ECS414U/A - Object Oriented Programming
Week 3 lab session (5 marks)

Lecturer: Bruno Ordozgoiti

8 February 2022

Rules & instructions: This assignment is individual work. You must complete it by yourself. You can discuss it with your demonstrator and other students in your lab session, and you are in fact encouraged to do so. However, you must not share your solution with other students, so that they can submit it as their own, or request that others share their solution with you. This would result in disciplinary action.

You are given a program consisting of the following files: `BankingApp.java, Account.java, User.java, Agent.java, Client.java, CorporateClient.java` and `PremiumClient.java`, as well as some tests. The program is incomplete and does not compile yet. You must modify the source code to satisfy the list of requirements below. When you compile and run the program, a series of tests will be run (you will need to uncomment some of the test invokations, as specified in `BankingApp.java`). When you think that your code is ready, you must submit the following files to your demonstrator for them to run the tests:

- `Agent.java`
- `Client.java`
- `PremiumClient.java`
- `CorporateClient.java`

You will get marks for each passed test (a test is passed if and only if it outputs `Test OK`). Your demonstrator and the module staff may ask you questions about your code, if something is unclear. In that case, only once the answers to these questions are clear, you will get marks for the corresponding test. If any issues are encountered, you can keep working on the code and try again for the duration of the lab session. After the session ends, you will have additional time to submit your code to QM+.

Important! Requirements to get the marks:

- **It is mandatory to attend the lab session in order to get the marks.** You must report to your demonstrator on MS Teams (even if you attend in person).

- You must submit your files (see above) to QM+ before 23:59 Thursday, 17 Feb. This applies to everyone, even those of you who completed the assignment and showed it to your demonstrator in the lab. The activity to submit your code is *Week 3 lab submission*. If you didn't have time to get your code evaluated by your demonstrator during the lab, it will be assessed after the submission deadline.

Advice: Try to follow good coding practices and the principles of OOP mentioned in the lectures. Even though a passed test may get you the marks, there are many ways to write code, and some are arguably better than others. Try to discuss your approach with your demonstrator and fellow students, and try to understand whether there is something you could have done better. When you get the solutions, compare them to your code. The more effort you put into writing good code, and the better you understand the concepts involved in these exercises, the better your chances will be of doing well in the assessed tests and mini-project.

Note: This time you will have to be more careful writing your code. **Some approaches are not allowed**. You will have to make use of the notions of inheritance and polymorphism seen in the lectures. Forbidden approaches are specified in each requirement. Recall that we distinguish between **fields, methods** and **constructors.**

Inspect the provided code (except the tests) and make sure you understand its contents as you do the exercise. In particular, note that the class `User` implements functionalities common to all users in the system, including clients and staff.

The **key to resolving this exercise** is to figure out the adequate inheritance hierarchy. Thus, you are always allowed to modify this.

List of requirements:

R1 (1 mark) Add a constructor to the class `Client`, taking one parameter corresponding to its name. The constructor must also initialize its `Account` field to a default `Account` object. **You are not allowed to define any new fields or methods in** `Client`. Remember you can always modify the inheritance hierarchy.

There are two special types of client, corporate and premium, implemented by the classes `PremiumClient` and `CorporateClient`.

R2 (1 mark) **Corporate clients** behave just like regular clients, but they have an additional integer attribute representing their Company Registration Number (CRN). You must implement a constructor to make sure corporate clients can be instantiated, taking their name and CRN as parameters (in that order). **You are not allowed to define any new fields or methods in** `CorporateClient`.

R3 (1 mark) **Premium clients** are like regular clients, but they earn a bonus every time they make a deposit. You must implement a constructor to make sure premium clients can be instantiated, taking their name as a parameter, and setting their accumulated bonus (the field `bonus`) to 0. You must also implement a method `deposit` that in addition to making a deposit for the given amount, increases the accumulated bonus by 0.1% of the deposited amount. **You are not allowed to define any new fields in** `PremiumClient`.

R4 (2 marks) Agents are members of staff in charge of dealing with clients. Each agent is assigned a maximum of 5 clients of any kind. The class `Agent`, therefore, must hold a list of the clients assigned to the corresponding agent.

`Agent` must implement the following methods:

- `addClient`, to assign a new client to an agent;
- `getClients`, to retrieve the list of clients assigned to an agent.

You are only allowed to add these two methods, without overloading. You are only allowed to add one extra field to `Agent`, on top of the two provided already. You must modify the constructor to make sure that all fields are initialized properly.