**Project Assessment**

**GENERAL INSTRUCTIONS:**Please carefully read the below instructions

The objective of this assessment is to check your ability to complete a project as per the provided “Project Design”.

**You are expected to –**

1.      Write the source code for the classes, methods and packages **EXACTLY** as mentioned in the “**Project Design**” section.

2.      Ensure that the names of the packages, classes, methods and variables **EXACTLY MATCH** with the names specified in the “Project Design” section.

3.      Understand the project requirements and ACCORDINGLY WRITE the code and logic in the classes and methods so as to meet all given requirements.

**Creating the project and testing it –**

1.      You are expected to create your project locally using eclipse (or any other IDE) on your desktop.

2.      Once you are ready with the code, you should upload the src folder of your project in .zip format, using the “Upload Zip File” button.

IMPORTANT NOTE 1 : The extension of the zip file should be ONLY .zip (any other zip formats such as .7z  will produce unexpected results)

IMPORTANT NOTE 2 : The .zip file should contain zip of ONLY the src folder structure from your project. (If the zip file has anything other than the src folder structure, the result will be unexpected. Do not zip the entire project folder structure. Just do the zip of the src folder structure and upload it)

IMPORTANT NOTE 3 : The name of the .zip file should be <your employee number>.zip For e.g., if your emp no. is 12345, the zip file should be named 12345.zip.

3.      After uploading the zip file, you can click on “Compile & Test” button and the assessment engine will compile your source code and test it using its pre-defined test-cases.

4.      If some of the test-cases fail, you can make the fixes in your source code locally on your desktop, and again repeat the above two steps.

5.      Once you are finished with all the fixes, you can click on “Final Submission” button, which will show you the final result/score.

**NOTE that –**

6.      The assessment engine will create objects and invoke methods as per the project design, and while doing so, it will use your packages, classes and methods. If your packages, classes and methods have a name mismatch or method prototype mismatch w.r.t the expected “Project Design”, the tool will show it as an ERROR. If your packages, classes and methods match as per the names but do not perform the expected functionality, the tool will show it as a FAILURE.

7.      Unless specified in the Project Design, DO NOT use **System.exit(0)** anywhere in your code. Using **System.exit(0)** in your project code will cause the CPC test engine to exit and it will not be able to run all test-cases.

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

**Participant Management System**

**Project Objective:**

The Technical co-ordinator of AZB company wants to compute the training score of participants and insert those details into the database.

The co-ordinator will receive the following data from the user:

1.      The name of the candidate

2.      2 Theory marks [ Maximum marks 40]

3.      2 Practical marks. [ Maximum marks 60]

4.      If the person has Sports quota(YES/NO)

5.      The Id of the participant needs to be computed.

6.      The total marks of the participant need to be computed. The total marks is the sum of average of the 2 theory marks and average of the 2 practical marks

7.      The result needs to be computed as follows

a.      A person **with Sports Quota** is considered PASS if he gets 70% and FAIL if not

b.      A person **without Sports Quota** is considered PASS if he gets 75% and FAIL if not

**Project Design:**

**A. Database Design:**

**1.      Create a new user in database [ To be done in the backend by using sql commands ]**

a)      **Note:**Do **NOT** use the default scott/tiger account of oracle for this project.  You will have to create a new user in the below mentioned format.

b)      Username/password :  B<batchnumber><employeeid>

For example, if your batch number is **39806** and Employee number is **12345**, then the oracle user should be **B3980612345** and the password should be **B3980612345**

c)      For JDBC connection, only use **orcl** as service name and **1521** as port number

**2.      Steps for creating a new user**

a)      Open command prompt

b)      sqlplus / as sysdba

c)      Create user <username> identified by <password>;     [ For example to create a user named “test” with password “test” : create user test identified by test; ]

d)      Grant connect,resource to <username>;  [ E.g: grant connect,resource to test;]

e)      Commit;

f)       Exit;

**3.      Create Table [ To be done using sql commands, after logging-in as the new user that has been created in above step ]**

**Table Name : PARTICIPANT\_TABLE**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| **ParticipantID** | Varchar2(6) | This field is the Primary Key. |
| **ParticipantName** | Varchar2(15) |  |
| **TheoryMark1** | Number(2) |  |
| **TheoryMark2** | Number(2) |  |
| **PracticalMark1** | Number(2) |  |
| **PracticalMark2** | Number(2) |  |
| **SportsQuotaPresent** | Varchar2(3) | **Can take only ‘Yes’ or ‘No’**  **[Case Sensitive]** |
| **Total** | Number(3) |  |
| **Result** | Varchar2(15) |  |

**4.      Create Sequence:**

**Sequence Name : PARTICIPANTID\_SEQ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sequence Name** | **Minimum Value** | **Max Values** | **Incremental value** | **Start Value** |
| **PARTICIPANTID\_SEQ** | 3000 | 9999 | 1 | 3000 |

**B. System Design:**

|  |  |
| --- | --- |
| **Name of the package** | **Usage** |
| com.wipro.participant.service | This package will contain the class which takes the user input, validates it and invokes the DAO methods |
| com.wipro.participant.bean | This package will contain the bean class |
| com.wipro.participant.dao | This package will contain the DAO class that will do the database related operation |
| com.wipro.participant.util | This package will contain the class to establish database connection and also the class that handles the user defined exception. |

**Package: com.wipro.participant.util**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **DBUtil** |  | DB connection class |
|  | public static Connection **getDBConnection**() | Establish a connection to the database and return the java.sql.Connection reference |
| **InvalidInputException** |  | User defined exception class. |
|  | public Sting toString() | Should return the String **“Invalid Data in Input”**. The details about when it must be thrown is given in the respective methods of the Administrator class |

**Package: com.wipro.participant.bean**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **ParticipantBean** |  | Bean class |
|  | private String id; | Participant Id |
|  | private String name; | Participant Name |
|  | private  int theoryMark1; | Marks in theory1 –Max mark 40 |
|  | private  int theoryMark2 | Marks in theory2 – Max marks 40 |
|  | private  int practicalMark1; | Marks in practical1 – Max marks 60 |
|  | private int practicalMark2; | Marks in practical1 – Max marks 60 |
|  | Private boolean sportsQuotaPresent; | ‘**Yes**’ if he has SportsQuota, ‘**No**’ otherwise |
|  | private int total; | Total Marks – Max marks 100 |
|  | private String result | Result of the Student |

**Package: com.wipro.participant.dao**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **ParticipantDAO** |  | DAO class |
|  | public String **generateId**(String studentname) | ·         This method generates the Participant id and returns it.  ·         The participant id is a combination of last two letters of the name in uppercase followed by the sequence number generated by **PARTICIPANTID\_SEQ**  ·         Eg) NA2000-Here name of the participant is Veena and the sequence number generated by the sequence is 3000 |
|  | public String createParticipant(ParticipantBean bean) | ·         This method should insert the bean values into the database  ·         If the insertion of data into the table was successful, the method should return “**SUCCESS**”  ·         If it could not insert the values into the table, the method should return “**FAIL**” |

**Package: com.wipro.participant.service**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **Administrator** |  | Main class |
|  | public static void **main**(String[] args) | The code that is needed to test your program goes here. A sample code is shown at the end of the document. |
|  | public String **addParticipant**(ParticipantBean bean) | ·         The following are the conditions under which a user defined exception **InvalidInputException** should be thrown.  Ø  If bean is null  Ø  If bean’s name is empty String  Ø  If bean’s name contains less than 2 characters  Ø  If bean’s sportsQuotaPresent contains anything other than ‘Yes’ or ‘No’  **[Note:Case Sensitive]**  ·         This exception should be handled within the **addPariticipant** method itself.  ·         If this exception is caught, then this method is expected to return a String “**Invalid Data in Input**”.  **NOTE:** Do **NOT**use **System.exit(0)** while handling the exception.  ·         If bean’s theoryMark1 and theoryMark2 contains value, lesser than 0 or greater than 40, then this method should return ”**THEORY MARK IS INVALID**”  ·         If bean’s practicalMark1 and practicalMark2 contains value lesser than 0 or greater than 60, the method should return ”**PRACTICAL MARK IS INVALID**”  ·         **Compute Participant ID**  Ø  If the bean contains valid values, this method should call the generateId method ofDAO  class to obtain the participantId. Inititalize the bean with the value received  ·         **Compute total and result**  ·         The total(out of 100) is the sum of the average of the 2 theoryMarks and the average of two Practical Marks  ·         For eg) If a participant has scored the following marks, his total marks must be computed as follows:  TheoryMark1-35  TheoryMark2-46  PracticalMark1-56  PracticalMark2-48  Then total=(29+52)=81  ·         Initialize the bean’s total with the computed value  ·         Compute the result and initialize the bean  ·         If the bean’s sportsQuotaPresent is true, the result is “**PASS**” if the participant scores 70 and above, else result is “**FAIL**”  ·         If the bean’s sportsQuota is false, the result is “**PASS**” if the participant scores 75 and above, else result is “**FAIL**”  ·         Initialize the bean’s result property  ·         Invoke **createParticipant method** of the DAO class to insert the bean into the database.  ·         On successful storage of the participant details, to the table,  the function should return the Id and result of the particular ParticipantBean  ·         [ For Eg, if the Student ID generated is   VE2000, and the result is PASS then the success message should be **VE2000:PASS**  ·         If by any reason, the record is not stored, then the function should  return the String **Error** |

**Main Method:**

**You can write code in the main method and test all the above test cases. A sample code of the main function to test the first test case is shown below for your reference.**

public static void main(String[] args) {

ParticipantBean bean=new ParticipantBean();

bean.setName("ABC");

bean.setPracticalMark1(40);

bean.setPracticalMark2(40);

bean.setTheoryMark1(30);

bean.setTheoryMark2(30);

bean.setSportsQuotaPresent("Yes");

Administrator admin= new Administrator();

System.out.println(admin.addParticipant(bean));

}