CSc102 2019
Practical 11 (Object-Oriented Programming)

The Problem

The focus of this week's prac is on writing classes in Java. We will be developing a very simple banking application. We will be writing two new classes: one to represent a bank account, and one to manage a list of bank accounts. The class diagrams for the classes that make up our program are shown below.

Account
-accHolder: String -accNumber: String -balance: double
+Account(accNumber: String, accHolder: String, balance: double) +deposit (amt: double): void +withdraw(amt: double): void +getAccNumber(): String +getAccHolder(): String +getBalance(): double +toString(): String

AccountList
-account: LinkedList<Account>
+add(acc: Account): void +remove(acc: Account): void +getNumAccounts(): int +getAccount(k: int): Account +getAccount(accNumber: String): Account +getAccountsForName(customer: String): List<Account>

Bank
-account: AccountList -in: Scanner
+Bank() -createAcc (): void -queryAccByNumber (): void -queryAccByName (): void -displayAccountByNumber (accNo: String): void -findAccountByNumber (accNo: String): Account -displayAccountbyName (name: String): void -displayAccount (acc: Account): void -doDeposit ( ): void -doWithdraw ( ): void -listAll (): void -runTests (): void

The Tasks

1. **CLASSWORK:** Collect your last test paper, and correct the questions you got wrong. Make sure you show your tutor the corrected questions before the end of the prac.

2. **HOMEWORK:** The skeleton of the banking program is available on RUconnected. The first step is to write the `Account` class (for the moment we can simply store the account objects in a `List`).

   The Account class needs to store the details of the account holder (this can just be a string with the customer's name, for example `"G Wells"`), the account number (a string — you can simply make the account numbers up, for example `"123-456"`), and the current balance (a `double` value). The constructor should initialise all the properties. The `deposit` method should simply add the given amount (passed as a parameter) to the current balance, while the `withdraw` method should subtract the given amount from the balance, *if* the balance is greater than the withdrawal amount. If a withdrawal would reduce the balance below zero, the `Withdraw` method should throw an exception with a suitable message.

3. Complete the methods in the program class so that the various operations provided for by the menu all work correctly (and so that the tests pass). You should tackle these one by one. For example, get the "Create account" function working first, then the "Query by number" function, then "Deposit", etc. Print a neat message showing the outcome of each operation. For example, a successful withdrawal should display a one-line message confirming the operation and giving the new balance. Your program should handle all exceptions gracefully (e.g. converting string values into a `double`, and attempting to withdraw more than the current balance).

4. Once you have completed parts 1 and 2, and the program is working correctly, add the `AccountList` class to your program. This should have a method to add an account (i.e. an `Account` object) to the list of accounts, which should throw an exception, if an account with the same account number already exists in the list. The methods to remove an account, and to return the number of accounts in the list are fairly straightforward. The `getAccountsForName` method should return a list of accounts for a given customer name. The `getAccount` method should be overloaded. One version should take an integer parameter *n*, and simply return the *n*'th account in the list (you will need this for the listing of all the accounts). The other should take an account number and either return the corresponding `Account` object or `null`, if the specified account number is not found in the list.

To Hand In:

This prac is due by 5:00pm on Monday 7 October. You should hand in your solution to the problem, with the missing parts of the skeleton file completed, and the two new classes.