ASSESSED EXERCISE 4: Loan Repayment

Deadline: To be submitted on KLE by Friday 16/December/2022, 13:00.
Weighting: 25% of the entire module mark.

Module Learning Outcomes Assessed
1. Show practical experience of the basic concepts of computer programming.
2. Evaluate the suitability of computer language data and control structures to achieve basic problem-solving.
3. Use basic software engineering principles to design and implement computer programs.

In this practical you will be implementing some of the object oriented concepts that you have learned in this module.

Task 1

You need to start by creating three account classes called `SavingsAccount`, `CurrentAccount` and `LoanAccount` with the following attributes and behaviour:

Class	Attributes	Behaviour

`SavingsAccount`	`interestRate`
	`balance`
		`getBalance`
		`addInterest`
		`makeDeposit`

`CurrentAccount`	`interestRate`
	`balance`
		`getBalance`
		`addInterest`
		`makeDeposit`

`LoanAccount`	`interestRate`
	`balance`
		`getBalance`
		`addInterest`
		`makePayment`

The names of the attributes and behaviour should make obvious to you what their role/function should be. Here is an explanation for each:

`balance`
The balance of each account in pounds. This should be a floating point number in order to accommodate pence amounts. This attribute should be `private` to the class.

`interestRate`
The **monthly** interest rate of each account. This should be a class variable (i.e. not an instance variable) and also `private` to the class. In other words, any change to this

attribute should reflect on all the instances of the class it belongs to. This attribute should be initialised for each class as follows:

The `interestRate` for `SavingsAccount` should be initialised to 0.008 (i.e. 0.8%).
The `interestRate` for `CurrentAccount` should be initialised to 0.0006 (i.e. 0.06%).
The `interestRate` for `LoanAccount` should be initialised to 0.012 (i.e. 1.2%).

`getBalance`
This method should take no arguments and return the balance. It is required because balance will not be accessible directly.

`addInterest`
This method should compute and add the interest to the account using its interest rate. It takes no arguments and returns nothing.

`makeDeposit`
This method should take the amount to be deposited to the account as an argument and add that to the balance.

`makePayment`
This method should take the amount to be paid into the loan account and subtract that from the balance.

Each of the three classes should also have **constructor methods**. You are expected to use two constructors (see **polymorphism/overloading**) for each of the classes `SavingsAccount` and `CurrentAccount`: one that takes no arguments and initialises the balance to 0.0 and another that takes and uses a starting balance for the account to be created.

There should only be one constructor method for the `LoanAccount` class that should take and use a starting balance (i.e. the amount borrowed when the account is created).

You are expected to write your own code to test that the above classes work as expected. This test code will not form part of your assessment.

Task 2

Create a class called `Application` that will use the above three classes. In this class you should create a `main` method that will simulate two people (John and Mary) who each borrow money from the bank in terms of a loan. Your program needs to work out how long (in months) it will take for each of them to repay their loan and how much money they will have saved by that time.

The following information is provided for you:

John borrowed £22000. His monthly income is £2600, his monthly outgoings (utility bills, telephone bills, food expenses etc) are £1300. John decided to pay £500 towards his loan every month and save what remains from his income in a **current account** that he opened (with no starting balance) at the same time he got his loan.

Mary also borrowed £18000. Her monthly income is also £2800, her monthly outgoings are £1400. Mary decided to pay £350 towards her loan every month and save the rest of her income in a **savings account** that she opened (with £200 starting balance) at the same time she got her loan.

John and Mary both get their income on the first day of the month. On the same day they make their loan payment, set aside their monthly outgoing budget for the month ahead and they deposit what remains into their respective accounts (savings and current).

At the end of the month the bank attributes interest to all its accounts.

Recall that you can calculate the interest to a balance using:

```
interest = interestRate * balance
```

Your program should use the classes created in task 1 appropriately. You should use iteration to simulate what happens every month from the time John and Mary obtained their loans to the time they paid them off.

Your program should produce only two output lines like the ones below:

```
John repaid his loan after ? months. His current account balance at that time was £?
Mary repaid her loan after ? months. Her savings account balance at that time was £?
```

where in your case of course there will be a number instead of a "?".

You are also expected to write your program in such a way that it can allow you to easily change any figures mentioned above. For example, if you wanted to also forecast how long it would take Mary to repay her dept if she decided to pay £300 (instead of £225) monthly into her loan you should only need to make a single change in your program to reflect that.

Please note that you need not bother with formatting your floating point numbers to appear with two decimal figures. For example, it would be ok if an amount appears as `£28.43389271206` instead of `£28.43`.

Marking scheme (total 100%)

Description	Mark
Implementation of the requested class attributes for the three account classes.	20%
Implementation of the requested class behaviour (constructors and other methods) of the three account classes.	40%
Implementation of the main application using the three account classes.	40%

Make sure that you submit your completed project (as a zip/rar file of the entire Apache NetBeans project folder) for assessment. Ask for help if you are unsure about how to compress your project folder.

What resources might I use to get started?
Ans: Laboratory CSL3.104 and CR010, CR012 and CR014 PCs and software.
Contact: Bappaditya Mandal (Email: b.mandal@keele.ac.uk)
Office: Colin Reeves CR36, Hours: Ask during any of the practical classes in CSL3.104 or by appointment via email.