CCJP_TrustInsuranceCompany

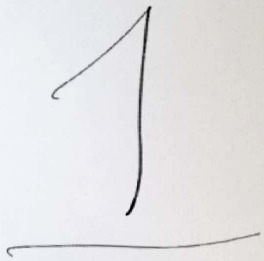
Trust Insurance Company is one of the Life Insurance Company which covers 30% of the New York population. It will give loan for the policy holders based on several criteria. Company will receive a loan request details file every month which contains the details of the policy holder's PAN number, type, requested loan amount, accumulated premium amount etc. It is in need of automatedsoftware which can process the loan request details file and decide whether a policy holder is eligible or not for the requested loan based on the criteria. They have approached you as their software consultant. As an initial requirement they have asked you to develop the following API.

API:This will read loan request details file containing the policy holder's Policy Code, PAN number, policy Start Date, accumulated premium amount, requested Loan Amount, and it will return a map (structure of the map is specified in Output Map Structure section). The API should perform the following functions

1. Identify the List of policy holders who are eligible for the loan.
2. Identify the List of policy holders who are not eligible for the loan.
3. Calculate the policy end date and the net premium amount for all the valid policy holders.
4. Report the List of error records in the given file.
5. Report the List of duplicate requests in the given file.

Output Map structure:

Key Integer	**Value-Map<String, List<PolicyHolderVO>>**	
1	The value should be a HashMap with ELG and NELG as keys and ArrayList of policy holder details who fall under that category. (ELG denotes eligible and NELG denotes not eligible based on the loan eligibility)	
	Key-String	**Value-List<PolicyHolderVO>**
	ELG	
	NELG	
2	Thevalue should be a HashMapcontaining the error codes as key and value should be the array list of PolicyHolderVOobjects with that errors.	



View Problem

Example: If there are no policy holders who are eligible for a loan, then place an empty list as the value for key "ELG".

Skeleton File for Development:

1. You must need to create a new Java Project in by using Package Explorer view in Eclipse IDE, in order to download the skeleton file.
2. The skeleton file to be used for development can be downloaded by clicking on "Download Code Template" button in the Eclipse – Ebox view
3. Save the skeleton file inside your workspace, and import the java file inside the "src" folder of the new java project, created in step 1.

Important instructions:

1. VO and exception classes are provided with the skeleton code. The attribute names, getters / setters and equals method of VO class, should not be modified should be used as it is.
2. Do not modify the skeleton file or, change the visibility of VO / exception classes or add new public methods
3. Do not modify any attribute names, method prototypes, return types provided as part of the skeleton file
4. Do not hard code the return values of business methods, if found your code will not be considered for processing.

Technical Specifications:

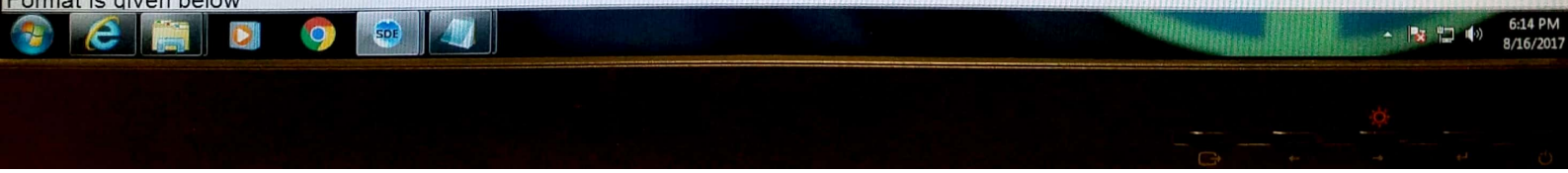
ClassName	Method Name	Input Parameters	Sample Input	Output Parameters*
TrustLoanSanctioner	loanProcessor	**String filePath** – path of the folder along with the filename where the data feed is located. The path will be dynamic and hence passed as input to this method. **String sanctionDate**-date is passed as a parameter on which the loan is sanctioned to policy holder (In dd/MM/yyyy format)	oanRequestDetails.txt	Map<Integer, Map>

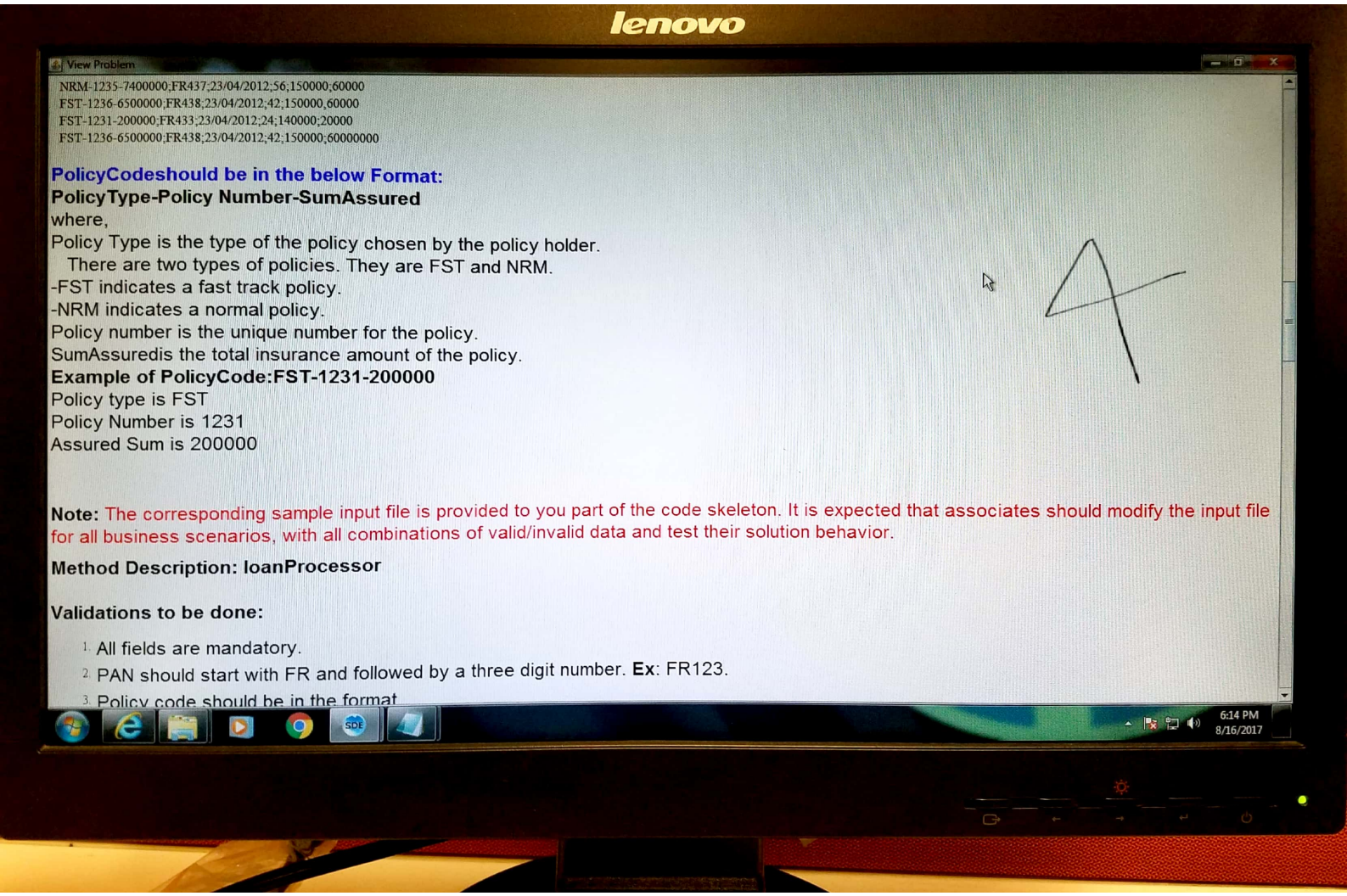
*where the inner map is a Map<String,List<PolicyHolderVO>>

Sample File:

Note: Associates to create the files with the required data for testing (based on the format provided in the screenshot below)

Format is given below





View Problem

TrustLoanSanctioner	loanProcessor	String filePath – path of the folder along with the filename where the data feed is located. The path will be dynamic and hence passed as input to this method. String sanctionDate-date is passed as a parameter on which the loan is sanctioned to policy holder (In dd/MM/yyyy format)	oanRequestDetails.txt	Map<Integer, Map>

***where the inner map is a Map<String,List<PolicyHolderVO>>**

Sample File:

Note: Associates to create the files with the required data for testing (based on the format provided in the screenshot below)

Format is given below

Policy Code;Pan;policy StartDate;Period;Accumulated Premium Amount;Requested Loan Amount;

FST-1231-200000;FR433;23/04/2012;24;140000;20000
NRM-1232-350000;FR434;16/05/2012;48;140000;10000
FST-1233-250000;FR435;06/03/2012;36;49000;30000
FST-1234-350000;FR436;18/04/2012;72;650000;40000
NRM-1235-7400000;FR437;23/04/2012;56;150000;60000
FST-1236-6500000;FR438;23/04/2012;42;150000;60000
FST-1231-200000;FR433;23/04/2012;24;140000;20000
FST-1236-6500000;FR438;23/04/2012;42;150000;60000000

PolicyCodeshould be in the below Format:

PolicyType-Policy Number-SumAssured

where,

Policy Type is the type of the policy chosen by the policy holder.

There are two types of policies. They are FST and NRM.

-FST indicates a fast track policy.

-NRM indicates a normal policy.

View Problem

7. Policy Start date should be in the format, dd/MM/yyyy
8. Accumulated Premium amount should not be greater than the sum assured.

If any of the above validations fail, the system should throw a **"TrustLoanException".**

The feed needs to be parsed and the output should be a Map<Integer, Map> containing the data in the structure explained in Section ***"Output Map Structure"***

Each of the keys is explained in the below sections:

For Key 1:

The value will be a Map <String, List<PolicyHolderVO>>. The inner map should contain the ELGas key for policy holders who are eligible for loan and value will be ArrayList of PolicyHolderVO objects.

NELG should be the key for the policy holders who are not eligible for loan and the value to this key will be ArrayList of PolicyHolderVO objects.

Policy end date should be calculated and set in PolicyHolderVO along with other details

Policy End date = Policy Start Date + period (in number of months)

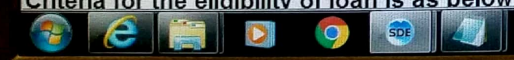
Example:

Policy holder Details: NRM-1232-350000; FR434; 30/12/2012; 48; 140000; 10000

Policy Start date is 30/12/2012 and Period is 48 months

Policy End date = 30/12/2016

Criteria for the eligibility of loan is as below



View Problem

Note: The corresponding sample input file is provided to you part of the code skeleton. It is expected that associates should modify the input file for all business scenarios, with all combinations of valid/invalid data and test their solution behavior.

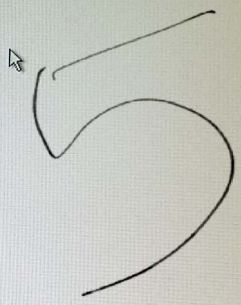
Method Description: loanProcessor

Validations to be done:

1. All fields are mandatory.
2. PAN should start with FR and followed by a three digit number. **Ex**: FR123.
3. Policy code should be in the format

   PolicyType-Policy Number-Sum Assured

4. Policy type must be either FST or NRM. **Ex: FST**-1231-200000-12.
5. Policy number should be a non-zero number
6. Sum Assured should be a number
7. Policy Start date should be in the format, dd/MM/yyyy
8. Accumulated Premium amount should not be greater than the sum assured.



If any of the above validations fail, the system should throw a**"TrustLoanException".**

The feed needs to be parsed and the output should be a Map<Integer, Map> containing the data in the structure explained in Section ***"Output Map Structure"***

Each of the keys is explained in the below sections:

For Key 1:

6:14 PM
8/16/2017

View Problem

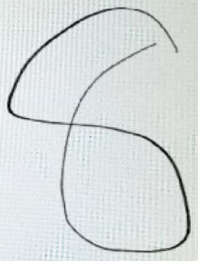
FST	>60% of Accumulated premium Amount and >70% of Assured Sum	NELG

***Eligibility status (NELG/ELG) must be in upper case letters**

If eligibility is ELG you need to calculate the net premium based on the below formula and set in the PolicyHolderVO along with other details.
If eligibility is NELG, then net premium amount = accumulated premium amount from file.
Steps for calculating the net premium amount (for those who are eligible for loan):

Net Premium Amount= Calculated Premium Amount + Increase in Premium

Calculated Premium Amount = Sum Assured/ Period
Increase in premium = (loan amount+interest) /loan period where,
loan amount = requested loan amount from file
interest = loan *1.2*loan period/100
loan period = no of months between policy end date and loan sanctioned date
Refer to section For Key1 for policy end date calculation.
loan sanctioned date is passed as input parameter to this method

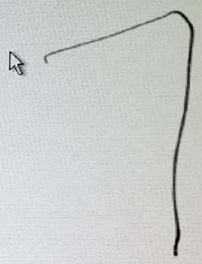


Example:
Policy holder Details:NRM-1232-350000; FR434; 30/12/2012; 48; 140000; 10000
SanctionDate: 30/12/2013(Provided as input to the method)
Policy type = NRM
Requested loan amount =10000
Accumulated premium amount =140000
To check for Loan eligibility:
40% of accumulated premium amount =56000
As requested loan amount is less than 40% of accumulated premium amount eligibility is ELG

6:14 PM
8/16/2017

Criteria for the eligibility of loan is as below

Policy Type	Requested Loan Amount	Eligibility Status
NRM	>40% of accumulated premium Amount	NELG
NRM	<40% of accumulated premium Amount	ELG
FST	<60% of Accumulated premium amount	ELG
FST	>60% of Accumulated premium Amount and <70% of Assured Sum	ELG
FST	>60% of Accumulated premium Amount and >70% of Assured Sum	NELG



*Eligibility status (NELG/ELG) must be in upper case letters

If eligibility is ELG you need to calculate the net premium based on the below formula and set in the PolicyHolderVO along with other details.
If eligibility is NELG, then net premium amount = accumulated premium amount from file.

Steps for calculating the net premium amount (for those who are eligible for loan):

Net Premium Amount= Calculated Premium Amount + Increase in Premium

Calculated Premium Amount = Sum Assured/ Period
Increase in premium = (loan amount+interest) /loan period where,

View Problem

=7291.67+397.77=7689.44(This value needs to be set as premium amount in the PolicyHolderVO object)

*For non-eligible applicants the net premium amount will be the same as the accumulated premium amount obtained from the file

For Key 2:

Thevalue to this key (Key2)will be a Map<String, List<PolicyHolderVO>>.

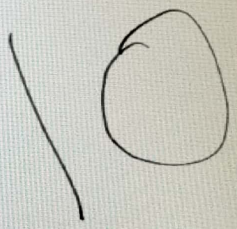
The inner map should contain the error codes (DUP and INV) as key and value should be the array list of PolicyHolderVO objects with the appropriate errors.

DUP key:Policy number is unique number given for each policy. If there are any duplicate records (records having the same policy number) then they must be put in this map with key as DUP and the ArrayList of duplicate records as value.

INV Key: If the requested loan amount is greater than the SumAssured, then they must be put in this map with key as **INV** and the ArrayList of invalid records as value.

Sample Output Map:

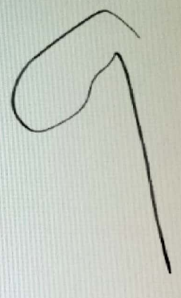
Key-Integer	**Value-Map<String, List<PolicyHolderVO>>**	
1	**Eligibility**	**Arraylist of PolicyHolderVOobjects**
	ELG	Arraylist of PolicyVO objects who are eligible to take the loan
	NELG	Arraylist of PolicyVO objects who are not eligible to take the loan
2	**PAN number**	**Arraylist of PolicyHolderVO objects**
	DUP	Arraylist of PolicyVO objects which are duplicate (All the duplicate records should be present here)
	INV	Arraylist of PolicyVO objects which are invalid (i.e. requested loan amount is greater than the Sum Assured)



6:14 PM
8/16/2017

View Problem

Example:
Policy holder Details:NRM-1232-350000; FR434; 30/12/2012; 48; 140000; 10000
SanctionDate: 30/12/2013(Provided as input to the method)
Policy type = NRM
Requested loan amount =10000
Accumulated premium amount =140000
To check for Loan eligibility:
40% of accumulated premium amount =56000
As requested loan amount is less than 40% of accumulated premium amount eligibility is ELG



Calculating Net Premium Amount:
Calculated Premium Amount =350000/48=7291.67
Policy end date =30/12/2012 + 48 months = 30/12/2016
Loan period =no of months between 30/12/2016 (policy end date) and 30/12/2013(sanctioned date) = 36
Interest=loan*1.2*loan period/100
Interest=10000*1.2*36/100=4320
Increase in premium = (loan amount+interest) /loan period
Increase in premium = (10000+4320)/36=397.77
Hence,
Net premium amount = Premium Amount + Increase in premium
=7291.67+397.77=7689.44(This value needs to be set as premium amount in the PolicyHolderVO object)
*For non-eligible applicants the net premium amount will be the same as the accumulated premium amount obtained from the file

For Key 2:
Thevalue to this key (Key2)will be a Map<String, List<PolicyHolderVO>>.
The inner map should contain the error codes (DUP and INV) as key and value should be the array list of PolicyHolderVO objects with the appropriate errors.

6:14 PM
8/16/2017