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.

^^^

Interest Calculation

Project Objective:

Create a console based Java application that would allow the clerk of a bank to compute and inform the clients how much will be the maturity amount for recurring deposit for a tenure of 'x' years

Recurring Deposit:

Recurring deposit (RD) scheme is offered by almost all banks, which help people with regular incomes. Under this scheme, the customer deposits a minimum fixed amount every month, and bank pays the interest at the pre-determined rates

Project Design:

A. System Design:

Name of the package	**Usage**
com.wipro.bank.acc	This package will contain the Account related classes
com.wipro.bank.exception	This package will contain the user defined exception class
com.wipro.bank.main	This package will contain the MainClass that is used to test the application
com.wipro.bank.service	This package will contain the class that is used to validate the data and invoke the Account Classes to calculate maturity amount

Package: com.wipro.bank.exception

Class	**Method and Variables**	**Description**
BankValidationException		**An Exception Class**
	public String toString()	Returns the message "Invalid Data"

Package: com.wipro.bank.acc

Class	**Method and Variables**	**Description**
Account		Abstract Class
	int tenure;	Total number of **years**
	float principal;	Principal amount
	float **rateOfInterest**;	Interest rate
	public void setInterest(int age, String gender)	This method is used to find the interest rate based on the below calculation: Gender / Age / Rate of Interest Male / <60 / 9.8 Male / >=60 / 10.5 Female / <58 / 10.2 Female / >=58 / 10.8 • This function should**initialize** the**rateOfInterest**membervariable with the correct rateOfInterest based on the calculation that is given above
	public float calculateMaturityAmount(float totalPrincipleDeposited, float maturityInterest)	This function returns the sum of the totalPrincipleDeposited and the maturityInterest
	public abstract float calculateInterest();	This is an abstract method used to calculate the interest
	public abstract float calculateAmountDeposited();	This is an abstract method which returns the amount the user has deposited for the given tenure

Package: com.wipro.bank.acc

Class	**Method and Variables**	**Description**
RDAccount		Inherits Account Class
public RDAccount(int tenure, float principal)		• Parameterized Constructor
public float calculateAmountDeposited()		• Should return the total amount accumulated at the end of the tenure • For eg) for a principal of 1000 Rs per month, and for a tenure 5 years the function should return 60000 • (i.e principal*tenure*12)
public float calculateInterest();		• In RD Account the interest is Quarterly Compounding interest. • P * (((1+r/n)^nt) - 1) • Where p – Principal

	• r– rateofinterest/100 • n - 4 (no of quarters in a year) • t – no of months remaining converted as years (60/12) An example calculation is given in the below table Note: declare all local variables as float and no rounding of data anywhere in calculation. Use : Math.pow

RD Interest Calculation Table

- For eg) if the user is depositing 1000 Rs per month and rateOfInterest is 9.8 then the following is the calculation for a tenure of 5 years:
  …. Till 60 months.
  This function should then return 17491.52 which is the total interest earned

month	Principal [p]	Rate Of Interest [r]	r/100	1+r/n	months remaining	Remaining Months as years [t]	nt	(1+r/n)^nt	cRate ((1+r/n)^nt) - 1	Interest paid P*cRate
1	1000	9.8	0.098	1.0245	60	5	20	1.622703806	0.622703806	622.7038
2	1000	9.8	0.098	1.0245	59	4.916666667	19.66667	1.609664133	0.609664133	609.6641
3	1000	9.8	0.098	1.0245	58	4.833333333	19.33333	1.596729245	0.596729245	596.7292
4	1000	9.8	0.098	1.0245	57	4.75	19	1.583898298	0.583898298	583.8983
5	1000	9.8	0.098	1.0245	56	4.666666667	18.66667	1.571170457	0.571170457	571.1705

Package: com.wipro.bank.service

Class	**Method and Variables**	**Description**
BankService		Class
	public boolean validateData(float principal, int tenure, int age, String gender)	• This method is used to check if the input value given by the user's is a valid or not The following are the conditions of a valid data • principal for RD should be minimum 500 • tenure should be either 5 or 10 only • gender can be either male or female**not case sensitive** • age should be between 1 to 100

		• If any of the values are invalid the user should throw the user defined exception **BankValidationException** • The exception should be caught in the same function. The function should return a false value in case of the exception • If all the values are valid, the function should return true
	public void calculate(float principal,int tenure, int age, String gender)	• This method will first invoke the validateData method. • If the validateData method returns true, only then the following operations are performed ➢ creates RDAccounts Object ➢ Invokes the setInterest method passing age and gender and get the rateOfInterest initialized ➢ Invoke the calculateInterest method and print the returned value ➢ Invoke the calculateAmountDeposited() method and print the returned value ➢ Invoke the calculateMaturityAmount () method and print the value

Package : com.wipro.bank.main

Class	**Method and Variables**	**Description**
MainClass		**Main Class**
	public static void main(String[] args)	Get the following input from the user 1. Get the tenure period [Tenure can be either 5/10] 2. Get the Principal amount [minimum principal amount is 500] 3. Get User age 4. Get gender (male/female – not case sensitive) After receiving all this data, invoke the calculate method of BankService class present in com.wipro.bank.service package and test your program

The main method of the Main Class may look like this:

```
public static void main(String a[])
{
        int tenure = 5;
        float principal = 1000;
        int age = 23;
        String gender = "Male";
        BankService b=new BankService();
        b.calculate(principal, tenure, age, gender);
}
```

Change the values of variables declared and test your program, if it is generating the correct output.