**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 banking process which provides below functionalities:

1. Create **Current Account** and do deposit and withdraw functions.
2. Create **Fixed Deposit Account** and calculate the maturity amount based on deposit and tenure
3. Create **Recurring Deposit Account** and calculate the maturity amount based on deposit and tenure.

**Skeleton File for Development:**

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



**Technical Requirements:**

You are required to do the exercise following below conditions.

|  |
| --- |
| **<<Abstract>>**  +***BankAccount*** |
| -customerId :int  -name:String  -balance:double  -accounts:List<String> |
| <<constructor>> +BankAccount(int,String,double,List)  +doDeposit(double):double  +doWithdraw(double):double  +*calculateFixedAccount(double,int):double*  *+calculateRecurringAccount(double,int):double* |

**<<Extends>>**

|  |
| --- |
| + **SmartBankAccount** |
| +calculateFixedAccount(double,int):double  +calculateRecurringAccount(double,int):double |

**Step 1:** Create an **abstract** class **BankAccount** with below mentioned private member variables and public methods:

|  |  |
| --- | --- |
| customerId | int |
| name | String |
| balance | double |
| accounts | List<String> |

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 | doDeposit | double amount | double | This method accepts amount as parameter and adds amount to balance and returns balance |
| public | doWithdraw | double amount | double | This method accepts amount as parameter and deducts amount from balance and returns balance |
| Public abstract | calculateFixedAccount | double amount, int months | double | This method takes amount and months as parameters and calculates fixed maturity amount and returns it |
| Public abstract | calculateRecurringAccount | double amount, int months | double | This method takes amount and months as parameters and calculates recurring maturity amount and return it. |

**Business Rules:**

|  |  |
| --- | --- |
| **Methods** | **Business Condition** |
| calculateFixedAccount | Amount should be greater than 9999 and less than 1000001 and tenure should be less than 121 months and greater than 0. Return should be format to 2 decimal places.  Hint :Use DecimalFormat API |
| calculateRecurringAccount | Amount should be greater than 999 and less than 50001 and tenure less than 61 months and greater than 0. Return should be format to 2 decimal places.  Hint :Use DecimalFormat API |

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

**Step 2: Create** class **SmartBankAccount** which **extends** BankAccount and give implementation for abstract methods calculateFixedAccount and calculateRecurringAccount. Use below formulas to calculate fixed deposit and recurring deposit .For fixed deposit interest is fixed at 10% and for recurring deposit interest is fixed at 7%.

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

The values for all the attributes should **be injected via constructor based injection** , the default **custId** should be **100**, **name** should be **‘Joe’**, **balance** should be **10000**, **accounts** should be a list containing values **“CurrentAccount”, “RecurringAccount”, “FixedAccount”.** Values of list should be fetched from properties file called **accounts.properties** usingproperty configuration concept by creating a bean of PropertyPlaceholderConfigurer in spring container via XML file.

**accounts.properties**

|  |  |
| --- | --- |
| **Key** | **Value** |
| Current | CurrentAccount |
| Recurring | RecurringAccount |
| Fixed | FixedAccount |

**Fixed Amount** : A=P(1 + r/n)nt

A=final amount, P=initial principal balance, r=interest rate, n=number of times interest applied per time period, t=number of time periods elapsed

e.g: p=10000, r=10%=0.1, n=12, t=60

result = 10000\*( ( 1+ ( o.10 ) /12 ) 12\*(60/12) )

A=16453.09

**Recurring Amount**: (P\*n)+(P(n(n+1)/(2\*12))\*(r/100))

P=initial principal balance, r=interest rate, n=number of times interest applied per time period

e.g: p=10000, r=7%, n=12

result = (10000\*12) + (10000(12(12+1)/24)\*(0.07))=124550.00

**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 Mr/Ms: Joe

Your customer id is :100

Please select account type

1 CurrentAccount

2 RecurringAccount

3 FixedAccount

**1**

Select transaction type

1.Deposit

2.Withdraw

**1**

Enter amount to be deposited

**10000**

Your balance amount is 20000.00

**Sample Input Output 2:**

Welcome Mr/Ms: Joe

Your customer id is :100

Please select account type

1 CurrentAccount

2 RecurringAccount

3 FixedAccount

**1**

Select transaction type

1.Deposit

2.Withdraw

**2**

Enter amount to be withdrawn

**5000**

Your balance amount is 5000.00

**Sample Input Output 3:**

Welcome Mr/Ms: Joe

Your customer id is :100

Please select account type

1 CurrentAccount

2 RecurringAccount

3 FixedAccount

**2**

Enter monthly installment

**15000**

Enter tenure in months

**10**

Your balance after 10 months will be 154812.50

**Sample Input Output 4:**

Welcome Mr/Ms: Joe

Your customer id is :100

Please select account type

1 CurrentAccount

2 RecurringAccount

3 FixedAccount

**3**

Enter FD amount

**100000**

Enter tenure in months

**36**

Your balance after 36 months will be 134818.18