Loan Metric Calculation

You are developing a Spring REST Application and your team decides to use a set of service classes to process loan requests and perform a variety of calculations.

Requirements

1- Complete the rest services in LoanController using the right annotations, rest naming conventions and input validations if applicable.

**public** Loan **getLoan**(Long loanId)

**public** LoanMetric **calculateLoanMetric**(Long loanId)

**public** Loan **getMaxMonthlyPaymentLoan**()

2- Complete the implementation of ILoanMetricCalculator in

ConsumerLoanMetricCalculator and StudentLoanMetricCalculator and implement the LoanMetricFactory @Service class to create the right instance of ILoanMetricCalculator based on the Loan type (student or consumer).

Additionally complete the isSupported method in ILoanMetricCalculator to validate if the loan type is supported.

For StudentLoanMetricCalculator, the method isSupported should also validate the borrower's age is greater than 18 and lower than 30

3- Calculate the metric of the loan with the following business rules: The formula to calculate the monthly payment amount is as follows:

Students:

**monthlyInterestRate** = (annualInterest /12 ) / 100

monthlyPayment = 0.8 \* (requestedAmount \* monthlyInterestRate) / (1 - (1 + monthlyInterestRate)^((-1) \* termMonths) )

Consumers:

**monthlyInterestRate** = (annualInterest /12 ) / 100

monthlyPayment = (requestedAmount \* monthlyInterestRate) / (1 - (1 + monthlyInterestRate)^((-1) \* termMonths) )

4- Get the max monthly payment loan.

Metric Example

If a borrower is asking for a Consumer loan for $10,000 in 2 years:

Borrower JSON

{

loanId: 1,

requestedAmount: 10000,

termYears: 2,

termMonths: 0,

annualInterest: 6,

type: "consumer",

loanOfficerId: 100,

borrower: {

name: "James Gosling",

annualIncome: "100000",

delinquentDebt: false,

annualDebt: 3000,

creditHistory: 6

}

}

The result of the metrics will be:

Loan type = consumer

Monthly Rate = 0.005

Monthly Payment = (10000\*0.005)/(1-(1+0.005)^-24) = 443.20

Important Notes

**NOTE 1:** Implementations of IloanMetricCalculator must be injected. Complexity O(1) to get the instance is a plus.

**NOTE 2:** Feel free to edit the mock generated in LoanGeneratonUtil.createLoan(Long loanId). **NOTE 3:** The Creation of IloanMetricCalculatorinstances using “new” will be considered an error causing the rejection of the whole exercise.

**NOTE 4:** Any comment in the code for understanding what is being done is a plus **NOTE 5:** Adding unit tests (at least on important methods) is a plus

**NOTE 6:** Use of collection streams also add value to the solution