**LendPro Loan Management System**

**Important Instructions:**

* Please read the document thoroughly before you code.
* Import the given skeleton code into your Eclipse.
* Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
* You can create any number of private methods inside the given class.
* You can test your code from **main()** method of the program
* Using Spring Core develop the application using **xml** configuration. Object creation and

Initialization of variables should be done through constructor injection only.

**Assessment Coverage:**

* **Classes, Objects and Members, Construction Injection**
* **Inheritance, Collection, Property Configuration**

Purpose of this exercise is to simulate a loan process that provides below functionality:

Calculate the Equated Monthly Installment (EMI) based on the provided information and the configured interest rates for different loan types.

**Technical Requirements:**

You are required to do the exercise following below conditions.

|  |
| --- |
| **<<Abstract>>**  +***Loan*** |
| - customerId :int  - customerName:String |
| **<<constructor>>**  + Loan(int,String) |
| **<<methods>>**  + calculateEMI (double,int,String):double |

**<<Extends>>**

|  |
| --- |
| + **SmartLoan** |
| - interestRatesMap : Map<String, Double> |
| **<<constructor>>**  + SmartLoan(int,String, Map<String, Double>) |
| **<<methods>>**  + calculateEMI (double,int,String):double |

An **abstract** class **Loan** with below mentioned private member variables, constructor and public methods are provided as part of the code skeleton:

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| customerId | int |
| name | String |

Create a class **SmartLoan** that extends the class Loan with below mentioned private member variables and public methods:

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| interestRatesMap | Map<String,Double> |

Define a **public parameterized constructor with all the above variables in the same order of parameters**, along with getter and setter methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Specifier/Modifier** | **Method**  **Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| public | **calculateEMI** | double loanAmount,  int tenure,  String loanType | double | This method accepts loan amount, tenure and loan type as parameters and calculates EMI and returns the same. |

**Business Rules:**

|  |  |
| --- | --- |
| **Methods** | **Business Condition** |
| **calculateEMI** | Loan amount should be greater than 0 and tenure should be greater than 0 months and loan type should be available in the properties file. Return value should be format to 2 decimal places.  Hint : Use DecimalFormat API |

**Loan** class should be registered as a **bean** as ‘**abstract= true**’ with the spring container via **XML file**.

**Create** class **SmartLoan** which **extends** Loan and give implementation for abstract method calculateEMI. Use below formula to calculate emi.

**SmartLoan** class should be registered as a **bean** with the spring container via **XML file** with **bean id** as **smartLoan**.

The values for all the attributes should **be injected via constructor-based injection**, the default **customerId** should be **12345**, **customerName** should be **‘John’**, and properties should be fetched from the properties file called **accounts.properties** usingthe property configuration concept by creating a bean of PropertyPlaceholderConfigurer in spring container via XML file.

**loanTypes.properties**

|  |  |
| --- | --- |
| **Key** | **Value** |
| personalLoan | 0.085 |
| homeLoan | 0.075 |
| carLoan | 0.09 |

**Note**: Key values are case sensitive.

**EMI Calculation**:

EMI = (P \* r \* (1 + r)^n) / ((1 + r)^n - 1)

P=loan amount, r=interest rate, n=number of months based on loan type

e.g: p=10000, n=12, loan type = carLoan

EMI = (10000 \* 0.09 \* (1 + 0.09)^12) / ((1 + 0.09)^12 - 1)

EMI = 874.51

**General Design Constraints:**

* Ensure that all the Java Coding Standards are followed.
* Assume that the method inputs are valid always, hence exceptional blocks are not needed to be included in the development.

**Sample Input Output 1:**

Welcome to Loan Processing System

Customer Name: John

Customer ID: 12345

Enter loan amount

**90000**

Enter loan tenure in months

**9**

Enter loan type

**homeLoan**

Your EMI for 9 months will be $10315.1

**Sample Input Output 2:**

Welcome to Loan Processing System

Customer Name: John

Customer ID: 12345

Enter loan amount

**525000**

Enter loan tenure in months

**15**

Enter loan type

**carLoan**

Your EMI for 15 months will be $37136.61

**Sample Input Output 3:**

Welcome to Loan Processing System

Customer Name: John

Customer ID: 12345

Enter loan amount

**450000**

Enter loan tenure in months

**21**

Enter loan type

**propertyLoan**

Invalid Input