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.

^^

Credit Card Bill Generator

Project Objective:

Create a console based Java application that would allow users to generate CreditCard bill based on annual interest rate for the given month.

Project Design:

A. System Design:

Name of the package	Usage
com.wipro.ccbill.entity	This package will contains the credit card bill and transaction related classes
com.wipro.ccbill.exception	This package will contains the user defined exception class
com.wipro.ccbill.main	This package will contains the MainClass that is used to test the application

Package: com.wipro.ccbill.exception

Class	Method and Variables	Description
InputValidationException		**An Exception Class**
	public String toString()	Override toString() method such that it Returns the message "**Invalid Input Data**"

Package: com.wipro.ccbill.entity

Class	Method and Variables	Description
Transaction		
	private String creditCardNo	
	private Date dateOfTransaction	java.util.Date
	private String transactionDesc	
	private double transactionAmount	
	private String transactionType	

	public Transaction ()	No args constructor
	public Transaction (String creditCardNo, Date dateOfTransaction, String transactionDesc, double transactionAmount, String transactionType)	Overloaded constructor to initialize all the member variables
	Create getters and setters	Right click on eclipse and generate getters and setters from Source

Package: com.wipro.ccbill.entity

Class	Method and Variables	Description
CreditCardBill		
	private String creditCardNo	
	private String customerId	
	private Date billDate	
	private Transaction monthTransactions[]	object array of Transaction class
	public CreditCardBill ()	No args constructor
	public CreditCardBill (String creditCardNo, String customerId, Date BillDate, Transaction monthTransactions[])	Overloaded constructor to initialize the input string
	public double getTotalAmount(String transactionType)	This method iterates throughmonthTransactions[] and returns the total amount of the given transaction type Valid transaction Type DB- amount spent on card CR- amount paid to card If given transactionType is not CR or DB return 0
	public double calculateBillAmount()	This method calls the 1. Call validateData method, if it returns “valid” proceed to step 2 else handle the exception and return 0.0 2. Check whether the monthsTransaction is not null and array length is more than 0 and proceed to step 3 else if there are no transactions for the month return 0.0 3. getTotalAmount with debit(DB) type and gets the

		total amount spend on card. 4. getTotalAmount with credit(CR) type and gets the total amount paid to card. 5. Difference is the outstanding to be paid 6. Calculate the interest for the outstanding using the given formula **Interest calculated = (outstanding amount x 19.9% per year / 12)** 7. Add up outstanding amount and interest calculated as bill amount and return the same
	public String validateData() throws InputValidationException	This checks whether the given data is valid or not. 1. CreditCardNo should not be null 2. CreditCardNo should be of length 16 exactly 3. CustomerId should not be null 4. CustomerId should be of length 6 minimum • If any of these condition fails the function should throw**InputValidationException** • If all condition passes return "**valid**".

Package : com.wipro.ccbill.main

Class	Method and Variables	Description
MainClass		**Main Class**
	public static void main(String[] args)	Checks the application: Creates CreditCardBill object and passes input to all the fields. Gets the total bill amount to be paid. **Note:** Sample code given below

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

public static void main(String a[])

{

```
try{
      Transaction monthsTransaction[] = newTransaction[5];
      SimpleDateFormat formatter = newSimpleDateFormat("dd/MM/yyyy");


monthsTransaction[0]=newTransaction("1111222233334444",formatter.parse("01/02
/2016"),"household",8000.0,"DB");

monthsTransaction[1]=newTransaction("1111222233334444",formatter.parse("05/02
/2016"),"Textile",12000.0,"DB");

monthsTransaction[2]=newTransaction("1111222233334444",formatter.parse("12/02
/2016"),"movie",2700.0,"DB");

monthsTransaction[3]=newTransaction("1111222233334444",formatter.parse("16/02
/2016"),"paid",19000.0,"CR");

monthsTransaction[4]=newTransaction("1111222233334444",formatter.parse("19/02
/2016"),"household",4500.0,"DB");
      CreditCardBill ccbill
= newCreditCardBill("1111222233334444","ABC123",formatter.parse("02/03/2016")
,monthsTransaction);
      System.out.println("Total to be paid : "+ccbill.calculateBillAmount());
      }catch(Exception e){}

}
```

This prints: **`Total to be paid : 8335.983`**

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

Test Cases

The following are some of the test cases used in the Program.

1. Test for creditcard details are valid
2. Test for invalid credit card details
3. Test total amount already paid (DB)
4. Test total amount to be paid(CR)
5. Test total amount for invalid transaction type
6. Test total amount to be paid with interest
7. Test for null Transaction Array
8. Test for zero length Transaction Array
9. Test for Customer Id is null
10. Test if Exception is thrown

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 main method that you write **will not** be used to evaluate your code.

7. 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.

^^

PROJECT DESCRIPTION: Bubble Sort

- Create a program to arrange the elements of a given array of size 5 in ascending order.

Sorting elements in the array in ascending order:-
If the array is {56, 34, 1, 89, 3}, then the output should be {1, 3, 34, 56, 89}

PROJECT REQUIREMENTS:

Functional Requirements:

This table provides list of basic requirements of this system. These can be understood in details by looking at the class specifications in the “Project Design” section below.

Requirement ID	Requirement
REQ-01	An array of five numeric elements should be sorted in ascending order and stored in the same array.
REQ-02	If the user is not entering proper values, the array should be set to null and it should result in a user defined exception.
REQ-03	If the user is entering elements less than 5, then the program should display an array of that size containing only zeros.

Project Design:

Name of the package	Usage
com.wipro.sort.bubblesort	Contains the class that performs the bubble sort
com.wipro.sort.main	Contains the main class
com.wipro.sort.exceptions	Contains the user defined exception classes

Package: com.wipro.sort.bubblesort

Class	Method and Variables	Description
BubbleSort		
	`public int[] sort(int array[])`	The array containing the numbers to be sorted is passed to the sort method. The method will return the sorted array.

Package: com.wipro.sort.main

Class	Method and Variables	Description
MainClass	**Main class**	
	`static void main(String args[])`	This is the main method of your program, which you would use to check your code locally. However, please note that the assessment engine will use its own methods to test the above class specifications. This method should receive 5 command line arguments.

Package: com.wipro.sort.exceptions

Class	Method and Variables	Description
EmptyArrayException	**Exception class.**	**Raise this exception when the array passed to the sort() method of BubbleSort class is null.**
	String toString()	Used to specify an error message.

Test-cases:

This table provides list of test-cases that will be run to test the encoder-decoder system. While creating the class specifications, care must be taken to ensure that these conditions are met.

Test-Case			
If the following methods are executed, the output should be the values specified as below.			
S. No.	**Method being tested**	**Code**	**Desired Output from the method**
1	`publicint[] sort(int array[])ofBubbleSortclass`	`int [] inputArray= {23,1,9,3,0}; BubbleSort bSort=newBubbleSort(); array=bSort.sort(inputArray);`	{0,1,3,9,23}
2	`publicint[] sort(int array[])ofBubbleSortclass`	`int [] inputArray1= {1,2,3,4,5}; BubbleSort bSort=newBubbleSort(); array=bSort.sort(inputArray);`	{1,2,3,4,5}
3	`publicint[] sort(int array[])ofBubbleSortclass`	`int [] inputArray2= {1,2,1,4,1}; BubbleSort bSort=newBubbleSort(); array=bSort.sort(inputArray);`	{1,1,1,2,4}
4	`publicint[] sort(int array[])ofBubbleSortclass`	`int [] inputArray3= {5,5,5,5,5}; BubbleSort bSort=newBubbleSort(); array=bSort.sort(inputArray);`	{5,5,5,5,5}
5	`publicint[] sort(int array[])ofBubbleSortclass`	`int [] inputArray4=null; BubbleSort bSort=newBubbleSort(); array=bSort.sort(inputArray);`	null
6	`publicint[] sort(int array[])ofBubbleSortclass`	`int [] inputArray5={12,23}; BubbleSort bSort=newBubbleSort(); array=bSort.sort(inputArray);`	{0,0}

The skeleton of the main() method is given below –

You are expected to code and complete the remaining class hierarchy as per above specifications so that the program works as expected.

```
public static void main(String[] args) {
            int length = args.length;
            int [] array= new int[5];
```

```
                if (length ==0)
                      {
                            array=null;
                      }else{
                // ASSIGNING  the arguments elements to ARRAY

                try{
                for (int i=0;i<length ;i++)
                {
                      array[i]= Integer.parseInt(args[i]);
                }
                }catch(NumberFormatException eNum){
                      array=null;
                }
                      }
                BubbleSort bSort=new BubbleSort();
                array=bSort.sort(array);
                if(array!=null){
                System.out.println("Output:");
          for (int i=0;i< length ;i++){
                      System.out.print( "   " + array[i] );
          }
                }
                else{
                      System.out.println("No ouput");
                }
    }
```

Sample Output:

D:\java>java MainClass 23 1 9 3 0

```
Output:
   0   1   3   9   23
```

D:\java>java MainClass 1 2 3 4 5

```
Output:
   1   2   3   4   5
```

D:\java>java MainClass 1 2 1 4 1

```
Output:
   1   1   1   2   4
```

D:\java>java MainClass 5 5 5 5 5

```
Output:
   5   5   5   5   5
```

D:\java>java MainClass

```
The array is null
```

D:\java>java MainClass 23 45

```
Output:
   0   0
```