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

Lecturer: Bruno Ordozgoiti

22 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, 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`, `Client.java`, `Agent.java`, `User.java`, `Prompt.java` and `WindowCloser.java`. The program provides a user interface for a bank agent, in charge of a number of clients.

You must modify the source code to satisfy the list of requirements below. This time, your demonstrator will check that the requirements are satisfied by running your program and testing the GUI. When you think that your code is ready, you must submit the file `BankingApp.java` to your demonstrator for them to run the tests. This time you are free to write additional source files from scratch. However, you may not modify any other of the provided files —only `BankingApp.java`. If you write additional source files, you must of course send them to your demonstrator as well.

You will get marks for each satisfied requirement. 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 requirement. 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 (`BankingApp.java` and any other source files you have created) to QM+ before 23:59 Sunday, 20 March. 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 5 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: This exercise is more challenging than the previous ones. So far we have dealt with fundamental features of Java. Now, however, we are relying heavily on a higher-level library. This means you are going to have to think in a more abstract fashion. You should consult the AWT documentation to understand the classes you are using: `https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html`. Note that this link points to the documentation for Java 7. Some classes may have undergone small changes in the version of Java you are using. Finally, the advice from previous lab sessions, regarding good practices, still applies.

Note: If you change your code but the compiler seems to ignore it, try removing all `.class` files. It is recommended that you compile into a different directory (after removing all `.class` files!), e.g. `javac -d build BankingApp.java`, and then run your program there. **List of requirements:**

R1 (1 mark) Modify the code so that when you click on the "Print client list" button, the list of names of the clients is printed.

R2 (2 marks) Try the button "Add client". The pop-up window that appears is currently useless. Modify the code so that the pop-up window prompts us for the name of the client to be added. A client with the given name must be added to the portfolio of the agent. Analyze the existing code to understand the resources at your disposal.

R3 (1 mark) Uncomment the code marked as "`Uncomment for R3`" (in the constructor of `BankingApp` and in the `addClient` method). Observe that now, when you add a client, a button is added to the bottom of the window. Modify the code so that when each of these buttons is pressed, the information of the corresponding client is printed on the output area in the window. The information should include the client's name and the balance of their account.

R4 (1 mark) Try the button "Deposit". Currently, nothing happens. Modify the code so that when it is clicked, a pop-up window appears, which prompts us for the name of a client and an integer amount. The given amount must be deposited to the given client's account, and a success message printed, if the given name indeed corresponds to a client. If it doesn't, you must print an error message and do nothing else. The pop-up window must have a button to submit the input, and must be closed when this button is clicked. All messages must be printed to the console in the window, not the standard output.

Requirement dependencies:

- In order to test R2 you first need to satisfy R1.
- In order to test R4 you first need to satisfy R3.

Bonus requirements. If you have time to spare and would like to go for some extra points, try implementing the following features:

BR1 (1 mark) Add a button so that we can withdraw from, as well as deposit into, a client's account. This can be easily accomplished with some copying and pasting, but try to go for a more elegant solution.

BR2 (2 marks) Add a button that gives us the option to remove clients. Note that this should result in the removal of the corresponding button from R3 as well.