**Important Instruction:**

1. **Please read the document thoroughly before you code.**
2. **Import the given skeleton code into your Eclipse.**
3. **You have to create the input file for the methods.**
4. **Refer/Use the solution file only when you are not able to complete the case study within 3.5 hours**
5. **Business Scenario:**

UNO Bank Private limited offers a variety of investment options to its customers. The company has grown from 10 customers to more than 10000 customers in 1 year. Now, the bank has planned to go digital and has planned to do a onetime conversion of all its data into a digitized mode. For this exercise, all the data currently stored as text files needs to be organized in some predefined formats.

**Develop an application for the below 2 requirements.**

**Requirement 1:**

Provide the details of the customers grouped by the type of account (Savings/NRI/Wealth Mgmt)

**Requirement 2:**

Calculate the maturity amount and store the data in the system.

**Maturity Amount Calculation Logic:**

Maturity Amount = Deposit Amount + (Deposit Amount \* RateOfInterest/100)

The rate of interest should be calculated based on the below table.

Duration Range = (No of days in between start date and maturity date)

|  |  |
| --- | --- |
| **Duration Range** | **Rate Of Interest** |
| 0 – 200 | 6.75 |
| 201 – 400 | 7.5 |
| 401 – 600 | 8.75 |
| >600 | 10 |

**Ex: For**

3112233,Anju,WM,MUT-2222,25000,22-03-2013,30-03-2014

No of days = 373

Maturity Amount = 25000 + (25000\*6.75/100) = 26687.5

1. **Skeleton File for Development:**

Import the below attached skeleton code into your eclipse project and implement the required functionalities



**Technical Specifications:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Sample Input** | **Output Parameters** |
| BankDepositAccountOrganizer | processBankDepositData | **String filePath** – path of the folder along with the filename where the data feed is located. | **"C:\\data\\accountdetails.txt"** | Map<String,List> |
| BankDepositAccountOrganizer | calculateMaturityAmount | **Date** DepositStartDate ,**Date** DepositMaturityDate,**int** DepositAmount | **31-03-2013,** **30-03-2014,25000** | **float** maturity amount |

**Input File:** Create an input file with the name “**accountdetails.txt”** and store the sample customer details records(each line is a record). Use the delimiter “,” to separate the field values.

The record format is given below

**ParentAccountNumber,Name,TypeOfaccount,LinkedDepositAccNumber,DepositAmount,DepositStartDate, DepositMaturityDate**

Field Constraints:

**ParentAccountNumber –** Primary Account Number of the Customer

**Name** – Name of the Account Holder

**TypeOfaccount** should be – SAV/NRI/WM denoting Savings/NRI/Wealth Mgmt respectively.

**LinkedDepositAccNumber –** Account Number of the Linked Deposit Account. I.e. one customer will have multiple deposit accounts which are linked to one parent account number.

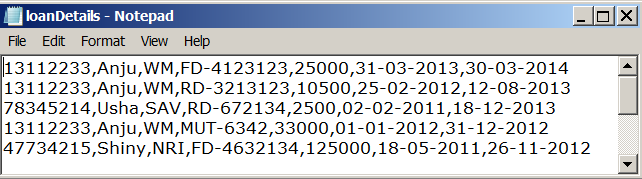
Format of the LinkedDepositAccNumber is **FD-4123123 –** where FD denotes the type of deposit and 4123123 is a unique number. Possible types of deposits are FD/RD/MUT denoting Fixed Deposit/Recurring Deposit/Mutual Funds.

**Deposit Amount –** Amount which is deposited in the Linked Deposit Account

**Deposit Start Date –** The date on which the deposit was started

**Deposit Maturity Date –** The date on which the deposit matures

Sample File:



**Method Logic (to be implemented):**

This will read the text file and based on the type of account (SAV/NRI/WM), it will calculate the maturity amount and return a map in the following format,

Two value objects have been given with the skeleton code for storing the details.

The ParentAccountNumber, Name and the type of account should be stored in the **ParentAccountVO** object and the rest of the details should be stored in the **LinkedDepositVO** object.

Since a single customer will have multiple linked deposits, the ParentAccountVO contains a List of LinkedDepositVO objects.

\* Both the ParentAccountVO and the LinkedDepositVO is provided with the skeleton code, the attribute names, getters / setters and equals method should not be modified should be used as it is.

Validations to be done:

1. All fields are mandatory.
2. The account number should be numeric and should not start with 0
3. Both the dates in the input file should be in dd-MM-yyyy format.
4. The possible values for type of account fields are

* WM
* SAV
* NRI

**Please note that all the above text is in uppercase**

1. TheLinkedDepositAccNumber should start with one of the below options

* FD
* RD
* MUT

**Please note that all the above text is in uppercase**

#### Output Map structure:

|  |  |
| --- | --- |
| Key - String | Value – List<ParentAccountVO> |
| WM | List< ParentAccountVO> having account type as WM |
| SAV | List< ParentAccountVO> having account type as SAV |
| NRI | List< ParentAccountVO> having account type as NRI |

**Sample Output Hashmap:**

|  |  |
| --- | --- |
| Key | Value |
| WM | {{13112233,Anju,WM,FD-4123123,25000, 31-03-2013,30-03-2014,26875.0}, {13112233,Anju,WM,RD-3213123,10500,25-02-2012,12-08-2013,11418.75 },  {13112233,Anju,WM,MUT-6342,33000,01-01-2012,31-12-2012, 35475.0}} |
| SAV | {{78345214,Usha,SAV,RD-672134,2500,02-02-2011,18-12-2013,2750.0}} |
| NRI | {{47734215,Shiny,NRI,FD-4632134,125000,18-05-2011,26-11-2012,135937.5}} |

1. **Solution:**

**The attached code is one of the possible solution meeting the given requirement. Please refer/use this only if you are not able solve the above given problem scenario within 3.5 hours.**

****

**Note: You will not find the solution file in the actual assessment.**