Assignment 2

Instructions

You will be completing the two class templates attached: Account and Bank.

Account Class

Create the necessary **instance variables**. These instance variables should **only be accessible inside the Account class**.
The account class will need to store

- An account number, which will be a whole number.
- An account balance representing a quantity of cash.
- An account name.

Complete the bodies of the class methods and the class constructors. DO NOT change any method signature.

- Default constructor
  - Leave blank.
- Constructor with 3 arguments
  - Sets the instance variables of the class
  - Limitations of instance variables must be imposed (stated below)
- getAccountName
  - retrieves the name of the account
- setAccountName
  - returns a Boolean value representing whether the account name has been updated
  - only updates the name of the account if is valid
- getAccountNumber
  - retrieves the account number
- setAccountNumber
  - returns a Boolean value representing whether the account number has been updated
  - only updates the account number if it is valid
- getAccountBalance
  - retrieves the balance of the account
- setAccountBalance
  - returns a Boolean value representing whether the account balance has been updated
  - only updates the balance of the account if it is valid
- equals
  - returns a Boolean value of whether one object is equal in VALUE to an account object

- toString
    - returns a string summary of the account

The following limitations must be imposed via methods:

- The account number can only contain positive numeric values and must be between 5 and 9 digits
- The account name can only contain the following characters:
    - The letters a to z (upper and lowercase) (zero or many times)
    - The hyphen character (zero or many times)
    - The single quote character (zero or one time)
    - The space character (zero or one time)
- The account name must be a minimum of 4 characters
- The account balance can contain positive and negative values but must have a maximum precision of 2 decimal places.
    - Good values: 10, 10.1, 10.22, .1, .23, 0.87, -3.34, -100
    - Not good values: .333, 10.579

IF ANY OF THE ABOVE LIMITATIONS / CONDITIONS ARE NOT MET

- In the constructor
    - The specific invalid instance variable be initialized to a default value that adheres to the limits/conditions required.
        - NOTE: null should never set as a default value
- In the setter method
    - The instance variable should not be updated. The current value should remain.

Adding additional, inner methods are permitted.

Bank Class

Create the necessary **instance variables**. These instance variables should **only be accessible inside the Bank** class.

- It's Bank name. The bank name can only contain alphabetical characters
    - The letters a to z (upper and lowercase) (zero or many times)
    - The numbers 0 to 9 (between zero and three times)
    - The ampersand (&) character (zero or may times)
    - The space character (zero or one time)
    - Bank name should be a minimum of 8 characters
- It's Branch location (one possible value from the enum of BranchLocations)
    - Please add a minimum of 5 BranchLocations value
- A collection of all the Accounts created (from the Account class)

IF ANY OF THE ABOVE LIMITATIONS / CONDITIONS ARE NOT MET

- In the constructor
    - All instance variables should be initialized to a non-null default value that adheres to the limitation and conditions required
- In the setter method
    - The instance variable should not be updated. The current value should remain.

The Bank class will also include the following, which will be explained later

- A method to get a specific account by its account number
- Methods to add an account.
    - Ensure the account number is unique. This is a requirement for the account to be added the Bank.
- Methods to view a specific account
- Methods to modify an account
- Methods to delete an account

Complete the bodies of the class method and the class constructor. DO NOT change any method signature.

- Default constructor
  - Leave blank.
- Constructor with 2 arguments x 2
  - Sets the instance variables of the class
  - Limitations of instance variables must be imposed (stated above)
- getBankName
  - retrieves the name of the bank
- setBankName
  - returns a Boolean value representing whether the bank name has been updated
  - only updates the name of the bank if it is valid
- setBranchLocation x 2
  - returns a Boolean value representing whether the bank branch has been updated
  - only updates the bank branch if it is valid
- getBranchLocation
  - returns a string representation of the branch location instance variable
- getAccountByNumber
  - searches each account in the bank collection of account.
  - returns an Account object if the account number is found in the list of Accounts
  - if the account number is not found, returns an empty object using the default constructor
- addAccount x 2
  - Adds Account to the collections of accounts.
  - account number must be unique in the Bank to be added to the collections of accounts
  - If account number is NOT unique in bank collection of accounts, account is not added
  - returns a Boolean value representing if the account has been added to the list of bank accounts.
- viewAccount(int)
  - searches bank collection of accounts and returns the account that matches the specified account number
  - if the account number is not found, return an empty object using the default constructor
- viewAccount(byte)
  - searches bank collection of accounts by index number (starting from zero)
  - returns the account located at the specified index
  - If the position is not valid, return an empty object using the default constructor

- modifyAccount x 6
    - ability to change the account name and/or balance. See viewAccount to determine how searching for the account should be done.
    - Returns a Boolean value representing whether the specified account has been edited.
        - If BOTH account name and account balance are valid values, the Account instance variables should be modified and the boolean value of true should be returned
        - If EITHER account name or account balance have an invalid value, NIETHER Account instance variable should be modified and the boolean value of false should be returned.
- deleteAccount(int)
    - Attempts to delete the account by searching the list of accounts by account number
        - Deletes the account if found
    - returns a boolean value representing if the account has been found & deleted.
- deleteAccount(byte)
    - Attempts to delete the account by searching the list of accounts by the index number (starting from zero)
        - Deletes the account if found
    - returns a Boolean value representing if the account has been found & deleted.

Adding additional, inner methods is permitted.

Submission

Submit 2 individual .java files on Blackboard: Account.java & Bank.java
Your package name should be **assignment2**.
To view the individual files, right-click on any .java file, select the option Open In, then choose the options Explorer (if you're on Windows) or Finder (if you're on Mac)

Deductions

The following deductions will apply if applicable

Name	Percentage	Description
Late Submission	10% per day	If assignment is not handed in by the due date, this penalty will be applied every 24 hours
File Not Named Correctly	5%	Case-Sensitive naming convention must be followed
Package Not Named Correctly	5%	Case-Sensitive naming convention must be followed
Instance variables & accessibility levels errors	2% per incorrect declaration	Instance variables must be appropriately declared
Submission contains files or folders that are not .java source files	2% per non-java source file	Any extra folder or file found will be penalized at the specified rate
Any other issue that arises	5% per issue fixed	Method signatures altered, syntax or runtime errors fixed

Evaluation

A jUnit tester will be created to evaluate your assignment. There will be a series of tests performed. Your mark will consist of testsPassed / totalTests.
The tester will be released before the deadline so you can see if your java code is running as it should.