**Group-2**

1. Flight record retrieval

**Grade settings**: Maximum grade: 100  
**Based on**: JAVA CC JDBC - MetaData V1 - ORACLE (w/o Proj Struc)  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 32 s

Retrieve Flights Based on Source and Destination

Zaro Flight System wants to automate the process in their organization.  The flight details are available in the database, the customer should have the facility to view flights which are from a particular source to destination.

You being their software consultant have been approached by them to develop an application which can be used for managing their business.  You need to implement a java program to view all the flight based on source and destination.

**Component Specification:** **Flight (Model Class)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** | **Responsibilities** |
| **Flight** | int flightId  String source  String destination  int noOfSeats  double flightFare | Include getters and setter method for all the attributes.  Include a five argument constructor in the given order – flightId, source, destination, noOfSeats and flightFare. |  |

**Note:**The class and methods should be declared as public and all the attributes should be declared as private.

**Requirement 1:**Retrieve all the flights with the given source and destination

The customer should have the facility to view flights which are from a particular source to destination. Hence the system should fetch all the flight details for the given source and destination from the database. Those flight details should be added to a ArrayList and return the same.

**Component Specification:** **FlightManagementSystem**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type(Class)** | **Attributes** | **Methods** | **Responsibilities** |
| Retrieve all the flights with the given source and destination | FlightManagementSystem |  | public ArrayList<Flight> viewFlightBySourceDestination(String   source,String destination) | This method should accept a Source and a destination as parameter and retrieve all the flights with the given source and destination from  the database. Return these details as ArrayList<Flight>. |

**Note:**The class and methods should be declared as public and all the attributes should be declared as private.

 The **flight** table is already created at the backend. The structure of flight table is:

|  |  |
| --- | --- |
| **Column Name** | **Datatype** |
| flightId | integer |
| source | varchar2(30) |
| destination | varchar2(30) |
| noofseats | integer |
| flightfare | double |

Sample records available in **flight** table are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Flightid** | **Source** | **Destination** | **Noofseats** | **Flightfare** |
| 18221 | Malaysia | Singapore | 50 | 5000 |
| 18222 | Dubai | Kochi | 25 | 50000 |
| 18223 | Malaysia | Singapore | 150 | 6000 |
| 18224 | Malaysia | Singapore | 100 | 7000 |

To connect to the database you are provided with **database.properties** file and **DB.java** file. **(Do not change any values in database.properties file)**

Create a class called **Main** with the main method and get the inputs like **source** and**destination** from the user**.**

**D**isplay the details of flight such as flightId, noofseats and flightfare for all the flights returned as ArrayList<Flight> from the method **viewFlightBySourceDestination** in **FlightManagementSystem** class.

If no flight is available in the list, the output should be “**No flights available for the given source and destination**”.

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.  
  
Ensure to follow object oriented specifications provided in the question description.  
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

**Sample Input / Output 1:**

Enter the source

**Malaysia**

Enter the destination

**Singapore**

Flightid Noofseats Flightfare

18221 50 5000.0

18223 150 6000.0

18224 100 7000.0

**Sample Input / Output 2:**

Enter the source

**Malaysia**

Enter the destination

**Dubai**

No flights available for the given source and destination

### Automatic evaluation[[+]](javascript:void(0);)

#### Flight.java

1

2 *public* *class* Flight {

3

4 *private* *int* flightId;

5 *private* String source;

6 *private* String destination;

7 *private* *int* noOfSeats;

8 *private* *double* flightFare;

9 *public* *int* getFlightId() {

10 *return* flightId;

11 }

12 *public* *void* setFlightId(*int* flightId) {

13 *this*.flightId = flightId;

14 }

15 *public* String getSource() {

16 *return* source;

17 }

18 *public* *void* setSource(String source) {

19 *this*.source = source;

20 }

21 *public* String getDestination() {

22 *return* destination;

23 }

24 *public* *void* setDestination(String destination) {

25 *this*.destination = destination;

26 }

27 *public* *int* getNoOfSeats() {

28 *return* noOfSeats;

29 }

30 *public* *void* setNoOfSeats(*int* noOfSeats) {

31 *this*.noOfSeats = noOfSeats;

32 }

33 *public* *double* getFlightFare() {

34 *return* flightFare;

35 }

36 *public* *void* setFlightFare(*double* flightFare) {

37 *this*.flightFare = flightFare;

38 }

39 *public* Flight(*int* flightId, String source, String destination,

40 *int* noOfSeats, *double* flightFare) {

41 *super*();

42 *this*.flightId = flightId;

43 *this*.source = source;

44 *this*.destination = destination;

45 *this*.noOfSeats = noOfSeats;

46 *this*.flightFare = flightFare;

47 }

48

49

50

51 }

52

#### FlightManagementSystem.java

1 *import* java.util.ArrayList;

2 *import* java.sql.\*;

3

4

5 *public* *class* FlightManagementSystem {

6

7 *public* ArrayList<Flight> viewFlightBySourceDestination(String source, String destination){

8 ArrayList<Flight> flightList = *new* ArrayList<Flight>();

9 *try*{

10 Connection con = DB.getConnection();

11

12 String query="SELECT \* FROM flight WHERE source= '" + source + "' AND destination= '" + destination + "' ";

13

14 Statement st=con.createStatement();

15

16 ResultSet rst= st.executeQuery(query);

17

18 *while*(rst.next()){

19 *int* flightId= rst.getInt(1);

20 String src=rst.getString(2);

21 String dst=rst.getString(3);

22 *int* noofseats=rst.getInt(4);

23 *double* flightfare=rst.getDouble(5);

24

25 flightList.add(*new* Flight(flightId, src, dst, noofseats, flightfare));

26 }

27 }*catch*(ClassNotFoundException | SQLException e){

28 e.printStackTrace();

29 }

30 *return* flightList;

31 }

32

33 }

#### Main.java

1 *import* java.util.Scanner;

2 *import* java.util.ArrayList;

3

4 *public* *class* Main{

5 *public* *static* *void* main(String[] args){

6 Scanner sc=*new* Scanner(System.in);

7 System.out.println("Enter the source");

8 String source=sc.next();

9 System.out.println("Enter the destination");

10 String destination=sc.next();

11

12 FlightManagementSystem fms= *new* FlightManagementSystem();

13 ArrayList<Flight> flightList=fms.viewFlightBySourceDestination(source,destination);

14 *if*(flightList.isEmpty()){

15 System.out.println("No flights available for the given source and destination");

16 *return*;

17 }

18 System.out.println("Flightid Noofseats Flightfare");

19 *for*(Flight flight : flightList){

20 System.out.println(flight.getFlightId()+" "+flight.getNoOfSeats()+" "+flight.getFlightFare());

21 }

22

23 }

24 }

#### DB.java

1 *import* java.io.FileInputStream;

2 *import* java.io.IOException;

3 *import* java.sql.Connection;

4 *import* java.sql.DriverManager;

5 *import* java.sql.SQLException;

6 *import* java.util.Properties;

7

8 *public* *class* DB {

9

10 *private* *static* Connection con = *null*;

11 *private* *static* Properties props = *new* Properties();

12

13

14 //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

15 *public* *static* Connection getConnection() throws ClassNotFoundException, SQLException {

16 *try*{

17

18 FileInputStream fis = *null*;

19 fis = *new* FileInputStream("database.properties");

20 props.load(fis);

21

22 // load the Driver Class

23 Class.forName(props.getProperty("DB\_DRIVER\_CLASS"));

24

25 // create the connection now

26 con = DriverManager.getConnection(props.getProperty("DB\_URL"),props.getProperty("DB\_USERNAME"),props.getProperty("DB\_PASSWORD"));

27 }

28 *catch*(IOException e){

29 e.printStackTrace();

30 }

31 *return* con;

32 }

33 }

34

#### database.properties

1 #IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE

2 #ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY

3 #YOU CAN CHANGE THE VALUE OF THE PROPERTY

4 #LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD IN DB.java using this properties file only.

5 #Do not hard code the values in DB.java.

6

7 DB\_DRIVER\_CLASS=oracle.jdbc.driver.OracleDriver

8 DB\_URL=jdbc:oracle:thin:@127.0.0.1:1521:XE

9 DB\_USERNAME=${sys:db\_username}

10 DB\_PASSWORD=${sys:db\_password}

11

## Grade

Reviewed on Monday, 7 February 2022, 6:33 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
**Assessment Completed Successfully**  
[[+]](javascript:void(0);)**Grading and Feedback**

**=============================================**

## 2. Get Text and Display Welcome Message

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 16 s

Amir owns “Bouncing Babies” an exclusive online store for baby toys.

He desires to display a welcome message whenever a customer visits his online store and makes a purchase.

Help him do this by incorporating the customer name using the Lambda expression.

**Requirement 1:**Display Welcome message

Amir wants to display a welcome message for his customers. The method displayText is used to display the name of the customer who made an online purchase from his store.

**Component Specification: DisplayText Interface – This is a Functional Interface.**

| **Type(Interface)** | **Methods** | **Responsibilities** |
| --- | --- | --- |
| DisplayText | public void displayText(String   text) | The purpose of this method is to display the welcome message by including the text provided as an argument by using Lambda expression. |
| DisplayText | public default String getInput() | This method should get a String (name of the customer) as input from the user and return the same. This method should be a default method. |

**Annotate the interface with the appropriate annotation**

**Component Specification: Main class**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | **Type(Class)** | **Methods** | **Responsibilities** |
| Display welcome message | Main | public static DisplayText welcomeMessage() | This method should return a DisplayText object. To do this, implement the lambda expression to print the text received as a parameter in the displayText method as “Welcome <text>”. |

**In the Main class write the main method and perform the given steps :**

* Invoke the static method welcomeMessage(). It returns a DisplayText object.
* Capture the DisplayText object in a reference variable.
* Using that reference, invoke the default method getInput.
* It will return a String. Capture that String in a variable.
* Using the reference of DisplayText, invoke the displayText method by passing the String as a parameter.
* The output should be as shown in the sample data mentioned below.

**Note :**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object oriented specifications provided in the question.

Ensure to provide the name for classes, interfaces and methods as specified in the question.

Adhere to the code template, if provided.

**Sample Input 1 :**

**Watson**

**Sample Output 1 :**

Welcome Watson

### Automatic evaluation[[+]](javascript:void(0);)

#### DisplayText.java

1 *import* java.util.\*;

2 @FunctionalInterface

3 *public* *interface* DisplayText

4 {

5 *public* *void* displayText(String text);

6 *public* *default* String getInput()

7 {

8 Scanner read = *new* Scanner(System.in);

9 String str = read.next();

10 *return* str;

11 //return null;

12 }

13 }

#### Main.java

1 *public* *class* Main

2 {

3 *public* *static* DisplayText welcomeMessage()

4 {

5

6 DisplayText dis = (str)->{

7

8 System.out.println("Welcome "+str);

9 };

10 *return* dis;

11 }

12 *public* *static* *void* main(String args[])

13 {

14 DisplayText dis=welcomeMessage();

15 String text = dis.getInput();

16 dis.displayText(text);

17

18 }

19 }

## Grade

Reviewed on Wednesday, 1 December 2021, 10:14 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
[[+]](javascript:void(0);)**Grading and Feedback**

**=============================================**

3. Generate Password

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

**Important Instructions:**

**·         Please read the document thoroughly before you code.**

**·         Import the given skeleton code into your Eclipse.(if provided)**

**·        Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.**

**·        You can create any number of private methods inside the given class.**

**·        You can test your code from main() method of the program**

The system administrator of an organization wants to set password for all the computers for security purpose. To generate a strong password, he wants to combine the username of each user of the system with the reverse of their respective usernames. Help them by using Lambda expressions that caters to their requirement.

**Requirement 1:** PasswordInfo

The Administrator wants to generate password for each system by making use of the passwordGeneration method based on the username which is passed as a string.

**Component Specification:** Password Info Interface – This is a Functional Interface.

|  |  |  |
| --- | --- | --- |
| **Type(Interface)** | **Methods** | **Responsibilities** |
| **PasswordInfo** | public String passwordGeneration(String username) | This method is used to generate the password based on the username and hence returns the generated password |

**Component Specification: Computer Class**

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Methods** | **Responsibilities** |
| Computer | public static PasswordInfo passwordPropagation() | This method should return a PasswordInfo object. To do this, implement the lambda expression to get the password. |
| public static void displayUserDetails(String systemNo,String username,PasswordInfo passwordInfoObj) | This method is used to print the Password Info such as the systemNo, password along with the message, “Your password is generated successfully!!!” based on the systemNo, username, passwordInfoObj which is passed as an argument. |

In the Computer class write the main method and perform the given steps:

* Get the systemNo and username from the user.
* Invoke the static method passwordPropagation(). It returns a passwordInfo object with the definition of the passwordGeneration method.
* Capture the PasswordInfo object in a reference variable.
* Invoke the displayUserDetails method by passing systemNo, username and passwordInfoObj as parameters.
* Inside the userDetails method, you should invoke the passwordGeneration method using the passwordInfo object and the output should be displayed as shown in the sample input/output.
* The output should be as shown in the sample data mentioned below.

**Note:**

* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
* Ensure to use the lambda expression.
* Ensure to follow the object oriented specifications provided in the question.
* Ensure to provide the name for classes, interfaces and methods as specified in the question.
* Adhere to the code template, if provided.

**Sample Input 1:**

Enter system no

**Tek/1234**

Enter username

**Manoj Kumar**

**Sample Output 1:**

Password Info

System no: Tek/1234

Password: Manoj KumarramuK jonaM

Your password is generated successfully!!!

=========================================================================

4. Watican Museum Manipulation

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 60 s **Maximum memory used**: 64 MiB **Maximum execution file size**: 320 KiB

**Important Instructions:**

**· Please read the document thoroughly before you code.**

**· Import the given skeleton code into your Eclipse.(if provided)**

**· Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.**

**· You can create any number of private methods inside the given class.**

**· You can test your code from the main() method of the program.**

Watican Museum is one of the famous museums, they have collections of houses paintings, and sculptures from artists. The Museum management stores their visitor's details in a text file. Now, they need an application to analyze and manipulate the visitor details based on the visitor visit date and the visitor address.

**You are provided with a text file – VisitorDetails.txt, which contains all the visitor details like the visitor Id, visitor name, mobile number, date of visiting and address. Your application should satisfy the following requirements.**

1. View visitor details within two given dates.

2. View visitor details which are above a particular mentioned visitor address.

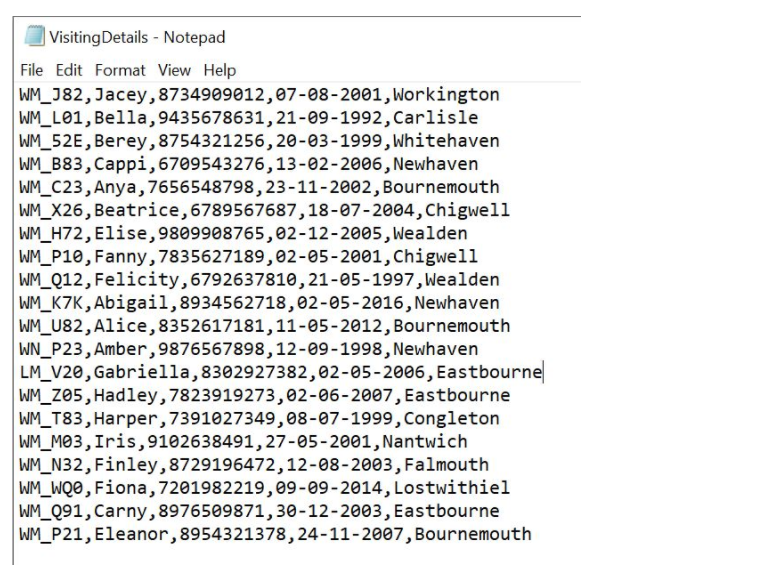
You are provided with a code template which includes the following:

* Visitor class which includes the attributes visitorId, visitorName, mobileNumber, dateOfVisiting and address with all the getters and setters.
* VisitorUtility class which includes the following method declarations.
  + public List<Visitor> generateVisitor(String filePath)
  + public boolean isValidVisitorId(String visitorId)
  + public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate)
  + public Stream<Visitor> viewVisitorDetailsByAddress (Stream<Visitor> visitorStream, double address)
* InvalidVisitorIdException class which inherits the Exception class.
* Main class with a main method which creates the required user interface for the application.
* VisitorDetails.txt which contains all the visitor details like visitor id, visitor name, mobile number, date of visiting and address.

**Note:**

* The Visitor class and the Main class will be provided with all the necessary codes. Please do not edit or delete any line of the code in these two classes.
* Fill your code in the InvalidVisitorIdException class to create a constructor as described in the functional requirements below.
* Fill your code in the respective methods of VisitorUtility class to fulfil all the functional requirements.
* In the VisitorDetails.txt file, each visitor detail has information separated by a comma, and it is given as one customer detail per line.

**Sample data in VisitorDetails.txt file**



**Functional Requirements:**

Fill your code in the respective class and method declarations based on the required functionalities as given below.

|  |  |  |
| --- | --- | --- |
| **Class** | **Attributes/ Methods** | **Rules/ Responsibility** |
| VisitorUtility | public List < Visitor> generateVisitor(String filePath) | Read the text file and convert each line in the text file as String and store it in a List. Each String from the List should be converted into a visitor object and each visitor object should be stored in a List. Return the List of visitors.  **Note:**  Before converting the separated string into a visitor object, the identified visitorId should be validated using the isValidVisitorId method. |
| VisitorUtility | public boolean isValidVisitorId (String visitorId) | Should check whether the provided visitorId is valid or not.  If valid, this method should return true.  If invalid, this method should handle an InvalidVisitorIdException with a message “<visitorId> is Invalid Visitor Id”.  **Validation Rules:**  · Length of the visitorId should be exactly 6.  · The visitorId should start with “WM\_” and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9).  Example.  WM\_A23 |
| InvalidVisitorIdException | Create a constructor with a single String argument and pass it to the parent class constructor. | This class Should inherit the Exception class. The constructor should pass the String message which is thrown to it by calling the parent class constructor. |

**Requirement 1: View visitor details between the dates of visiting**

|  |  |  |
| --- | --- | --- |
| **Class** | **Attributes/ Methods** | **Rules/ Responsibility** |
| VisitorUtility | public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate) | From the provided Stream of Visitor, separate the visitor details which has the date of visiting between fromDate and toDate (both inclusive). Return the separated visitor details as a list. |

**Requirement 2: View visitor** **details which are above a particular mentioned address**

|  |  |  |
| --- | --- | --- |
| **Class** | **Attributes/ Methods** | **Rules/ Responsibility** |
| VisitorUtility | public Stream<Visitor> viewVisitorDetailsByAddress(Stream<Visitor> visitorStream, String address) | From the given Stream of Visitor, separate the visitor details based on address, which has **a particular mentioned**address as provided. Return the separated Stream of visitor. |

**Note:**

1. All inputs/ outputs for processing the functional requirements should be case sensitive.
2. Adhere to the Sample Inputs/ Outputs
3. In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
4. All the Date values used in this application must be in “dd-MM-yyyy” format.
5. Adhere to the code template.
6. Fill all your required codes in the respective blocks. Do not edit or delete the codes provided in the code template.
7. The Sample Inputs/ Outputs given below are generated based on the Sample data given in the VisitorDetails.txt file.
8. Please do not hard code the output.

**Sample Input/ Output 1:**

WM\_52E is Invalid Visitor Id

WM\_K7K is Invalid Visitor Id

WN\_P23 is Invalid Visitor Id

LM\_V20 is Invalid Visitor Id

WM\_WQ0 is Invalid Visitor Id

1. ViewVisitorDetailsByDateOfVisiting

2. ViewVisitorDetailsByAddress

Enter your choice

**1**

Enter the starting date

**19-05-2004**

Enter the ending date

**07-04-2012**

WM\_B83 Cappi 6709543276 13-02-2006 Newhaven

WM\_X26 Beatrice 6789567687 18-07-2004 Chigwell

WM\_H72 Elise 9809908765 02-12-2005 Wealden

WM\_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM\_P21 Eleanor 8954321378 24-11-2007 Bournemouth

**Sample Input/ Output 2:**

WM\_52E is Invalid Visitor Id

WM\_K7K is Invalid Visitor Id

WN\_P23 is Invalid Visitor Id

LM\_V20 is Invalid Visitor Id

WM\_WQ0 is Invalid Visitor Id

1. viewVisitorDetailsByDateOfVisiting

2. viewVisitorDetailsByAddress

Enter your choice

**2**

Enter the address

**Eastbourne**

WM\_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM\_Q91 Carny 8976509871 30-12-2003 Eastbourne

5. Hospital Management\_Streams

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

**Laxmi Hospital** is a world-class health care institution providing patient treatment with specialized medical and nursing staff and medical equipment. It typically provides an emergency department to treat urgent health problems ranging from fire and accident victims to sudden illness. The hospital maintains a register to maintain the records of the patients who enter the emergency department. The receptionist at the helpdesk would like to filter the patients based on a criterion. Develop a java application for the same using Streams.

**Requirements:**

1.     Read the patient records from the file.

2.     Retrieve the patient details for the specified date interval.

3.     Retrieve the  patient details which are from a particular area (address).

**Component Specification: Patient (POJO Class)**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** |
| Patient | String patientId  String patientName  String contactNumber  String dateOfVisit  String patientAddress | Getters and Setters are given in the code skeleton. |

**Component Specification: PatientUtility**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Methods** | **Responsibilities** |
| PatientUtility | public List <Patient> fetchPatient(String filePath) | Read the file using File I/O or Java Streams and return the validated list of patient records. It should filter the valid patient records based on the valid patient Id using the method isValidPatientId ().  **Note:** Make sure that the user-defined exception is handled in this method itself. |
| PatientUtility | public boolean isValidPatientId (String patientId) | **Validation Guidelines for Valid Patient ID:**   * The length of the Patient Id should be exactly 6. * The Patient Id should start with “WM\_” and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9). Example. WM\_A10.   Check whether the patient Id is valid or not. If invalid, this method should handle an InvalidPatientIdException with a message “<patientid> is an Invalid Patient Id”. |
| PatientUtility | public List<Patient> retrievePatientRecords\_ByDateOfVisit(Stream<Patient> patientStream, String fromDate, String toDate) | From the provided stream of patient, separate the patient details which has the date of visit between fromDate and toDate (both inclusive) and return the resultant patient records as a list. |
| PatientUtility | public Stream<Patient> retrievePatientRecords\_ByAddress(Stream<Patient> patientStream, String address) | From the given stream of patient, filter the patient details based on the user input address, and return the separated Stream of patients. |

**Component Specification: InvalidPatientIdException (User defined Exception)**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Methods** | **Responsibilities** |
| InvalidPatientIdException | public InvalidPatientIdException(String message) | This constructor should set the message to the superclass. |

**Note:**The class and methods should be declared as public and all the attributes should be declared as private.

You are provided with a text file –PatientRegister.txt, which contains all the patient details like the patient Id, patient name, contact number, date of visit, and patient address. You can add any number of records in the text file to test your code.

**Note:**

·       In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.

·       Ensure to follow the object-oriented specifications provided in the question description.

·       Ensure to provide the names for classes, attributes, and methods as specified in the question description.

·       Adhere to the code template, if provided.

**Sample Input/Output 1:**

Invalid Patient Id are:

WM\_52E is an Invalid Patient Id

WM\_K7K is an Invalid Patient Id

WN\_P23 is an Invalid Patient Id

LM\_V20 is an Invalid Patient Id

WM\_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**1**

Enter the start date

**02-03-2003**

Enter the end date

**02-12-2005**

WM\_X26 Beatrice 6789567687 18-07-2004 Texas

WM\_H72 Elise 9809908765 02-12-2005 Washington

WM\_N32 Finley 8729196472 12-08-2003 Pennsylvania

WM\_Q91 Carny 8976509871 30-12-2003 Virginia

**Sample Input/Output 2:**

Invalid Patient Id are:

WM\_52E is an Invalid Patient Id

WM\_K7K is an Invalid Patient Id

WN\_P23 is an Invalid Patient Id

LM\_V20 is an Invalid Patient Id

WM\_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**2**

Enter the address

**Carolina**

WM\_C23 Anya 7656548798 23-11-2002 Carolina

WM\_T83 Harper 7391027349 08-07-1999 Carolina

WM\_P21 Eleanor 8954321378 24-11-2007 Carolina

**Sample Input/Output 3:**

Invalid Patient Id are:

WM\_52E is an Invalid Patient Id

WM\_K7K is an Invalid Patient Id

WN\_P23 is an Invalid Patient Id

LM\_V20 is an Invalid Patient Id

WM\_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**1**

Enter the start date

**03-02-2020**

Enter the end date

**02-02-2021**

No patient records available during this interval

**Sample Input/Output 4:**

Invalid Patient Id are:

WM\_52E is an Invalid Patient Id

WM\_K7K is an Invalid Patient Id

WN\_P23 is an Invalid Patient Id

LM\_V20 is an Invalid Patient Id

WM\_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**3**

Invalid Option

### Automatic evaluation[[+]](javascript:void(0);)

#### HospitalManagement/PatientRegister.txt

1 WM\_J82,Jacey,8734909012,07-08-2001,Colorado

2 WM\_L01,Bella,9435678631,21-09-1992,Connecticut

3 WM\_52E,Berey,8754321256,20-03-1999,Indiana

4 WM\_B83,Cappi,6709543276,13-02-2006,Pennsylvania

5 WM\_C23,Anya,7656548798,23-11-2002,Carolina

6 WM\_X26,Beatrice,6789567687,18-07-2004,Texas

7 WM\_H72,Elise,9809908765,02-12-2005,Washington

8 WM\_P10,Fanny,7835627189,02-05-2001,Virginia

9 WM\_Q12,Felicity,6792637810,21-05-1997,Colorado

10 WM\_K7K,Abigail,8934562718,02-05-2016,Indiana

11 WM\_U82,Alice,8352617181,11-05-2012,Indiana

12 WN\_P23,Amber,9876567898,12-09-1998,Pennsylvania

13 LM\_V20,Gabriella,8302927382,02-05-2006,Connecticut

14 WM\_Z05,Hadley,7823919273,02-06-2007,Connecticut

15 WM\_T83,Harper,7391027349,08-07-1999,Carolina

16 WM\_M03,Iris,9102638491,27-05-2001,Texas

17 WM\_N32,Finley,8729196472,12-08-2003,Pennsylvania

18 WM\_WQ0,Fiona,7201982219,09-09-2014,Washington

19 WM\_Q91,Carny,8976509871,30-12-2003,Virginia

20 WM\_P21,Eleanor,8954321378,24-11-2007,Carolina

#### HospitalManagement/src/InvalidPatientIdException.java

1

2 //public class InvalidPatientIdException{

3 //FILL THE CODE HERE

4 *public* *class* InvalidPatientIdException *extends* Exception{

5 *public* InvalidPatientIdException(String message){

6 *super*(message);

7 }

8 }

9

10

11

12

#### HospitalManagement/src/Main.java

1 *public* *class* Main {

2

3 *public* *static* *void* main(String[] args){

4

5 // CODE SKELETON - VALIDATION STARTS

6 // DO NOT CHANGE THIS CODE

7

8 *new* SkeletonValidator();

9 // CODE SKELETON - VALIDATION ENDS

10

11 // FILL THE CODE HERE

12

13 }

14

15 }

16

17

#### HospitalManagement/src/Patient.java

1 //DO NOT ADD/EDIT THE CODE

2 *public* *class* Patient {

3

4 *private* String patientId;

5 *private* String patientName;

6 *private* String contactNumber;

7 *private* String dateOfVisit;

8 *private* String patientAddress;

9

10 //Setters and Getters

11

12 *public* String getPatientId() {

13 *return* patientId;

14 }

15 *public* *void* setPatientId(String patientId) {

16 *this*.patientId = patientId;

17 }

18 *public* String getPatientName() {

19 *return* patientName;

20 }

21 *public* *void* setPatientName(String patientName) {

22 *this*.patientName = patientName;

23 }

24 *public* String getContactNumber() {

25 *return* contactNumber;

26 }

27 *public* *void* setContactNumber(String contactNumber) {

28 *this*.contactNumber = contactNumber;

29 }

30 *public* String getDateOfVisit() {

31 *return* dateOfVisit;

32 }

33 *public* *void* setDateOfVisit(String dateOfVisit) {

34 *this*.dateOfVisit = dateOfVisit;

35 }

36 *public* String getPatientAddress() {

37 *return* patientAddress;

38 }

39 *public* *void* setPatientAddress(String patientAddress) {

40 *this*.patientAddress = patientAddress;

41 }

42

43

44

45

46 }

47

#### HospitalManagement/src/PatientUtility.java

1 *import* java.util.List;

2 *import* java.util.stream.Stream;

3 *import* java.util.ArrayList;

4 *import* java.io.File;

5 *import* java.io.FileNotFoundException;

6 *import* java.util.Scanner;

7 *import* java.util.regex.\*;

8 *import* java.util.stream.Collectors;

9 *import* java.text.ParseException;

10 *import* java.text.SimpleDateFormat;

11 *import* java.util.Date;

12

13

14 *public* *class* PatientUtility {

15

16 *public* List <Patient> fetchPatient(String filePath) {

17

18

19 //FILL THE CODE HERE

20 List <Patient> patients =*new* ArrayList<>();

21 *try*{

22 File *register* =*new* File(filePath);

23 Scanner reader=*new* Scanner(*register*);

24 *while*(reader.hasNextLine()){

25 Patient p = *new* Patient();

26 String[] infos=reader.nextLine().split(",");

27 *try*{

28 *if*(isValidPatientId(infos[0])){

29 p.setPatientId(infos[0]);

30 p.setPatientName(infos[1]);

31 p.setContactNumber(infos[2]);

32 p.setDateOfVisit(infos[3]);

33 p.setPatientAddress(infos[4]);

34 patients.add(p);

35 }

36 }

37 *catch*(InvalidPatientIdException e1){

38 System.out.println(e1.getMessage());

39 }

40 }

41 reader.close();

42 }

43 *catch*(FileNotFoundException e){}

44 *return* patients;

45

46 //return null;

47 }

48

49

50 *public* *boolean* isValidPatientId (String patientId)throws InvalidPatientIdException

51 {

52

53 //FILL THE CODE HERE

54 Pattern p =Pattern.compile("WM\_[A-Z][0-9]{2}$");

55 Matcher m=p.matcher(patientId);

56 *boolean* ne =m.matches();

57 *if*(!ne){

58 *throw* *new* InvalidPatientIdException(patientId+"is an Invalid Patient Id.");

59

60 }

61 //return inValid;

62 *return* ne;

63 }

64

65

66 *public* List<Patient> retrievePatientRecords\_ByDateOfVisit(Stream<Patient> patientStream, String fromDate, String toDate)

67 {

68 //FILL THE CODE HERE

69 SimpleDateFormat simpleDateFormat=*new* SimpleDateFormat("dd-MM-yyyy");

70 *return* patientStream

71 .filter((p)->{

72 *try*{

73 Date start=simpleDateFormat.parse(fromDate);

74 Date end= simpleDateFormat.parse(toDate);

75 Date current =simpleDateFormat.parse(p.getDateOfVisit());

76 *return* start.compareTo(current)\*current.compareTo(end)>=0;

77 }

78 *catch*(ParseException e){}

79 *return* *false*;

80 }).collect(Collectors.toList());

81 // return null;

82 }

83

84

85

86 *public* Stream<Patient> retrievePatientRecords\_ByAddress(Stream<Patient> patientStream, String address)

87 {

88

89 //FILL THE CODE HERE

90 *return* patientStream.filter(p->address.equals(p.getPatientAddress()));

91 //return null;

92

93

94

95 }

96

97 }

98

#### HospitalManagement/src/SkeletonValidator.java

1 *import* java.lang.reflect.Method;

2 *import* java.util.List;

3 *import* java.util.logging.Level;

4 *import* java.util.logging.Logger;

5 *import* java.util.stream.Stream;

6

7 /\*\*

8 \* @author TJ

9 \*

10 \* This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth auto evaluation

11 \*

12 \*/

13 *public* *class* SkeletonValidator {

14

15 *public* SkeletonValidator() {

16

17

18 validateClassName("Patient");

19 validateClassName("PatientUtility");

20 validateClassName("InvalidPatientIdException");

21 validateMethodSignature(

22 "fetchPatient:java.util.List,isValidPatientId:boolean,retrievePatientRecords\_ByDateOfVisit:java.util.List,retrievePatientRecords\_ByAddress:java.util.stream.Stream",

23 "PatientUtility");

24

25 }

26

27 *private* *static* *final* Logger LOG = Logger.getLogger("SkeletonValidator");

28

29 *protected* *final* *boolean* validateClassName(String className) {

30

31 *boolean* iscorrect = *false*;

32 *try* {

33 Class.forName(className);

34 iscorrect = *true*;

35 LOG.info("Class Name " + className + " is correct");

36

37 } *catch* (ClassNotFoundException e) {

38 LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "

39 + "and class name as provided in the skeleton");

40

41 } *catch* (Exception e) {

42 LOG.log(Level.SEVERE,

43 "There is an error in validating the " + "Class Name. Please manually verify that the "

44 + "Class name is same as skeleton before uploading");

45 }

46 *return* iscorrect;

47

48 }

49

50 *protected* *final* *void* validateMethodSignature(String methodWithExcptn, String className) {

51 Class cls = *null*;

52 *try* {

53

54 String[] actualmethods = methodWithExcptn.split(",");

55 *boolean* errorFlag = *false*;

56 String[] methodSignature;

57 String methodName = *null*;

58 String returnType = *null*;

59

60 *for* (String singleMethod : actualmethods) {

61 *boolean* foundMethod = *false*;

62 methodSignature = singleMethod.split(":");

63

64 methodName = methodSignature[0];

65 returnType = methodSignature[1];

66 cls = Class.forName(className);

67 Method[] methods = cls.getMethods();

68 *for* (Method findMethod : methods) {

69 *if* (methodName.equals(findMethod.getName())) {

70 foundMethod = *true*;

71 *if* (!(findMethod.getReturnType().getName().equals(returnType))) {

72 errorFlag = *true*;

73 LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName

74 + "' method. Please stick to the " + "skeleton provided");

75

76 } *else* {

77 LOG.info("Method signature of " + methodName + " is valid");

78 }

79

80 }

81 }

82 *if* (!foundMethod) {

83 errorFlag = *true*;

84 LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName

85 + ". Do not change the " + "given public method name. " + "Verify it with the skeleton");

86 }

87

88 }

89 *if* (!errorFlag) {

90 LOG.info("Method signature is valid");

91 }

92

93 } *catch* (Exception e) {

94 LOG.log(Level.SEVERE,

95 " There is an error in validating the " + "method structure. Please manually verify that the "

96 + "Method signature is same as the skeleton before uploading");

97 }

98 }

99

100 }

## Grade

Reviewed on Monday, 7 February 2022, 6:04 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
**Assessment Completed Successfully**  
[[+]](javascript:void(0);)**Grading and Feedback**

=========================================================================

## 6. Technology Fest

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

**Institute of Technology**is organizing an All-India Technology Fest for various engineering colleges across the country. The management would like to automate the registration so that it is easier and more systematic while conducting the fest. Create a java application for the same using Threads.

**Component Specification: Participant (POJO Class)**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** |
| Participant | String name  String yearofstudy  String department  String collegeName  String eventName  double registrationFee | Getters, Setters, and  a five-argument constructor in the given order - name, yearofstudy, department, collegeName, eventName are included in the code Skeleton. |

**Requirements:**

·       To calculate the registration fee of the participant based on the event name.

·       To calculate the number of participants registered for a particular event.

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Event Name** | **Registration Fee** |
| 1 | Robocar | 1000 |
| 2 | PaperTalk | 500 |
| 3 | Quiz | 300 |
| 4 | Games | 100 |

**\*Note that Event name is case in- sensitive**

**Component Specification: EventManagement (Thread Class)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type (Class)** | **Attributes** | **Methods** | **Responsibilities** |
| EventManagement | List<Participant> TechList  String searchEvent  int counter |  | Include getters and setter methods for all the attributes. |
| EventManagement |  | public void calculateRegistrationFee(List <Participant> list) | Calculate the registration fee of the participant based on the event name. If the event name doesn’t exist, throw an InvalidEventException with an error message “Event Name is invalid”. |
| EventManagement |  | public void run() | Calculate the number of participants registered for a particular event. Increment the counter attribute based on the search. |

**Note:**The class and methods should be declared as public and all the attributes should be declared as private.

**Component Specification: InvalidEventException**

|  |  |  |
| --- | --- | --- |
| **Type (Class)** | **Methods** | **Responsibilities** |
| InvalidEventException | public InvalidEventException (String message) | To set the message string to the superclass. |

**Create a class called Main with the main method and perform the tasks are given below:**

·       Get the inputs as provided in the sample input.

·       Call the calculateRegistrationFee () method to calculate the registration fee of the participant based on the event name.

·       Print the list of Participant objects with the registration fee.

·       Get the event type to search to find the number of the participants registered for that particular event.

·       Handle the user-defined exception in the main method.

·       Display the output as shown in the sample input/output.

**Note:**

·       In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represent the output.

·       Ensure to follow the object-oriented specifications provided in the question description.

·       Ensure to provide the names for classes, attributes, and methods as specified in the question description.

·       Adhere to the code template, if provided.

**Sample Input/Output 1:**

Enter the number of entries

**3**

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

**rinu/4/EEE/mnm/robocar**

**fina/3/EEE/psg/papertalk**

**rachel/4/civil/kcg/quiz**

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm, EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg, EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg, EventName=quiz, RegistrationFee=300.0

Enter the event to search

**robocar**

Number of participants for ROBOCAR event is 1

**Sample Input/Output 2:**

Enter the number of entries

**3**

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

**rinu/4/EEE/mnm/robocar**

**fina/3/EEE/psg/papertalk**

**rachel/4/civil/kcg/quiz**

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm, EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg, EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg, EventName=quiz, RegistrationFee=300.0

Enter the event to search

**games**

No participant found

**Sample Input/Output 3:**

Enter the number of entries

**2**

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

**vishal/4/mech/vjc/flyingrobo**

**vivek/3/mech/hdl/games**

Event Name is invalid

### Automatic evaluation[[+]](javascript:void(0);)

#### TechnologyFest/src/EventManagement.java

1 *import* java.util.List;

2

3 *public* *class* EventManagement *implements* Runnable {

4 *private* List<Participant> TechList;

5 *private* String searchEvent;

6 *private* *int* counter=0;

7 *public* List<Participant>getTechList()

8 {

9 *return* TechList;

10

11 }

12 *public* *void* setTechList(List<Participant>techList)

13 {

14 TechList=techList;

15 }

16 *public* String getSearchEvent()

17 {

18 *return* searchEvent;

19 }

20 *public* *void* setSearchEvent(String searchEvent)

21 {

22 *this*.searchEvent=searchEvent;

23 }

24 *public* *int* getCounter()

25 {

26 *return* counter;

27 }

28 *public* *void* setCounter(*int* counter)

29 {

30 *this*.counter=counter;

31 }

32 //FILL THE CODE HERE

33

34 *public* *void* calculateRegistrationFee(List<Participant> list) throws InvalidEventException

35

36 {

37 *for*(Participant p:list)

38 {

39 *if*(p.getEventName().equalsIgnoreCase("robocar"))

40 {

41 p.setRegistrationFee(1000);

42 }

43 *else* *if*(p.getEventName().equalsIgnoreCase("papertalk")){

44 p.setRegistrationFee(500);

45

46 }

47

48 *else* *if*(p.getEventName().equalsIgnoreCase("quiz")){

49 p.setRegistrationFee(300);

50 }

51 *else* *if*(p.getEventName().equalsIgnoreCase("games")){

52 p.setRegistrationFee(100);

53 }

54 *else*{

55 *throw* *new* InvalidEventException("Event Name is Invalid");

56 }

57 }

58 //FILL THE CODE HERE

59 setTechList(list);

60 }

61

62 *public* *void* run()

63 {

64 String str="robocarpapertalkquizgames";

65 *if*(str.contains(*this*.getSearchEvent())){

66 *for*(Participant P:*this*.getTechList()){

67 *if*(*this*.getSearchEvent().equals(P.getEventName())){

68 counter++;

69 }

70 }

71 }

72 setCounter(counter);

73

74 //FILL THE CODE HERE

75

76 }

77 }

78

#### TechnologyFest/src/InvalidEventException.java

1 *public* *class* InvalidEventException *extends* Exception{

2 //FILL THE CODE HERE

3 *public* InvalidEventException(String str){

4 *super*(str);

5

6 }

7

8 }

9

#### TechnologyFest/src/Main.java

1

2 *import* java.util.Scanner;

3 *import* java.util.\*;

4 *public* *class* Main {

5 *public* *static* *void* main(String [] args)

6 {

7 // CODE SKELETON - VALIDATION STARTS

8 // DO NOT CHANGE THIS CODE

9

10 *new* SkeletonValidator();

11

12 // CODE SKELETON - VALIDATION ENDS

13

14 Scanner sc=*new* Scanner(System.in);

15 System.out.println("Enter the number of entries");

16 *int* n=sc.nextInt();

17 System.out.println("Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName");

18 List<Participant> list=*new* ArrayList<Participant>();

19 String strlist[]=*new* String[n];

20 *for*(*int* i=0;i<n;i++)

21 {

22 strlist[i]=sc.next();

23 String a[]=strlist[i].split("/");

24 Participant pt=*new* Participant(a[0],a[1],a[2],a[3],a[4]);

25 list.add(pt);

26 }

27 EventManagement em=*new* EventManagement();

28 *try* {

29 em.calculateRegistrationFee(list);

30 }

31 *catch*(InvalidEventException e)

32 {

33 e.printStackTrace();

34

35 }

36 System.out.println("Print participant details");

37 *for*(Participant p:list)

38 {

39 System.out.println(p);

40 }

41 System.out.println("Enter the event to search");

42 String srch=sc.nextLine();

43 em.setSearchEvent(srch);

44 em.run();

45 *int* count=em.getCounter();

46 *if*(count<=0){

47 System.out.println("No participant found");

48

49 }

50 *else*{

51 System.out.println("Number of participants for"+srch+"event is "+count); }

52 }

53 }

54

55

56

57

#### TechnologyFest/src/Participant.java

1 *public* *class* Participant {

2 *private* String name;

3 *private* String yearofstudy;

4 *private* String department;

5 *private* String collegeName;

6 *private* String eventName;

7 *private* *double* registrationFee;

8

9 //5 argument Constructor

10 *public* Participant(String name, String yearofstudy, String department, String collegeName, String eventName) {

11 *super*();

12 *this*.name = name;

13 *this*.yearofstudy = yearofstudy;

14 *this*.department = department;

15 *this*.collegeName = collegeName;

16 *this*.eventName = eventName;

17 }

18

19 *public* String getName() {

20 *return* name;

21 }

22 *public* *void* setName(String name) {

23 *this*.name = name;

24 }

25 *public* String getYearofstudy() {

26 *return* yearofstudy;

27 }

28 *public* *void* setYearofstudy(String yearofstudy) {

29 *this*.yearofstudy = yearofstudy;

30 }

31 *public* String getDepartment() {

32 *return* department;

33 }

34 *public* *void* setDepartment(String department) {

35 *this*.department = department;

36 }

37 *public* String getCollegeName() {

38 *return* collegeName;

39 }

40 *public* *void* setCollegeName(String collegeName) {

41 *this*.collegeName = collegeName;

42 }

43 *public* String getEventName() {

44 *return* eventName;

45 }

46 *public* *void* setEventName(String eventName) {

47 *this*.eventName = eventName;

48 }

49 *public* *double* getRegistrationFee() {

50 *return* registrationFee;

51 }

52 *public* *void* setRegistrationFee(*double* registrationFee) {

53 *this*.registrationFee = registrationFee;

54 }

55

56 @Override

57 *public* String toString() {

58 *return* "Participant [name=" + name + ", yearofstudy=" + yearofstudy + ", department=" + department

59 + ", collegeName=" + collegeName + ", eventName=" + eventName + ", registrationFee=" + registrationFee

60 + "]";

61 }

62

63

64

65

66 }

67

#### TechnologyFest/src/SkeletonValidator.java

1

2 *import* java.lang.reflect.Method;

3 *import* java.util.List;

4 *import* java.util.logging.Level;

5 *import* java.util.logging.Logger;

6 *import* java.util.stream.Stream;

7

8 /\*\*

9 \* @author TJ

10 \*

11 \* This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth auto evaluation

12 \*

13 \*/

14 *public* *class* SkeletonValidator {

15

16 *public* SkeletonValidator() {

17

18 //classes

19 validateClassName("Main");

20 validateClassName("EventManagement");

21 validateClassName("Participant");

22 validateClassName("InvalidEventException");

23 //functional methods

24 validateMethodSignature(

25 "calculateRegistrationFee:void","EventManagement");

26 validateMethodSignature(

27 "run:void","EventManagement");

28

29 //setters and getters of HallHandler

30 validateMethodSignature(

31 "getTechList:List","EventManagement");

32 validateMethodSignature(

33 "setTechList:void","EventManagement");

34

35 validateMethodSignature(

36 "getCounter:int","EventManagement");

37 validateMethodSignature(

38 "setCounter:void","EventManagement");

39

40 validateMethodSignature(

41 "getSearchEvent:String","EventManagement");

42 validateMethodSignature(

43 "setSearchEvent:void","EventManagement");

44

45 //setters and getters of Hall

46 validateMethodSignature(

47 "getName:String","Participant");

48 validateMethodSignature(

49 "setName:void","Participant");

50

51 validateMethodSignature(

52 "getYearofstudy:String","Participant");

53 validateMethodSignature(

54 "setYearofstudy:void","Participant");

55

56 validateMethodSignature(

57 "getDepartment:String","Participant");

58 validateMethodSignature(

59 "setDepartment:void","Participant");

60

61 validateMethodSignature(

62 "getCollegeName:String","Participant");

63 validateMethodSignature(

64 "setCollegeName:void","Participant");

65

66 validateMethodSignature(

67 "getEventName:String","Participant");

68 validateMethodSignature(

69 "setEventName:void","Participant");

70

71 validateMethodSignature(

72 "getRegistrationFee:double","Participant");

73 validateMethodSignature(

74 "setRegistrationFee:void","Participant");

75

76 }

77

78 *private* *static* *final* Logger LOG = Logger.getLogger("SkeletonValidator");

79

80 *protected* *final* *boolean* validateClassName(String className) {

81

82 *boolean* iscorrect = *false*;

83 *try* {

84 Class.forName(className);

85 iscorrect = *true*;

86 LOG.info("Class Name " + className + " is correct");

87

88 } *catch* (ClassNotFoundException e) {

89 LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "

90 + "and class name as provided in the skeleton");

91

92 } *catch* (Exception e) {

93 LOG.log(Level.SEVERE,

94 "There is an error in validating the " + "Class Name. Please manually verify that the "

95 + "Class name is same as skeleton before uploading");

96 }

97 *return* iscorrect;

98

99 }

100

101 *protected* *final* *void* validateMethodSignature(String methodWithExcptn, String className) {

102 Class cls = *null*;

103 *try* {

104

105 String[] actualmethods = methodWithExcptn.split(",");

106 *boolean* errorFlag = *false*;

107 String[] methodSignature;

108 String methodName = *null*;

109 String returnType = *null*;

110

111 *for* (String singleMethod : actualmethods) {

112 *boolean* foundMethod = *false*;

113 methodSignature = singleMethod.split(":");

114

115 methodName = methodSignature[0];

116 returnType = methodSignature[1];

117 cls = Class.forName(className);

118 Method[] methods = cls.getMethods();

119 *for* (Method findMethod : methods) {

120 *if* (methodName.equals(findMethod.getName())) {

121 foundMethod = *true*;

122 *if* (!(findMethod.getReturnType().getName().contains(returnType))) {

123 errorFlag = *true*;

124 LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName

125 + "' method. Please stick to the " + "skeleton provided");

126

127 } *else* {

128 LOG.info("Method signature of " + methodName + " is valid");

129 }

130

131 }

132 }

133 *if* (!foundMethod) {

134 errorFlag = *true*;

135 LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName

136 + ". Do not change the " + "given public method name. " + "Verify it with the skeleton");

137 }

138

139 }

140 *if* (!errorFlag) {

141 LOG.info("Method signature is valid");

142 }

143

144 } *catch* (Exception e) {

145 LOG.log(Level.SEVERE,

146 " There is an error in validating the " + "method structure. Please manually verify that the "

147 + "Method signature is same as the skeleton before uploading");

148 }

149 }

150

151 }

## Grade

Reviewed on Monday, 7 February 2022, 6:34 PM by Automatic grade  
**Grade** 74 / 100  
**Assessment report**

*Fail 1 -- test4CheckTheOutput::*

*$Expected output:"[Print participant details*

*ParticipantName=Weni*

*Yearofstudy=3*

*Department=civil*

*CollegeName=vjc*

*EventName=robocar*

*RegistrationFee=1000.0*

*ParticipantName=gina*

*Yearofstudy=2*

*Department=mech*

*CollegeName=vjc*

*EventName=quiz*

*RegistrationFee=300.0*

*ParticipantName=jos*

*Yearofstudy=4*

*Department=ece*

*CollegeName=vjec*

*EventName=games*

*RegistrationFee=100.0*

*ParticipantName=fida*

*Yearofstudy=1*

*Department=eee*

*CollegeName=vjec*

*EventName=papertalk*

*RegistrationFee=500.0*

*Enter the event to search*

*Number of participants for PAPERTALK event is 1]" Actual output:"[Enter the number of entries*

*Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName*

*Print participant details*

*Participant [name=Weni*

*yearofstudy=3*

*department=civil*

*collegeName=vjc*

*eventName=robocar*

*registrationFee=1000.0]*

*Participant [name=gina*

*yearofstudy=2*

*department=mech*

*collegeName=vjc*

*eventName=quiz*

*registrationFee=300.0]*

*Participant [name=jos*

*yearofstudy=4*

*department=ece*

*collegeName=vjec*

*eventName=games*

*registrationFee=100.0]*

*Participant [name=fida*

*yearofstudy=1*

*department=eee*

*collegeName=vjec*

*eventName=papertalk*

*registrationFee=500.0]*

*Enter the event to search*

*No participant found]"$*

*Check your code with the input :Weni/3/civil/vjc/robocar*

*gina/2/mech/vjc/quiz*

*jos/4/ece/vjec/games*

*fida/1/eee/vjec/papertalk*

*Fail 2 -- test6CheckTheOutputfor\_NCount::*

*$Expected output:"[Print participant details*

*ParticipantName=philip*

*Yearofstudy=4*

*Department=eee*

*CollegeName=mvc*

*EventName=robocar*

*RegistrationFee=1000.0*

*ParticipantName=susan*

*Yearofstudy=4*

*Department=eee*

*CollegeName=mvc*

*EventName=robocar*

*RegistrationFee=1000.0*

*ParticipantName=vivek*

*Yearofstudy=3*

*Department=civil*

*CollegeName=mvc*

*EventName=quiz*

*RegistrationFee=300.0*

*ParticipantName=vishal*

*Yearofstudy=3*

*Department=civil*

*CollegeName=mvc*

*EventName=papertalk*

*RegistrationFee=500.0*

*Enter the event to search*

*Number of participants for ROBOCAR event is 2]" Actual output:"[Enter the number of entries*

*Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName*

*Print participant details*

*Participant [name=philip*

*yearofstudy=4*

*department=eee*

*collegeName=mvc*

*eventName=robocar*

*registrationFee=1000.0]*

*Participant [name=susan*

*yearofstudy=4*

*department=eee*

*collegeName=mvc*

*eventName=robocar*

*registrationFee=1000.0]*

*Participant [name=vivek*

*yearofstudy=3*

*department=civil*

*collegeName=mvc*

*eventName=quiz*

*registrationFee=300.0]*

*Participant [name=vishal*

*yearofstudy=3*

*department=civil*

*collegeName=mvc*

*eventName=papertalk*

*registrationFee=500.0]*

*Enter the event to search*

*No participant found]"$*

*Check your code with the input :philip/4/eee/mvc/robocar*

*susan/4/eee/mvc/robocar*

*vivek/3/civil/mvc/quiz*

*vishal/3/civil/mvc/papertalk*

*robocar*

**Obtained Pass Percentage. Still few testcases failed . Kindly revisit the Solution**  
[[+]](javascript:void(0);)**Grading and Feedback**

Powered by

 ====================================================================