**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.

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

**Candidate Management System**

**Project Objective:**

Create a console based Java application that would allow the Admin of a training Institute to add and view Candidate information details as per the design specifications given below.  The data received from the user (Admin) will be stored in database and retrieved when required.

**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)      Type 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 : CANDIDATE\_TBL**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Description** |
| **ID** | Varchar2(6) | This field is the Primary Key. |
| **Name** | Varchar2(15) |  |
| **M1** | Number(3) |  |
| **M2** | Number(3) |  |
| **M3** | Number(3) |  |
| **Result** | Varchar2(15) |  |
| **Grade** | Varchar2(15) |  |

**4.      Create Sequence:**

**Sequence Name : CANDID\_SEQ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sequence Name** | **Minimum Value** | **Max Values** | **Incremental value** | **Start Value** |
| **CANDID\_SEQ** | 5000 | 7000 | 1 | 5000 |

**B. System Design:**

|  |  |
| --- | --- |
| **Name of the package** | **Usage** |
| **com.amar.candidate.service** | This package will contain the class which displays the console menu and take user input. |
| com.amar.candidate.bean | This package will contain the entity class named CandidateBean. |
| com.amar.candidate.dao | This package will contain the class that will do the database related JDBC code. |
| com.amar.candidate.util | This package will contain the class to establish database connection and also the class that handles the user defined exception. |

**Package: com.amar.candidate.util**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **DBUtil** |  | DB connection class |
|  | public static Connection **getDBConn()** | Establish a connection to the database and return the java.sql.Connection reference |
| **WrongDataException** |  | User defined exception class.  Override the toString() method of the Object class and return a String **“Data Incorrect”**. This exception will be thrown in the CandidateMain class methods whenever invalid input is given. The details about when it has to be thrown is given in the respective methods of the CandidateMain class |

**Package: com.amar.candidate.bean**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **CandidateBean** |  | **Bean class** |
|  | private String id | Student Id |
|  | private String name | Student name |
|  | private  int m1 | Mark in First Subject |
|  | private  int m2 | Mark in Second Subject |
|  | private  int m3 | Mark in Third Subject |
|  | private String result | Result |
|  | private String grade | Grade |
|  | setters & getters | Should create the getter and setter methods for all the attributes mentioned in the class |

**Package: com.amar.candidate.dao**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **CandidateDAO** |  | DAO class |
|  | public String **addCandidate**(CandidateBean CandidateBean) |          This method should take the values from the CandidateBean object and insert it into the database.           If the insertion is successful, then a String “SUCCESS” should be returned, else a String “FAIL” should be returned.           If any JDBC exception such as SQLException occur, this function should return “FAIL” |
|  | public ArrayList<CandidateBean> **getByResult**(String criteria) |          This method should use the JDBC select statement to retrieve the records based on the criteria given.           If the criteria String contains **“PASS”** then the **getByResult(String criteria)**function should return an ArrayList of all Candidates who have passed           If the criteria String contains **“FAIL”** then the **getByResult(String criteria)**function should return an  ArrayList of all Candidates who have failed           If the criteria String contains **“ALL”** then the **getByResult(String criteria)**function should return an ArrayList of all the Candidates           In any of the criteria’s “PASS/FAIL/ALL” if there are no matching records then the function should return **null**           In case of any JDBCExceptions in the database then a **null** value needs to be returned |
|  | public String **generateCandidateId**(String name) |          This method should contain the necessary code to create a new Candidate id.           CandidateID is a combination of **first 2 letters of name** in **uppercase** followed by **4 digit number that will be generated by the oracle sequence CANDID\_SEQ.**           For eg, the Candidate id for a Candidate Jacob could be JA2194           The function should return the generated Candidate id. |

**Package: com.amar.candidate.service**

|  |  |  |
| --- | --- | --- |
| **Class** | **Method and Variables** | **Description** |
| **CandidateMain** |  | 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 **addCandidate**(CandidateBean candBean) |          This method should add a CandidateBean Object to the database.           The following are the conditions under which a user defined exception **WrongDataException** (found in **com.amar.candidate.util** package) should be thrown.    candBean is null    candBean’s name is empty String    candBean’s name contains less than 2 characters     candBean’s mark1,mark2,mark3 contains marks which are  not within 0 to 100 range.           This exception should be handled within the **addCandidate(CandidateBean candBean)** function itself.           If this exception is caught, then the function is expected to return a String “**Data incorrect**”.  **NOTE:** Do **NOT**use **System.exit(0)** while handling the exception.           **Compute Candidate ID**    If the candBean object is valid, this function should call the **generateCandidateId (String name)** function of the **CandidateDAO**  class to obtain the candidate id. The candBean’s name should be passed as parameter to the **generateCandidateId (String name)** function           **The candBean’s id should be initialized using the Candidate Id that is received in the previous step**           **Compute result and grade**    The result and grade are computed using the following logic    M1=mark1 of candBean    M2=mark2 of candBean    M3=mark3 of candBean   |  |  |  | | --- | --- | --- | | **Total Marks** | **Result** | **Grade** | | (M1+M2+M3)>= 240 | PASS | Distinction | | (M1+M2+M3)>=  180 and (M1+M2+M3)<240 | PASS | First Class | | (M1+M2+M3)>=  150 and (M1+M2+M3)<180 | PASS | Second Class | | (M1+M2+M3)>= 105 and (M1+M2+M3) <150 | PASS | Third Class | | (M1+M2+M3)  <105 | FAIL | No Grade |            **Initialise the candBean’s result and grade with the computed values**           Invoke **addCandidate(CandidateBean CandidateBean)** of the CandidateDAO class to insert the candBean into the database.           On successful storage of the Candidate Details, to the table,  the function should return the CandidateID and result of the particular CandidateBean           [ E.g if the Candidate ID generated is   SA1001, and the result is PASS then the success message should be **SA1001:PASS**           If by any reason, the record is not stored, then the function should  return the String **Error** |
|  | public ArrayList<CandidateBean> displayAll(String criteria) |          This method should return the collection of the Candidates from the Candidate table who are matching the given criteria           The criteria string can have values such as “PASS/FAIL/ALL”.           If the criteria contains either “PASS/FAIL/ALL” then invoke **getByResult(String criteria) of CandidateDAO class and receive the collection**           If the criteria String contains any other values then the WrongDataException need to be thrown, and the function should return a null value.           **NOTE:** Do **NOT**use **System.exit(0)** while handling the exception. |

**Test Cases:**

**Below are the actual set of test cases that the CPC test engine will run in the background. Please ensure that the conditions mentioned in these test-cases are handled by your class design.**

Test 1: Test for Null Value for CandidateBean

Test 2: Test for Candidate Name to be Empty

Test 3: Test for Candidate Name less than 2 characters

Test 4: Test for Invalid range of Marks

Test 5: Test for correct CandidateId generation

Test 6: Test for checking correct computation of PASS, FAIL and grade

Test 7: Test for null value generation for different criteria (PASS/FAIL/ALL)

Test 8: Test the number of records retrieved for different criteria

**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) {

CandidateMain candidateMain = new CandidateMain();

String result = candidateMain.addCandidate(null);

System.out.println(result);

}