Last week you created a simple MyBankController class to control a set of bank accounts. When you tested this class, you should have noticed that it was very limited in terms of its ability to create and store and retrieve multiple account objects. This week in order to overcome these limitations, you will modify the controller class, so that each new BankAccount object created is stored in an ArrayList.

Assignment 3: MyBankController class

This modified class should declare fields for an array list of type *BankAccount*. When a MyBankController object is created, it should initialize the array list of type *BankAccount*. In addition to its ***field*** and ***constructor,*** the class should have the following methods:

createAccount (params)

As before, creates a new account object by accepting the necessary parameters from the user and then passing them to the new ***BankAccount(params)*** constructor. New this time is that these new objects are to be stored in the specified ArrayList as they are initialised.

listAllAccounts ()

This method uses a loop to list all accounts contained in the ArrayList. A message should be displayed for each account (call the **displayAccount()** method for the account object)

updateCustomerName(params)

This method passes two parameters (account number and customer name) and uses a loop to go through the list of all accounts and change the customer name of a

specified account. Use the BankAccount **getAccountNumber()** method to check if the account is the specified account to change.
A message should be displayed if the name is changed or if the account is not found.

*Note You must ensure that the BankAccount class has a setter method that allow us to change the customer's name.

removeAccount (param)

This **removeAccount()** method should pass one parameter – accountNumber. An iterator is used to find the account to be removed. If the account is found, remove it from the list and print a message that was removed. Use a return statement to exit the while loop and the method. If the account is not found, provide a message for the user.

Please don't forget to follow good industry practices:

- All class variables start with an underscore
- Write clear comments
- Classes start with a capital letter
- Methods start with a small letter
- Names must be self-explanatory
- Use ***guard*** statements instead of ***if-else*** statements to avoid nesting
- Always use crocodile brackets for if statements
- Declare class variables one per line
- Apply the rule of reusability (if appropriate)

Assignments submissions to follow guidelines specified on Blackboard. Please ensure that you submit code separately for each of the exercises given.

Deadline for submission is **Thursday 18th February @ 11.59pm**