Assignment 1

instructions and Grading Criteria

- This is an **individual** assessment. Please review the college's **Academic Integrity Policy** to ensure that you are completing your work in an academically honest manner.
- The majority of grades are assigned based on the correct completion of the required functionality and usage of Swift OOP principles. Where needed, you should use appropriate Swift techniques to **safely unwrap optionals** values.
- In addition to the required functionality, learners are expected to use the coding conventions demonstrated in class, meaningful variable naming, and clearly organized code. Comments are helpful but not required.
- This assignment is due on **3rd Feb 2023**. Late submissions will not be accepted.

Submission Checklist

For your submission to be graded, provide a **zip** file of your project, and a **screen recording** demonstrating the functionality you implemented.

1. Create a zip file of your project

- Name the zip file **A1_firstname_lastname.zip**. **.7zip or .rar files will not be accepted**.

3. Creating Your Screen Recording

- In the screen recording, demonstrate the app running in the console, and show the relevant output
- Max 5 min

3. In the assignment:

1. Upload your screen recording to **Microsoft OneDrive** and ensure that the link is set to: "Anyone with the link can view". Paste a link to the recording in the **submission comments**.
2. Submit your zip file containing the project

Academic Integrity

- This is an individual assessment.
- Permitted activities: Usage of Internet to search for syntax only; usage of course materials
- Not permitted:
  - Communication with others (both inside and outside the class)
  - Discussion of solution or approaches with others; sharing/using a "reference" from someone
  - Searching the internet for full or partial solutions
  - Sharing of resources, including links, computers, accounts

Problem Description

Create a console-based application in Swift that simulates a bank account system. The program should have the following classes:

Account: This class should have properties for the account holder's name, account number, and balance. It should have methods to deposit and withdraw money, as well as check the current balance.

ChequingAccount: This class should inherit from the Account class and add a property for the account's overdraft limit. It should override the withdraw method to ensure that the withdrawal does not cause the account to go into overdraft.

SavingsAccount: This class should also inherit from the Account class and add a property for the account's interest rate. It should have a method to add interest to the account balance.

Bank: This class should have properties for a list of accounts and a bank name. It should have methods to add and remove accounts, and to check the total amount of money held in all accounts.

The program should start by creating a bank instance, and then prompt the user to create chequing and savings accounts. The user should be able to deposit, withdraw and check the balance of the account. The program should also allow the user to add interest to the savings account and check the overdraft limit for the chequing account. The program should also allow the user to check the total amount of money held in all accounts. The program should have a loop to allow the user to perform multiple transactions.

Class Definitions

Account class

This class represents an account in the bank. Every account has:

a. Stored properties: **account holder's name**, **account number**, and **balance**.
b. Methods to **deposit money**, **withdraw money** and check the **current balance**.

Chequing Account class

This class should inherit from the **Account** class and add a property for the account's **overdraft limit**. It should override the **withdraw** method to ensure that the withdrawal does not cause the account to go into overdraft.

Savings Account class

This class should also inherit from the **Account** class and add a property for the account's **interest rate**. It should have a **method to add interest** to the account balance.

<u>Bank class</u>

1. This class represents a Bank. Every Bank has:
   a. Stored properties: **bank name** and **list of accounts**
   b. Methods to **add account**, **remove account**, and to check the total amount of money held in all accounts

END OF ASSESSMENT