**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 , **You don’t have to do any change**

**in main method.**

* Using Spring Core AOP develop the application using **Java Configuration**. Object creation and

Initialization of variables, autowiring should be done through **Java Configuration** , only annotations to be used .

**Time: 1 hour**

**Assessment Coverage:**

* **Annotations ,Before Advice , Around Advice**
* **AfterThrowing Advice, AfterReturning Advice**

Application created should be a demo of how to simulate an Atm transaction. For the demo sake we are creating a menu driven application which has one bank EasyBank and assume there is only one customer. Customer can do deposit , withdraw, change pin, check balance . Before all the transactions customer will be prompted to enter pin number . If pin number is correct he will be able to do the transaction else will be shown error message.

**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 develop an App following below conditions.

**Step 1:** Create a class **EasyBank** with below mentioned private member variables and public methods and annotate the class with **@Component**:

**Variables:**

**pinCode** of type **int** , **balance** of type **int, tempPin** of type **int**

Give appropriate **getters** and **setters**.

**Methods:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| EasyBank | doWithdraw | Int amount | void | This method accepts amount as parameter and deducts amount from balance and prints “You have successfully withdrawn <<amount>>”.If the withdraw amount is greater than balance will print “Insufficient Fund” |
| EasyBank | doDeposit | Int amount | void | This method accepts amount as parameter and add amount to the balance and prints “Your balance is <<balance>> |
| EasyBank | doChangePin | Int oldPin , int pin | void | This method should accept old pin and new pin from user . if old pin and existing pin is same , new pin is replaced with existing pin else throw exception. |
| EasyBank | showBalance | No-args | void | This method prints “ Your balance is <<balance>> |

**Note:**

1. pinCode is set 6789 and balance is set to 10000 as default . tempPin will be read from the user dynamically and cross checked with pinCode for validating pin.
2. New pin must be of 4 digit , first digit must not be a 0.

**Step 2:** Create class **LoginAspect**  and annotate with **@Component** and **@Aspect** and below mentioned member variable and public methods.

**Variable:**

**easyBank** of type **EasyBank**.

Annotate the variable with **@Autowired** annotation

**Methods:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Annotation** | **Method Name** | **Input Parameters** | **Output Parameter** | **Logic** |
| @Around | validateWithdraw  throws Throwable | ProceedingJoinPoint joinPoint | void | This method should verify tempPin with pinCode of easyBank object , if both are same it should print “Your remaining balance is <<balance>>” after the execution of withdraw method else should throw Exception |
| @Before | validateBalance  throws Exception | No-args | void | This method should verify tempPin with pinCode of easyBank object , if there is pinCode mismatch it should throw Exception |
| @AfterReturning | afterPinChange | No-args | void | This method should only print “You have successfully changed your pin” |
| @AfterThrowing | afterWrongPin | No-args | void | This method should only print “Invalid Pin” |

**Business Rules:**

|  |  |
| --- | --- |
| **Methods** | **Business Condition** |
| validateWithdraw | This method or aspect should be applied around doWithdraw method means this method or aspect will be applied before and after of execution of doWithdraw method . Before doWithdraw this method should check pincode . If pinCode is correct doWithdraw method will be called else it will throw Exception and doWithdraw method wont be called . After doWithdraw this method will print the remaining balance available. |
| validateBalance | This method or aspect should be applied before showBalance and doDeposit method , means this method or aspect will be applied before execution of showBalance and doDeposit method. This method should check whether pinCode is correct or not. If pinCode is wrong this method will throw Exception .If pinCode is correct showBalance method or deDeposit method will be executed and remaining balance will be printed. |
| afterPinChange | This method or aspect will be executed on successfull execution of doChangePin |
| afterWrongPin | This method or aspect will be executed if any method or aspect throws Exception for invalid pin |

**Step 3:** Create class **AopConfig**  and annotate it with **@Configuration** , **@EnableAspectJAutoProxy** and **@ComponentScan**

**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:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

1

Enter amount to deposit

5000

Enter pin

6789

Your balance is 15000

**Sample Input Output 2:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

1

Enter amount to deposit

5000

Enter pin

222

Invalid Pin

**Sample Input Output 3:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

2

Enter amount to withdraw

3000

Enter pin

6789

You have successfully withdrawn 3000

Your remaining balance is 7000

**Sample Input Output 4:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

2

Enter amount to withdraw

15000

Enter pin

6789

Insufficient Fund

Your remaining balance is 10000

**Sample Input Output 5:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

2

Enter amount to withdraw

3000

Enter pin

22

Invalid Pin

**Sample Input Output 6:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

3

Enter your current pin

6789

Enter 4 digit new pin

1234

You have successfully changed your pin

**Sample Input Output 7:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

3

Enter your current pin

2225

Enter 4 digit new pin

1234

Invalid Pin

**Sample Input Output 8:**

Select option

1.Deposit

2.Withdraw

3.Change Pin

4.Show Balance

5.Exit

3

Enter your current pin

6789

Enter 4 digit new pin

22

Invalid Pin