

**FACULTY: APPLIED AND COMPUTER SCIENCE DEPARTMENT: SOFTWARE STUDIES**

**ASBDX3B – Development Software 3.1 SEMESTER TEST 1 DATE : September 2018**

**TOTAL DURATION: 180 minutes TOTAL: 100 Marks EXAMINER: S. Moyo MODERATOR: MR S MHLONGO**

INSTRUCTIONS:

**INSTRUCTIONS:**

1. Log into the current PC.
2. Map a network drive to your folder on the exam server.

* Start – Run [\\](file:///\\10.250.10.233\)10.250.10.230 (log in with username: your student number; password: st….)
* Right click on Papers\ – choose: map a network drive. Change the drive letter to W:. Click Finish
* Open W:\RSM\ASBDX3B\_your group\Student number from My Computer. This is the folder in which you must store all your project files.
* Retrieve the SemesterTest.zip file as per invigilator instructions before the start of the test. Extract the compressed file and copy the project and paste it in W:\ASBDX3B\yourgroup\studentNumber. Files not in this folder will not be marked.
* Open the test project using NetBeans and rename it **SurnameBankAccApp** (e.g. **MoyoBankAccApp**)

1. Ensure that your projects are CORRECTLY handed in on Blackboard. If no project is handed in on Blackboard and it is not available on W:, or the project is handed in incorrectly, you get a zero mark for the practical code of this test.
2. Save frequently. No extra time is allocated when the power fails or when your PC must be rebooted. All the code must be saved in the mapped network drive
3. Provide the documentation comment for each source code file, specifying the purpose of the file and the author information.
4. Marks are awarded to the quality of code, and not only to completing the given tasks. Negative marks may be awarded for improper coding techniques used.
5. Hand in a **backup copy** of your compressed project on BlackBoard at the end of the test.

**Assessment regulations**:

1. You may not have any storage media in your possession while in the examination venue. You may also not access remote sites while in the examination venue.

2. No student is allowed to leave the examination venue unaccompanied. No student is allowed to leave the examination venue BEFORE the end of the test. All students are to leave once at the same time.

**Problem Statement**

A local micro-finance company offers the following services to its clients:

* Opening of new accounts
* Deposit into an existing account
* Withdrawal from an existing account

The accounts are either **Check** or **Savings** **accounts**. The promoter of the bank has hired you to develop an application that will provide the above functionalities.

**Part 1 - JUnit Testing** – **NetBeans Project Name: SurnameBankAccApp [92Marks]**

**Instructions:**

* *The source code of this question should be saved in the package:* ***vut.JUnitTest***

A class, **BankAccount**, is provided in the test project. The class stores the name of the account holder, the account number, account type, contact number and account balance. The class have constructors, accessor and mutator methods. Some mutator methods have been coded for you and some are partly done. Test all of them thoroughly. Do not validate the account type.

Using **JUnit testing,** verify that the parameterized constructor and mutator methods function correctly. If a method in the BankAccount class is found to be incorrect, modify the method so that it performs the correct functionality.

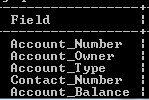
* + Code one test method for the constructor **testConstructor** to test, if the data are assigned correctly. Use the following data: (AccNo: 1001, Acc. name: Sipho Tree, contact No. : 0679990000, Account type: CHECK, Balance: 5000).
  + Code three methods**: testsetAccNoMutator1088, testsetAccNoMutator2001, testsetAccNoMutator-1015** to test the functionality of the **setAccNo** mutator.
  + Code 4 test methods**: testsetAccNameMutatorMduduziNhlapo**, **testsetAccNameMutatorMduNhlapo**, **testsetAccNameMutatorMdu**, **testsetAccNameMutatorMduduziNhlapo** to test the **setAccNam**e mutator using the names, Mduduzi Nhlapo, Mdu Nhlapo, Mdu and MduduziNhlapo respectively.
  + Code 4 test methods **testsetAccContactNo0718976655, testsetAccContactNo2718976655, testsetAccContactNo07189766557, testsetAccContactNo071+897665** to test the **setAccContactNo mutator.**

Also code the **ToString**() method to display the attributes as in the text area of **Figure 3.**

**Part 2 - JDBC** – **NetBeans Project Name: SurnameBankAccApp [Marks]**

**Program Specification**

You need to create a MySQL database named **BankAccountDB.** In this database create a table **Accountstbl** with the fields shown below. (*See class specification for the data type of the fields).*



**User Interface**

Your application includes two interfaces**:** the **AccountsWindow (Figure 1)** and the **AccountsTransactionWindow (Figure 2).**

In the class of the **AccountsWindow** interface, create the following methods:

* **initConnection() -** a method that will be used by the class to connect to the database, BankAccountsDB. THIS METHOD IS PROVIDED.
* **AddNewACcount() –** stores a new account holder to the **Accountstbl.**
* **getRecords()-** retrieves all the records from the **Accountstbl** .
* **displayOnTextBoxes()** –displays in the textboxes the records retrieved by the **getRecords()** method.
* **accountHolder –** uses an account number passed on to it as a parameter **to** retrieve a record from the **Accountstbl** and returns an object of type **BankAccount** class that has a matching account number.
* **withDraw(accNo, withdrawAmt)**- The withdrawal is made by updating the account balance using the formula: Account Balance = Account Balance
* **Deposit (accNo, depAmnt)-** updates the current account balance.

**Use message boxes to display messages if the withdrawal was granted**, or **Withdrawal refused, deposit was successful or unsuccessful, connection was successful or not, new account was created or not. The table should not allow duplicate account holders (similar account numbers)**

**AccountsWindow**

**View Transactions panel**

* When this window initially loads, the **View Transactions** panel is not visible **and a connection to the database must be established, IF IT DOES NOT EXIST YET.**
* The **Account owner**, **Account No** and the **Account Balance** textFields of this panel are not editable.
* The **View Accounts** button allows the user to view the records stored in a result set one at a time from first to last. When it has reached the end of the result set, it goes back to display the first record.
* The **Update button** is used to update the account type or contact number or both fields at the same time of the record in view. The user make changes in the textFields before clicking on this button to update. (DO NOT USE INPUT BOXES TO GET THE INPUT,READ FROM THE TEXTFIELDS AND UPDATE. ALSO DO NOT CREATE A METHOD FOR THIS , WRITE ALL THE CODE FOR UPDATING INSIDE THIS BUTTON).

**Transactions options panel**

The **New Account, Deposit** and the **Withdraw** buttons all load the **AccountsTransactionWindow.**

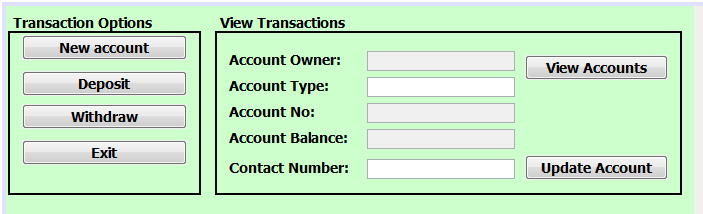
**New Account BUTTON**

When this button is clicked:

1. The **AccountsTransactionWindow** is loaded, and

The **Deposit** and **Withdraw radio buttons and the related textfields** are disabled as in the screenshot shown in **Figure 3**. (*Hint: Create methods in the* ***AccountsTransactionWindow*** *to disable these controls and then* ***call the methods in these button*** *as you know controls are private property they cannot be accessed directly by objects outside their classes.)*

1. The Title of the Window is also set as shown.

**Figure 1**

**Deposit BUTTON**

When this button is clicked:

1. The **AccountsTransactionWindow** is loaded, and
2. The **New Account** and **Withdraw radio buttons and the related textfields** are disabled.
3. The **Savings**, **Check** radio buttons, and the **name** and the **contact number** textField are disabled.

**Withdraw BUTTON**

When this button is clicked:

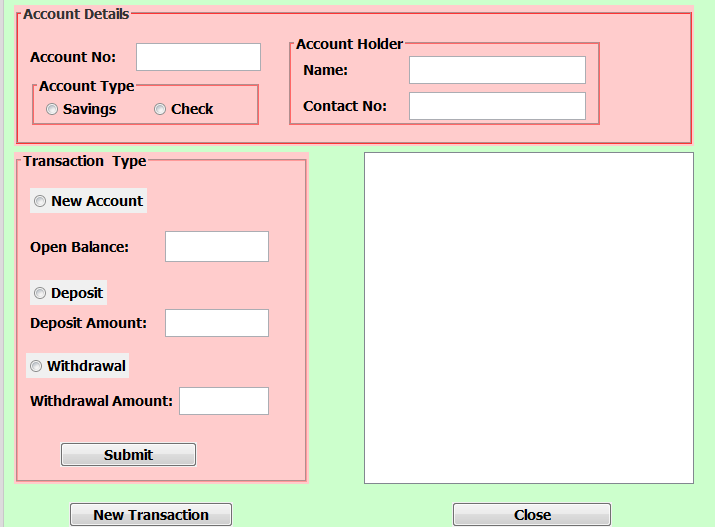
1. The **AccountsTransactionWindow** is loaded, and
2. The **Deposit** and **New Account radio buttons and the related textfields** are disabled.
3. The **Savings**, **Check** radio buttons, and the **name** and the **contact number** textField are disabled.

**Exit BUTTON**

The button stops the **application.**

**AccountsTransactionWindow**

Initially, the **AccountsTransactionWindow** looks like the interface below:

**Figure 2**

The interface must **contain** the following controls:

1. **2 Main jPanels**

**Account Details jPanels** should **contain:**

* **A textField** for the **account number** .numbers are from 1001 to 1099)
* The **Account Type** jPanel containing **2 radio buttons** for **check and saving account**
* The **Account Holder jPanel** containing **2 textFields** for the **name and contact number** of the account holder

**Transaction Type jPanel** should **contain:**

**Three jRadioButtons, labels and textFields**:

* **New Account jRadioButton,** **and a textField** for the input of the **opening balance amount.**
* **Deposit jRadioButton and a textField** for the input of the **deposit amount**.
* **Withdrawal jRadioButton and a textField** for the input of the **withdrawal amount**.
* A **Submit Button** for executing the transaction

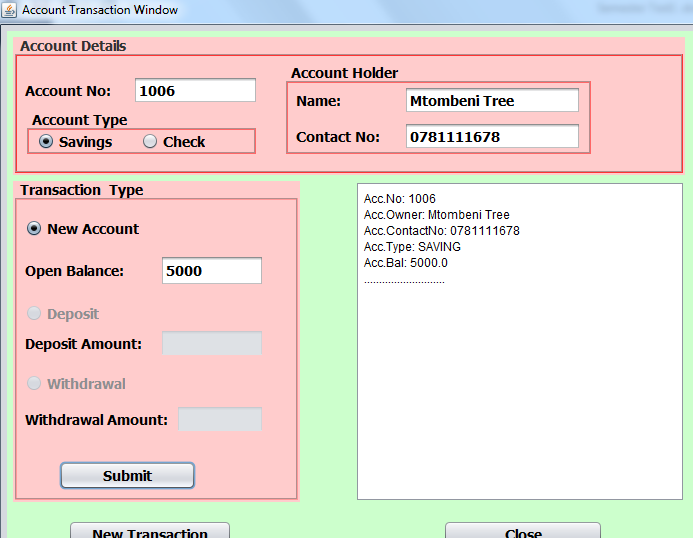
1. **An output textArea** for **displaying the details of all transactions** performed.
2. **Two independent** buttons**: New transaction** and **Close** buttons

**Submit button**

When the **Submit Button** is clicked, the following actions should be taken:

* Get the **bank account number**, from the account number input control.
* Get the **account type** from the selected **radio button.**
* If the **New Account radio button** (within the **Transaction Type panel**) is enabled:
  + Get the **account holder name** and **contact number** from the input controls.
  + Get the **open balance amount** from the text box
  + **Instantiate** a **BankAccount class object** and save the **new account information** to the database using the **AddNewACcount().**
* If the **Deposit radio button** (within the **Transaction Type panel**) is visible:
  + Get the **amount** to be deposited into the account from the **Deposit Amount text box control**
  + Call the **Deposit**() **method**
* If the **Withdraw radio buttons (**within the **Transaction panel)** is visible:
  + Get the **amount** needed from the **Withdrawal amount text box control**
  + Call the **Withdraw**() **method**

The output of each Transaction must be display the in the text area as shown in **Figure 3**. Call the **accountHolder()** method to display the output.

**Figure 3**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GOOD LUCK\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***