in the required format in the table

Evaluation logs

```
(1049, "Unknown database 'organization'")
```

Testcase #3 *(weight:1)*

Status	Failed
Execution time	0.00s
CPU	0s

Memory	0kB
Description	Testcase failed.
Annotation	NULL values are allowed in EmployeeId, LastName or FirstName

Evaluation logs

```
ERROR 1044 (42000) at line 3: Access denied for user 'doselect_hacker'@'localhost' to database 'organization'
```

Problem 3 : Validations [Lab 10 Ex-3]

CODING | SCORE: **100**

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 Source:
  visibility: public
  method definition:
   validate(String username, String password): method to authenticate username
and password(Use custom values for username and password for authentication)
    return type: boolean
```

Task

Create a **Source** class and implement below given method:

- **validate(String username, String password):** Use lambda expression to authenticate username and password(Use custom values "ABC" for username and "DEF" as password for authentication). Return true if authentication is successful else return false.

Implement using **Lambda expressions.**

NOTE

- Do not use any **for** loops or other control structures.
- Use the **stream API** methods for your implementations, else the test-cases might fail.

Sample Input

```
Alexa coded123
```

Sample Output

```
false
```

Solution

PARTIALLY ACCEPTED | SCORE: **50.0** / 100

Code Quality Analysis

Maintainable code

Quality score: 4.1

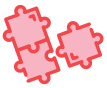
Deep Code Analysis Results

Straightforward approach

No cyclomatic constructs detected.

Very low modularity

No reusable components found.



Very low extensibility

The code is difficult to extend.

Java 8

```java
@FunctionalInterface
interface Validator {
    boolean isValid(String username, String password);
}

public class Source {
    public static void main(String[] args) {
        String username="Alexa";
        String password="coded123";

        Validator validator=(u, p)-> u.equals("ABC") && p.equals("DEF");
        boolean isValid=validator.isValid(username,password);
        System.out.println(isValid);
    }
}
```

Evaluation Details

Test_Method *(weight:1)*

Status	Passed
Execution time	5.19s
CPU	0s
Memory	1MB
Description	Testcase passed!

Test_Validate *(weight:1)*

Status	Failed
Execution time	1.96s
CPU	0s
Memory	436kB
Description	Testcase failed.

Evaluation logs

```
eval.java:8: error: cannot find symbol
assertEquals("true",String.valueOf(s.validate("ABC", "DEF")));
^
symbol: method validate(String,String)
```

```
location: variable s of type Source
eval.java:9: error: cannot find symbol
assertEquals("false",String.valueOf(s.validate("al", "12")));
^
symbol: method validate(String,String)
location: variable s of type Source
eval.java:10: error: cannot find symbol
assertEquals("false",String.valueOf(s.validate("fsd", "345")));
^
symbol: method validate(String,String)
location: variable s of type Source
eval.java:11: error: cannot find symbol
assertEquals("false",String.valueOf(s.validate("tink", "dsfr")));
^
symbol: method validate(String,String)
location: variable s of type Source
eval.java:12: error: cannot find symbol
assertEquals("false",String.valueOf(s.validate("lsrfg", "123dfg")));
^
symbol: method validate(String,String)
location: variable s of type Source
5 errors
```

Sample_TC *(sample)*

Status	Failed
Execution time	1.82s
CPU	0s
Memory	432kB
Description	Testcase failed.

Evaluation logs

```
eval.java:8: error: cannot find symbol
assertEquals("false",String.valueOf(s.validate("Alexa", "coded123")));
^
symbol: method validate(String,String)
location: variable s of type Source
1 error
```