**Problem Description**

**Important Instructions:**

* Please read the document thoroughly before you code
* Do not change the given '**input.txt**' file
* Do not change the skeleton code or the class name, method names, variable names, return types, 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

**Concepts Coverage:**

* Control Flow Statements
* Strings
* Functions
* Classes and Objects
* Collection Framework
* File Handling
* Exception Handling
* Database Handling

**Problem description:**

The finance department of a popular manufacturing unit is in the act of automating its business processes. As part of this process, one module of the Reimbursement Process needs to be automated.

The department gets numerous Reimbursement requests and processing them manually takes enormous time, therefore in order to automate, they need an application in Python programming language that filters only those requests which are **approved by the respective manager** and the requests which are**not older than 6 months(180 Days)**, then updates the corresponding local travel charges, accommodation charges and dining costs and finally calculates the total cost to be reimbursed and display this information as per the requirements.

Reimbursement requests are available in the text files in the coma separatedformats. A module that is to be developed should enable one to search those requests which have been approved by the manager and requests raised within a stipulated time, subsequently based on the grade, the number of days of stay, no. of km. of local traveling, the respective reimbursement amounts are calculated and updated for the record.

**Functional Requirement Specification:**

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Req. Name** | **Req. Description** |
| **1** | Parse Input | The input file has to be parsed and Employees reimbursement records need to be fetched for the **approved request by manager a**nd requests raised **not beyond 6 months after travel - 180 days** |
| **2** | Calculate the Total reimbursement amount and set to the reimburse object and build the list | The Employees' reimbursement amount needs to be calculated and then set the calculated value to the corresponding attribute of the object and add the object to the reimbursement list. |
| **3** | Store each reimbursement object details to the database and Retrieve necessary details based on the requirements | Store the details of each reimbursement object in the list to the database.  Search and retrieve the reimbursement details as per various requirements. |

**Technical Requirements:**

For all the functional requirements 1, 2, 3 and 4, component specification and method Specifications are given below. Please follow the same order to implement them using the code skeleton.

**Requirement 1:**Write the necessary part of the Python class '**EmpReimbursement**'.  All members of the class are **private variables.**

|  |
| --- |
| **EmpReimbursement** |
| request\_id  - string    employee\_code - string    date\_of\_request - date    grade - String    date\_of\_travel - date    no\_of\_days\_of\_stay - int    local\_travel\_in\_kms - double    manager\_approval - string    accomodation\_cost - float    dining\_cost - float    allowances - float    local\_travel\_cost - float    total\_reimbursement\_cost - float |

**Component Specification: EmpReimbursement (Model/Pojo  Class):**

|  |  |  |  |
| --- | --- | --- | --- |
| ***ClassName*** | ***Method Name*** | ***Input Parameters*** | ***Output Parameters*** |
| EmpReimbursement | A parameterized constructor with the arguments: request\_id, employee\_code, date\_of\_request, grade, date\_of\_travel, no\_of\_days\_of\_stay, local\_travel\_in\_kms and manager\_approval  respectively. | request\_id, employee\_code ,date\_of\_request, grade, date\_of\_travel,no\_of\_days\_of\_stay, local\_travel\_in\_kms, manager\_approval.  For other attributes, set its default values as its initial value. |  |
| EmpReimbursement | Included necessary getters and setters for the private attributes |  |  |

**Requirement 2:**Create a module **'utility.py'**to provide some generic utilities.  It contains the below-mentioned methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module name** | **Method Name** | **Input Parameters** | **Output Parameters** | **Description** |
| Utility | read\_file | String filename | List of records of strings | The method '**read\_file'** should read the specified file and return a list of records, that are read from the file; It should filter only records where the requests are approved by a manager and not beyond 6 months from the day of travel. Returns, these records. (with each record's fields separated by comma) |
| validate\_request\_id(request\_id) | request\_id | boolean | This method should accept **request\_id** as input, and validates the request\_id as per the business rules specified below:  This method should return the status as **True** for valid request\_id else, throw a user-defined exception "InvalidRequestIdException" if the request\_id is invalid and return the exception message **'Invalid Request Id'**.  **Business Rule:**  All request id's should have a minimum of **4 characters long**and should start with the letters **R**.  The n**ext two characters** should be the numeric value**0.**  The l**ast character**should be an **integer.** |
|  |  |  |  |
| convert\_date(str\_date) | Date of request or date of travel | Date object | This generic method should take the string date in string format and convert this string into **datetime.date**object method and return the same.  **Note**: No need for a time format and do not use the Pandas package for date conversion. |

**Component Specification: InvalidRequestIdException(Exception Class)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type(Class)** | **Attributes** | **Methods** | **Responsibilities** |
| InvalidRequestIdException |  | Include a constructor with a single argument 'message'  \_\_init\_\_(message) |  |

**Requirement 3:**Create another class '**EmpReimbursementService'** and write the below mentioned private attributes and public methods:

|  |  |  |  |
| --- | --- | --- | --- |
| ***ClassName*** | ***Attribute Name*** | ***Type of attributes*** | ***Responsibilities*** |
| EmpReimbursementService | emp\_reimbursement\_list | List |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Output Parameters** | **Description** |
| EmpReimbursementService | get\_emp\_reimbursement\_details | **.txt file** | List of EmpReimbursement objects | This method takes '**input.txt**' as its parameter and calls the **read\_file** method by passing this file to it as its argument.  The 'read\_file' method will r**eturn a list of string-type records** and pass this list of records to the '**build\_emp\_reimbursement\_list'**  method in the same class.  The method should **return**the **emp\_reimbursement\_list** of  'EmpReimbursementService' class. |
| EmpReimbursementService | calculate\_reimbursement\_costs | no\_of\_days,local\_kms\_travel and grade | List of float values | This method should calculate reimbursement costs like **accommodation costs, dining costs, local\_travel costs, allowances costs, and total reimbursement amounts**and**return**all these values as a**list.**  Calculate the accommodation, dining, local\_travel\_cost, allowances costs, and total Reimbursement amounts based on the constraints in the table given below, all costs in INR - depending on the **Grade and number of days of the stay.**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Grade** | **Accommodation Cost /Day** | **Dining Cost/ Day** | **Local Travel/ Km** | **Allowances/Day** | | Level 01 (Highest) | 10000 | 1000 | 22 | 1500 | | Level 02 | 10000 | 1000 | 22 | 1500 | | Level 03 | 4000 | 700 | 16 | 1000 | | Level 04 | 4000 | 700 | 16 | 1000 | | Level 05 | 2500 | 450 | 12 | 750 | | Level 06 | 2500 | 450 | 12 | 750 |     **Example**: If the number of days of stay is 5 for Level01 and total local travel in km is 150  then,  Accommodation cost = 50000.0  Dining cost = 5000.0  Local travel cost = 3300.0  Allowances = 7500.0  Total Reimbursement cost = 65800.0 |
|  |  |  |  |
| build\_emp\_reimbursement\_list | List of records | None | This method takes a list of records that are returned from the 'read\_file' method. The basic responsibility of this method is to build a list of **'EmpReimbursement'** objects.  This method should read every line from the list passed as the argument.  Then it should call the '**validate\_request\_id'** to validate the **request\_id** from the list.  If valid, then use the '**convert\_date'** method to convert the date from string format to Python's datetime.date format (**yyyy-mm-dd**).  Then build the "EmpReimbursement" objects from the values obtained in every line.  Use the '**calculate\_reimbursement\_costs**' method to get total\_reimbursement\_cost, by passing the travel details like no\_of\_days\_of\_stay, local\_travel\_in\_kms, and grade of the employee.  This method will return **a list of costs**, using that set all the required attribute values using the "**EmpReimbursement"** object.  Finally, add this object to the '**emp\_reimbursement\_list'**attribute in the class  'EmpReimbursementService'. |
| add\_reimbursement\_details | List | None | This method is for connecting to the database and inserting each and every valid reimbursement object in the list into the database. The name of the database table is **'reimbursement'**.  **Note:**The names of the columns of the table are same as that of private attributes of 'EmpReimbursement' class. |
| search\_reimbursement\_request | request\_id | Object of EmpReimbursement | This method should search for the reimbursement object in the database based on the **request\_id** passed as its parameter.  If found return that **'EmpReimbursement**' object else it should return a **None** object. |
|  | update\_costs | no\_days | List of objects | This method should take no\_of\_days\_of\_stay entered by the user as its parameter and it will give a **10% increment** to the **allowances** and **local travel cost** of all reimbursement requests if the no. of days of stay is **greater** than the no. of days entered by the user.  It should update the allowances and total reimbursement  column in the database and **return** all those reimbursement objects that gets updated. |

***Design Constraints:***

a.  Input file format is .txt and is comma-separated (Sample rows are added. You can add any number of rows to test your service class, from the main method.   This file is available on the platform's current directory so that no need to specify the path.

b.   Do not hard code the input file path inside any method - has to be used from the input argument only as per the code skeleton.

c.    File Structure is like below:

        <request\_id>,<employee\_code>,<date\_of\_request>,<grade>,<date\_of\_travel>,<no\_of\_days\_of\_stay>, <local\_travel\_in\_kms>,<manager\_approval>, <accomodation\_cost>, <dining\_cost>, <allowances>, <local\_travel\_cost>  and <total\_reimbursement\_cost>

d.   Assume that the currencies  are in **INR**

e.   Assume that the date fields in the file will be in the format **yyyy-mm-dd.**

f.    Do not change the data types of the value object given in POJO/ Model class

g.   Always convert the date field value to DateTime.date with the format, yyyy-mm-dd before setting in the "**EmpReimbursement"** value object.  Do not use 'datetime' format.  Only the date format is required.

h.    Calculate the Accommodation, Dining, LocalTravel, allowances costs, and total Reimbursement amounts based on the constraints in the table given above

i. Use the Oracle database to store the data.

j. The table names and the column names should be the same as specified in the table structure.

k. Database connection should be configurable; it should not be hard coded. The database information is specified in the database.properties file, which is also provided as part of the code skeleton.

l. Close all the resources after use.

**Note:** The code skeleton is made available in the Tekstac platform.

***Process flow:***

a.   The application will be invoked by calling the EmpReimbursementService. '**get\_emp\_reimbursement\_details'**method with the **input.txt** file as its argument.

b.   '**get\_emp\_reimbursement\_details'**calls the '**read\_file'** method by passing the file to it as an argument.

c.    The**'read\_file'** method should read the file and returns a list of records, that was read from the file; It should filter only records where the requests are approved by the manager and not beyond 6 months from the day of travel.

d.    Returns, these records as a list. (with each record's fields separated by a comma)

e.    Code the method '**build\_emp\_reimbursement\_list '** pass the output of the read\_file method to this method

f.     The '**build\_emp\_reimbursement\_list '**method reads every line from the list taken as its parameter, and validates the **request\_id** as per the business rules by invoking the appropriate method in the utility module.  If the request\_id is not valid, then raise the user-defined exception  'InvalidRequestIdException'. If valid, then convert the string values of the date\_of\_request and date\_of\_travel  field to date object by invoking the '**convert\_date'**method and create the object for **the "EmpReimbursement"** class then calculate and set all the costs by invoking the method '**calculate\_reimbursement\_costs**' method.  Finally, add each of these 'EmpReimbursement' objects to the list.

g. Then call the '**add\_reimbursement\_details'** method for adding the valid reimbursement request object details to the Oracle database.  This method need not return any object.

g.    In the **main.py file,** get the user input for **request\_id** to search and display the corresponding request details.  First, get the input, and check whether it is a valid request id, If valid,  invoke the appropriate method for searching the **request**using this idand pass the request\_id entered by the user as its parameter.  If found, return that object and display its details as specified in the sample input/output statement shown.  If the request\_id entered is an invalid one, then raise the 'InvalidRequestIdException' and **display** the exception message **'Invalid Request Id'**. If the searching method returns a **None** object, if no matching records are found for a valid id, then it should display the message as **'No record found'.**

h.    Then invoke the appropriate method **'update\_costs**' for displaying the updated request details of employees.  This method should increase the allowances and local travel cost reimbursement amount to 10% if the no. of days of stay is greater than the no. of days of stay entered by the user.  Display the details as specified in the sample input/output statement.

**Sample Input/output Statement 1:**

Enter the Request Id: R003

Request Id: R003  
Employee Code: E003  
Date of Travel: 2020-03-15 00:00:00  
No.of Days: 9.0  
Accommodation Cost: 36000.0  
Dinning Cost: 6300.0  
Local Travel cost: 5600.0  
Allowances: 9000.0  
Total Reimbursement Amount: 56900.0  
  
Enter the no. of days for giving expense increment:5  
  
The updated record details are:  
  
Request Id: R002  
Employee Code: E002  
Date of Travel: 2020-02-15 00:00:00  
No.of Days: 8.0  
Accommodation Cost: 32000.0  
Dinning Cost: 5600.0  
Local Travel cost: 7920.0  
Allowances: 8800.0  
Total Reimbursement Amount: 52800.0  
  
Request Id: R003  
Employee Code: E003  
Date of Travel: 2020-03-15 00:00:00  
No.of Days: 9.0  
Accommodation Cost: 36000.0  
Dinning Cost: 6300.0  
Local Travel cost: 6160.0  
Allowances: 9900.0  
Total Reimbursement Amount: 56900.0  
  
Request Id: R004  
Employee Code: E004  
Date of Travel: 2020-04-15 00:00:00  
No.of Days: 6.0  
Accommodation Cost: 60000.0  
Dinning Cost: 6000.0  
Local Travel cost: 10890.0  
Allowances: 9900.0  
Total Reimbursement Amount: 84900.0  
  
Request Id: R005  
Employee Code: E005  
Date of Travel: 2020-04-15 00:00:00  
No.of Days: 7.0  
Accommodation Cost: 70000.0  
Dinning Cost: 7000.0  
Local Travel cost: 10890.0  
Allowances: 11550.0  
Total Reimbursement Amount: 97400.0  
  
Request Id: R009  
Employee Code: E009  
Date of Travel: 2020-02-11 00:00:00  
No.of Days: 8.0  
Accommodation Cost: 80000.0  
Dinning Cost: 8000.0  
Local Travel cost: 10890.0  
Allowances: 13200.0  
Total Reimbursement Amount: 109900.0

**Sample Input/Output Statement 2:**

Enter the Request Id: R125  
Invalid Request Id  
  
Enter the no. of days for giving expense increment:10  
No Records updated

**Sample Input/output Statement 3:**

Enter the Request Id: R007  
No record found  
  
Enter the no. of days for giving expense increment:8  
  
The updated record details are:

Request Id: R003

Employee Code: E003  
Date of Travel: 2020-03-15 00:00:00  
No.of Days: 9.0  
Accommodation Cost: 36000.0  
Dinning Cost: 6300.0  
Local Travel cost: 6160.0  
Allowances: 9900.0  
Total Reimbursement Amount: 56900.0

 Show Required Files List

 - Grading and Feedback

 >Score : 16.00 / 100.00(Failed)

 >

 >Status of the requirements implemented are as below:

 >

 >Control Structures - 0.00 / 9.00(Not Completed)

 >    \* MEDIUM - Check for the implementation of control statements for calculating the accommodation costs, dining costs, local travel cost, allowances, and total reimbursement amount for level03 and level04 employees  - 0.00 / 2.00

 >    \* MEDIUM - Check for the implementation of control statements for calculating the accommodation cost, dining cost, local travel cost, allowances, and total reimbursement amount for level05 and level06 employees  - 0.00 / 2.00

 >    \* MEDIUM - Check for the implementation of control statements for calculating the accommodation cost, dining cost, local travel cost, allowances, and total reimbursement amount for level01 and level02 employees  - 0.00 / 1.00

 >    \* EASY - Check for the implementation of control statements for checking the length of a request id  - 0.00 / 1.00

 >    \* MEDIUM - Check for the implementation of control statements for checking the first character of the request id  - 0.00 / 1.00

 >    \* MEDIUM - Check for the implementation of control  statements for checking the characters of 1st and 2nd index of the request id  - 0.00 / 1.00

 >    \* DIFFICULT - Check for the implementation of control statements for searching the reimbursement object in the list based on the request\_id specified  - 0.00 / 1.00

 >

 >Presentation - 0.00 / 7.00(Not Completed)

 >    \* DIFFICULT - Check for the console outputs for displaying required employee details if request\_id passed is available in the emp\_reimbursement\_list  - 0.00 / 2.00

 >    \* EASY - Check for the console outputs for displaying required message if request\_id passed is not available in the emp\_reimbursement\_list  - 0.00 / 2.00

 >    \* DIFFICULT - Check for the console output for displaying the updated records of level01 if any  - 0.00 / 3.00

 >

 >Classes and Objects - 6.00 / 15.00(Not Completed)

 >    \* EASY - Check for the implementation of the parameterized constructor with 8 arguments in the 'EmpReimbursement' class  - 3.00 / 3.00

 >    \* EASY - Check for the creation of 'EmpReimbursement' class private attributes  - 3.00 / 3.00

 >    \* MEDIUM - Check for the implementation of the method calculate\_reimbursement\_costs for calculating the reimbursement costs like accommodation cost, dining cost, local travel cost, allowances, and total reimbursement amount and set the same to private attributes  - 0.00 / 3.00

 >    \* DIFFICULT -Check for the implementation of the add\_emp\_reimbursement\_details method for creating valid EmpReimbursement objects as per the requirements  - 0.00 / 6.00

 >

 >Collection Manipulation - 0.00 / 10.00(Not Completed)

 >    \* DIFFICULT - Check for the logic of adding EmpReimbursement objects to a list as per the requirements  - 0.00 / 5.00

 >    \* DIFFICULT - Check for the logic of 'search\_reimbursement\_request' method for retrieving employee object whose request id is as specified and return a corresponding employee object  - 0.00 / 5.00

 >

 >Functions - 2.00 / 12.00(Not Completed)

 >    \* MEDIUM - Check for the logic of the method 'calculate\_reimbursement\_costs' for calculating the  reimbursement costs and set the value to the corresponding variable  - 0.00 / 2.00

 >    \* MEDIUM - Check for the logic of the method 'read\_file' for accessing the specified file and returning the list of record of strings  - 0.00 / 1.00

 >    \* DIFFICULT - Check for the implementation of  'search\_reimbursement\_request' method for retrieving employee object based on request id as per the given business rules and return the employee object  - 0.00 / 2.00

 >    \* EASY - Check for the implementation of  'search\_reimbursement\_request' method for returning none if there is no specified request id in the list  - 0.00 / 1.00

 >    \* DIFFICULT - Check for the implementation of  'update\_costs' method for updating level01 employees reimbursement detailks and return the updated records as list  - 0.00 / 4.00

 >    \* MEDIUM - Check for the implementation of 'convert\_date' method for converting the string date object to date object and return it  - 2.00 / 2.00

 >

 >File Handling - 0.00 / 14.00(Not Completed)

 >    \* MEDIUM - Check for the logic for read the specified file and create the list of records  - 0.00 / 7.00

 >    \* DIFFICULT - Check for the implementation for reading the contents of file to build a list of ‘EmpReimbursement’ objects  - 0.00 / 7.00

 >

 >Database Handling - 0.00 / 20.00(Not Completed)

 >    \* DIFFICULT - Check for the logic for inserting a list of records to the database  - 0.00 / 10.00

 >    \* DIFFICULT - Check for the logic of 'update\_costs' method for updating level01 employee reimbursement costs and return all updated records  - 0.00 / 10.00

 >

 >Exception Handling - 8.00 / 13.00(Partially Completed)

 >    \* MEDIUM - Logic for implementing a constructor in place to raise an invalid customer number exception  - 8.00 / 8.00

 >    \* MEDIUM - Check for the implementation of control statements for checking the last character of the request id  - 0.00 / 5.00