

Bank Account Management System



Team Members: Natan Lellouche, Kyle Husbands
420-301-VA PROGRAMMING PATTERNS
Teacher: Adrian Constantin Onet
Monday, May 15th, 2022

Table of Contents

Introduction	3
Design Patterns	3
Internalization /Localization	3
Teller Menu	5
Main Menu	8
Client Menu	10
Client Sub Menu	12
Account Menu	17
Account Sub Menu	19
Transaction Menu	24
Junit Testing	36

1.0 Introduction

The Bank Account Management System (BAMS) is a limited bank transaction monitoring software. This software will be used by the bank tellers to add new clients, manage accounts, create transactions and return client reports. Transactions will be considered either between 2 accounts (move money from one account to the other) or cash transactions: deposit into an account or extract from an account. The teller will be able to create balance report sheets for a given client. Negative balances are not allowed

**** Note:** For the Database and the JUnit to run you need to change the path to the database******

1.1 Design Patterns / MVC's

In this project we needed to use some Design Patterns as well as to incorporate our project like that it followed the MVC guidelines. For the design patterns we tried to use a Factory Design pattern as it was the easiest to use in our project and allowed us to make our project run. As for the MVC we tried to make it like that the controller was an inbetween for the models and the view and implement it into our code.

2.0 Localization/ Internalization

For this term project we were forced to incorporate localization and internalization so that people who speak French would also be able to use our Banking System. The way that our group went about doing this was by creating two .properties files that would hold the text that we wanted for each specific language. After changing the String values for the two different languages all that was left was to ask the user what language he wanted and then have an if loop change the locale to that language.

```
run:
Welcome to your Banking Service! / Bienvenue dans votre service bancaire!
-----

What Language Would you like to use? / Quelle langue souhaitez-vous utiliser?

Press 1 for English
Press 2 for French
1

Welcome, would you like to login or create a teller?

Press 1 to login
Press 2 to create
```

```
run:
Welcome to your Banking Service! / Bienvenue dans votre service bancaire!
-----

What Language Would you like to use? / Quelle langue souhaitez-vous utiliser?

Press 1 for English
Press 2 for French
2

Bienvenue, souhaitez-vous vous connecter ou créer un caissier?

Appuyez sur 1 pour vous connecter
Appuyez sur 2 pour créer

1
Entrez votre numéro de connexion:
|
```

Some general example:

French internalization of the date, as well as the currency:

```
-----  
Numéro_de_compte: 5656  
  
Identité_du_client: 1111  
  
Type_de_compte: Chequer  
  
Solde: 0,00 €  
  
Date_d'ouverture: 15 mai 2022  
  
C'est_Actif:  
  true  
-----  
  
Entrez l'ID du compte à gérer:
```

1.0 Teller Menu

To begin, after choosing a language, you are asked to login by pressing 1 or create a teller by pressing 2, to access the “BAMS”.

Assuming there aren't any tellers in the database, we press 2 to create a teller, and we input the following information.

```
Welcome, would you like to login or create a teller?
```

```
Press 1 to login
```

```
Press 2 to create
```

```
2
```

```
Please create account
```

```
Enter your name:
```

```
Kyle
```

```
Create your login number:
```

```
1234
```

```
Create your login password:
```

```
9999
```

```
Enter your password again:
```

```
9999
```

If there is a teller in the database, we can press 1 to login, with the correct information:

```
Welcome, would you like to login or create a teller?
```

```
Press 1 to login
```

```
Press 2 to create
```

```
1
```

```
Enter your login number:
```

```
1234
```

```
Enter your login password:
```

```
9999
```

Exception Handling:

We also handle exceptions that make sure that our code is able to deal with all possible scenarios to ensure there are no incorrect inputs.

If the user tries to press 1 to login, but there aren't any tellers in the database, you will be forced to create one.

```
Welcome, would you like to login or create a teller?

Press 1 to login
Press 2 to create

1
Currently no Tellers exist, please create a Teller first!

Enter your name:
Kyle
Create your login number:
1234
Create your login password:
9999
Enter your password again:
9999
```

In this example, even though the user tried to login as a teller, since that teller does not exist it will not let you login.

```
Welcome, would you like to login or create a teller?

Press 1 to login
Press 2 to create

1
Enter your login number:
123432
Enter your login password:
78898

information not correct, try again!

Enter your login number:
_
```

Another exception that we have handled is if the user tries to create a user with the same login ID. In this case a message will pop up saying that

a Teller already has the same login and that you need to pick a different one. The data will also not be included into the Database.

```
Welcome, would you like to login or create a teller?
```

```
Press 1 to login
```

```
Press 2 to create
```

```
2
```

```
Please create account
```

```
Enter your name:
```

```
Natan
```

```
Create your login number:
```

```
1234
```

```
Create your login password:
```

```
6666
```

```
Enter your password again:
```

```
6666
```

```
Id must be unique!
```

```
Please create account
```

```
Enter your name:
```

2 Main Menu

If you successfully created or logged in with a teller, you will then be prompted with the main menu:

In our main menu we have 4 options, press 1 to view the client menu, press 2 to view the account menu, press 4 to view the transaction menu, and press 4 to exit the console.


```
Welcome Kyle!  
Please select one of the following options:  
  
Press 1 for client menu  
Press 2 for account menu  
Press 3 for Transaction menu  
Press 4 to exit
```

Exception Handling:

For this menu if the user picks a number that is above 4 the program will tell it that it must pick one of the desired numbers and re-display the menu.

```
Welcome Kyle!  
Please select one of the following options:  
  
Press 1 for client menu  
Press 2 for account menu  
Press 3 for Transaction menu  
Press 4 to exit  
5  
  
menu option cannot be greater than 4  
  
Please select one of the following options:  
  
Press 1 for client menu  
Press 2 for account menu  
Press 3 for Transaction menu  
Press 4 to exit
```

3 Client Menu

In our client menu we have 3 options, press 1 to view and manage all clients, press 2 to create a client, and press 3 to exit the client menu.

```
Welcome to client menu:  
  
Press 1 to view and manage all current clients  
Press 2 to create a client  
Press 3 to exit
```

we can also create a client by pressing 2.

```
Welcome to client menu:  
  
Press 1 to view and manage all current clients  
Press 2 to create a client  
Press 3 to exit  
  
2  
Enter a client ID:  
4444  
Enter your client first name:  
Max  
Enter your client last name:  
Jones  
Enter your client Identification:  
gym  
Enter your client Address:  
baltimore  
  
Max Jones was Added!
```

And we can now see Max Jones was successfully added to the list of clients.

```
LastName: Bridgman
```

```
Identification: social
```

```
Address: toronto
```

```
-----
```

```
-----
```

```
ClientID: 4444
```

```
FirstName: Max
```

```
LastName: Jones
```

```
Identification: gym
```

```
Address: baltimore
```

```
-----
```

```
Enter a client ID:
```

```
4444
```

```
What would you like to do with Max Jones
```

```
Press 1 to view client details
```

```
Press 2 to update Client
```

```
Press 3 to exit
```

3.1 Client Sub Menu

If we press 1, we will be redirected to the client sub menu, and you will be shown a list of all the current clients, and given an option to enter the id of a client you want to further access:

```
-----  
ClientID: 1111  
  
FirstName: Kyle  
  
LastName: Husbands  
  
Identification: drivers  
  
Address: license  
-----  
  
-----  
ClientID: 2222  
  
FirstName: Natan  
  
LastName: Lellouche  
  
Identification: medical  
  
Address: laval  
-----  
  
-----  
ClientID: 3333  
  
FirstName: Nathaniel  
  
LastName: Bridgman  
  
Identification: social  
  
Address: toronto  
-----  
  
Enter a client ID:
```

If we want to manage Natan Lellouche, we enter his client ID which is 2222, then we will be given a sub menu for that specific client.

In this sub menu there are 3 options, press 1 to view that client's details, press 2 to update that client's information, or press 3 to exit.

```
Enter a client ID:
```

```
2222
```

```
What would you like to do with Natan Lellouche
```

```
Press 1 to view client details
```

```
Press 2 to update Client
```

```
Press 3 to exit
```

If we press 1, we will be shown the information of Natan Lellouche:

```
What would you like to do with Natan Lellouche
```

```
Press 1 to view client details
```

```
Press 2 to update Client
```

```
Press 3 to exit
```

```
1
```

```
-----  
ClientID: 2222
```

```
FirstName: Natan
```

```
LastName: Lellouche
```

```
Identification: medical
```

```
Address: laval
```

```
-----
```

If we press 2, we will be given the option to update Natan Lellouche's information. And as you can see there is no option to update the client ID as we know this will violate the primary key rules of our database.

```
What would you like to do with Natan Lellouche
```

```
Press 1 to view client details
```

```
Press 2 to update Client
```

```
Press 3 to exit
```

```
2
```

```
Enter your client first name:
```

```
Jimmy
```

```
Enter your client last name:
```

```
Newtron
```

```
Enter your client Identification:
```

```
Security
```

```
Enter your client Address:
```

```
chicago
```

```
Client Updated!:
```

```
What would you like to do with Jimmy Newtron
```

```
Press 1 to view client details
```

```
Press 2 to update Client
```

```
Press 3 to exit
```

To confirm it updated, we can check by displaying the updated clients information: we can also create a client by pressing 2.

```
What would you like to do with Jimmy Newtron
```

```
Press 1 to view client details
```

```
Press 2 to update Client
```

```
Press 3 to exit
```

```
1
```

```
-----
```

```
ClientID: 2222
```

```
FirstName: Jimmy
```

```
LastName: Newtron
```

```
Identification: Security
```

```
Address: chicago
```

```
-----
```

Exceptions

The first exception that we handled in the client menu was that if we try to view the list of clients while there are no clients in the database, we will get an error message:

```
Welcome to client menu:

Press 1 to view and manage all current clients
Press 2 to create a client
Press 3 to exit

1
There are no current clients!

Welcome to client menu:

Press 1 to view and manage all current clients
Press 2 to create a client
Press 3 to exit
```

If we enter an incorrect ID of a client when trying to view its details we are given an error:

```
Press 1 to view and manage all current clients
Press 2 to create a client
Press 3 to exit
```

```
1
```

```
-----
ClientID: 1111
```

```
FirstName: Maxime
```

```
LastName: Paul
```

```
Identification: drivers
```

```
Address: mtl
```

```
-----
```

```
Enter a client ID:
```

```
2222
```

```
Client ID does not match:
```

```
Enter a client ID:
```

We also considered the possibility of a user trying to add a client with the same ID as one in the database, this violates primary key rules, so it is handled with the following error:


```
Welcome to client menu:
```

```
Press 1 to view and manage all current clients
```

```
Press 2 to create a client
```

```
Press 3 to exit
```

```
2
```

```
Enter a client ID:
```

```
1111
```

```
Enter your client first name:
```

```
Natan
```

```
Enter your client last name:
```

```
Lellouche
```

```
Enter your client Identification:
```

```
security
```

```
Enter your client Address:
```

```
chicago
```

```
Client Id already exists, cannot create Client!
```

4 Account Menu

When the Account menu is loaded the user will be given three options and depending on which option the user chooses the program will open up another page.

```
Welcome to Account menu:
```

```
Press 1 to view and manage all Accounts
```

```
Press 2 to create an Account
```

```
Press 3 to exit
```

By clicking 1 in the Account Menu the user will be able to see all of the accounts in general, and given an option to enter the id of an Account that you want to further access inside of the Account Sub Menu.

```
Welcome to Account menu:

Press 1 to view and manage all Accounts
Press 2 to create an Account
Press 3 to exit
1
-----
AccountNumber: 1212

ClientId: 1111

AccountType: Savings

Balance: $0.00

OpenDate: May 15, 2022

isActive:
  true
-----

Enter ID of account to manage:
```

If the user picks option 2, the user will be able to create an Account and assign it to a client.

```
Welcome to Account menu:

Press 1 to view and manage all Accounts
Press 2 to create an Account
Press 3 to exit
2
Enter your Account ID:
1212
Which client would you like to assign this account to?
1111
Which Type of account would you like to create?
Checkings
Account created!
```

4.1 Account Sub Menu

The Account Sub Menu can only be accessed once the user picks a specific Account that he would like to manage.

```
Enter ID of account to manage:

2323
What would you like to do with Account 2323?

Press 1 to view Account details
Press 2 to deactivate this account
Press 3 to activate this account
Press 4 to exit
```

If the user picks option 1 the program will display all of the details of that specific account.

```
What would you like to do with Account 1212?
```

```
Press 1 to view Account details
```

```
Press 2 to deactivate this account
```

```
Press 3 to activate this account
```

```
Press 4 to exit
```

```
1
```

```
-----
```

```
AccountNumber: 1212
```

```
ClientId: 1111
```

```
AccountType: Savings
```

```
Balance: $0.00
```

```
OpenDate: May 15, 2022
```

```
isActive:
```

```
  true
```

```
-----
```

If the user decides to pick option 2, the program will make sure that the account that the user has chosen has a balance of 0 and then it will deactivate said account.

```
What would you like to do with Account 1234?  
  
Press 1 to view Account details  
Press 2 to deactivate this account  
Press 3 to activate this account  
Press 4 to exit  
2  
The account was deactivated!
```

Much like option 2 if the user decides to pick option 3, the program will reactivate said account.

```
What would you like to do with Account 1234?  
  
Press 1 to view Account details  
Press 2 to deactivate this account  
Press 3 to activate this account  
Press 4 to exit  
3  
The account was activated!
```

Exceptions:

As many of the exceptions handled are the same for the client menu, we won't reiterate them, but we will discuss the new ones for the account menu.

If the user attempts to create an account while there are no current clients, you will be given the following error message:

```
Welcome to Account menu:
```

```
Press 1 to view and manage all Accounts
```

```
Press 2 to create an Account
```

```
Press 3 to exit
```

```
2
```

```
You must create a client to create an account!
```

```
Welcome to Account menu:
```

```
Press 1 to view and manage all Accounts
```

```
Press 2 to create an Account
```

```
Press 3 to exit
```

If the user attempts to create an account, but enters a client ID that does not exist, you will be given the following error message:

```
Welcome to Account menu:  
  
Press 1 to view and manage all Accounts  
Press 2 to create an Account  
Press 3 to exit  
2  
Enter your Account ID:  
1212  
Which client would you like to assign this account to?  
2222  
Client Id entered does not exist!  
Which client would you like to assign this account to?
```

If the user attempts to deactivate an account that has a balance greater than 0, they will be given the following error message:

```
What would you like to do with Account 1212?  
  
Press 1 to view Account details  
Press 2 to deactivate this account  
Press 3 to activate this account  
Press 4 to exit  
2  
You cannot deactivate an account with a balance!
```

5 Transaction Menu

If the user picks the Transaction menu, the program will display all of the options that a user can do regarding transactions.

```
Welcome to Transaction Menu

Press 1 to view list of all transactions
Press 2 make a transaction between accounts
Press 3 make a deposit
Press 4 make a withdraw
Press 5 reverse a transaction
Press 6 to exit
```

If the user picks option 1, they will be able to see all of the transactions that have been made in all accounts. These transactions will also be ordered by when they were created so that the user can easily see which Transaction is newer and which is older. It also tells you the type of transaction that was made, and the value of that transaction. The transaction ID is also automatically generated to increase by 1, for each transaction that is made.


```
1
-----
TransactionNumber: 1000

ToAccountNumber: 1212

FromAccountNumber: 0

TransactionDetail: Deposit

Value:
  $500.00
-----

-----
TransactionNumber: 1001

ToAccountNumber: 1212

FromAccountNumber: 0

TransactionDetail: Withdraw

Value:
  $100.00
-----
```

If the user picks option 2, the user will first be greeted with a list of only the **active** accounts, they will then be able to make a transaction between two Accounts. The program will ask the user to input two different account ID's and then ask them how much money they would like to transfer between the two.

Here are the current active accounts

AccountNumber: 1212

ClientId: 1111

AccountType: Savings

Balance: \$400.00

OpenDate: May 15, 2022

isActive:
true

AccountNumber: 2323

ClientId: 1111

AccountType: Checkings

Balance: \$0.00

OpenDate: May 15, 2022

isActive:
true

Which Accounts would you like to transfer to?
From Account:

```
Which Accounts would you like to transfer to?  
From Account:  
1212  
To Account:  
2323  
How much would you like to transfer?  
400  
You successfully transferred $400.00
```

For option 3, once the user clicks that button they will be allowed to make a deposit of any amount to a specific account. The program will check to make sure that the user is active and if it is, the amount will be updated into the Database and that account will have the money deposited plus its original sum.

```
Welcome to Transaction Menu
```

```
Press 1 to view list of all transactions
```

```
Press 2 make a transaction between accounts
```

```
Press 3 make a deposit
```

```
Press 4 make a withdraw
```

```
Press 5 reverse a transaction
```

```
Press 6 to exit
```

```
3
```

```
Which Account would you like to transfer to?
```

```
To Account:
```

```
1212
```

```
How much would you like to transfer?
```

```
500
```

```
You successfully deposited $500.00
```

Similarly ,for option 4, the user will be allowed to make a withdrawal to a specific account. The program will check to make sure that the user is active and if it is the amount that they want to withdraw is equal or less then the amount that they currently have. If everything checks out the amount will be updated into the Database and that account will have its original sum minus the money withdrawn.

```
Welcome to Transaction Menu
```

```
Press 1 to view list of all transactions
```

```
Press 2 make a transaction between accounts
```

```
Press 3 make a deposit
```

```
Press 4 make a withdraw
```

```
Press 5 reverse a transaction
```

```
Press 6 to exit
```

```
4
```

```
Which Account would you like to withdraw from?
```

```
From Account:
```

```
1212
```

```
How much would you like to withdraw?
```

```
100
```

```
You successfully withdrew $100.00
```

Option 5 of our transaction menu is to “reverse” any transaction, this essentially will create a new transaction that will perform the inverse operation.

First you will be prompted with a list of all the transactions made.

```
-----  
TransactionNumber: 1001  
  
ToAccountNumber: 1212  
  
FromAccountNumber: 0  
  
TransactionDetail: Withdraw  
  
Value:  
  $100.00  
-----  
  
-----  
TransactionNumber: 1002  
  
ToAccountNumber: 2323  
  
FromAccountNumber: 1212  
  
TransactionDetail: Transfer  
  
Value:  
  $400.00  
-----  
  
Enter ID of transaction to reverse:
```

You will then be prompted to enter the id of a transfer you would like to “reverse”. The program will then check the ID and match it with the transaction, it will then check which type of transaction was made, and will perform the necessary inverse operation.

```

-----
TransactionNumber: 1000

ToAccountNumber: 1212

FromAccountNumber: 2323

TransactionDetail: Transfer

Value:
  $400.00
-----

Enter ID of transaction to reverse:

1000
Are you sure you want to reverse the transaction of $400.00 $ from account 2323 to account 1212?

Type yes or no

yes
Transaction of $400.00 $ from account 2323 to account 1212 was Successfully Reversed

```

Example of reverse deposit:

```

-----
TransactionNumber: 1004

ToAccountNumber: 1212

FromAccountNumber: 0

TransactionDetail: Deposit

Value:
  $500.00
-----

Enter ID of transaction to reverse:

1004
Are you sure you want to reverse the deposit transaction of $500.00$ to account 1212?

Type yes or no

yes
Deposit Transaction of$500.00$ to account 1212was Successfully Reversed

```

Example of reverse withdraw:

```
-----  
TransactionNumber: 1006  
  
ToAccountNumber: 2323  
  
FromAccountNumber: 0  
  
TransactionDetail: Withdraw  
  
Value:  
  $200.00  
-----  
  
Enter ID of transaction to reverse:  
  
1006  
Are you sure you want to reverse the withdraw transaction of $200.00$ from account 2323?  
  
Type yes or no  
  
yes  
Withdraw Transaction of$200.00$ to account 2323was Successfully Reversed
```

Exceptions

One of the exceptions that is handled in this menu is that if a user tries to see the transaction history there are none the program will display a message saying that there are currently no transactions.

```
Welcome to Transaction Menu  
  
Press 1 to view list of all transactions  
Press 2 make a transaction between accounts  
Press 3 make a deposit  
Press 4 make a withdraw  
Press 5 reverse a transaction  
Press 6 to exit  
  
1  
There are no current Transactions!
```


Another exception in this menu is if the user tries to make a transaction between two accounts but there are fewer than two active accounts a message will show up saying that there must be at least two active accounts to pick this option

```
Welcome to Transaction Menu

Press 1 to view list of all transactions
Press 2 make a transaction between accounts
Press 3 make a deposit
Press 4 make a withdraw
Press 5 reverse a transaction
Press 6 to exit
2
"You must have at least 2 active accounts to make a transfer!"
```

A similar exception will be thrown for both the Deposit and the withdraw if no Account exist to do these actions

```
Welcome to Transaction Menu

Press 1 to view list of all transactions
Press 2 make a transaction between accounts
Press 3 make a deposit
Press 4 make a withdraw
Press 5 reverse a transaction
Press 6 to exit
3
You must have at least 1 active accounts to make a Deposit!
```

```
Welcome to Transaction Menu
```

```
Press 1 to view list of all transactions
```

```
Press 2 make a transaction between accounts
```

```
Press 3 make a deposit
```

```
Press 4 make a withdraw
```

```
Press 5 reverse a transaction
```

```
Press 6 to exit
```

```
4
```

```
You must have at least 1 active accounts to make a Withdraw!
```

if no transaction exists the user will be unable to choose the reverse transaction as an option(5).

```
Welcome to Transaction Menu
```

```
Press 1 to view list of all transactions
```

```
Press 2 make a transaction between accounts
```

```
Press 3 make a deposit
```

```
Press 4 make a withdraw
```

```
Press 5 reverse a transaction
```

```
Press 6 to exit
```

```
5
```

```
There are no current Transactions!
```

If the user tries to make a transfer between accounts that don't exist or that aren't currently active, we see this error message:

```
Which Accounts would you like to transfer to?  
From Account:  
23  
To Account:  
24  
You can only transfer between active accounts!
```

If the user tries to transfer more money between accounts that the current balance of the “from” account, this error message is thrown:

```
Which Accounts would you like to transfer to?  
From Account:  
2323  
To Account:  
1212  
How much would you like to transfer?  
500  
Insufficient funds to be transferred!
```

If the user tries to withdraw more money than what is currently in the account, this error is thrown:

```
Which Account would you like to withdraw from?  
From Account:  
2323  
How much would you like to withdraw?  
500  
Insufficient funds to be withdrawn!
```

If the user tries to make a reverse transfer, but the balance of the account no longer has enough money, the following error will be thrown:

```
Enter ID of transaction to reverse:
1002
Are you sure you want to reverse the transaction of $400.00 $ from account 1212 to account 2323?
Type yes or no
yes
Transaction funds are insufficient!
```

If the user tries to make a reverse deposit, but the account no longer has the money, the following error will be thrown, in this example the account had made a deposit of 1000\$, but later made a withdrawal of 500\$, so when attempting to reverse the original 1000\$ deposit, the error was thrown.

```
-----
TransactionNumber: 1009

ToAccountNumber: 1212

FromAccountNumber: 0

TransactionDetail: Withdraw

Value:
$500.00
-----

Enter ID of transaction to reverse:
1008
Are you sure you want to reverse the deposit transaction of $1,000.00$ to account 1212?
Type yes or no
yes
Transaction funds are insufficient!
```

6 JUnit Testing

To make sure that our code logic worked properly Kyle and I decided to make a couple of JUnit tests. We did specific test cases for each of the main parts including the Client, Account, Teller, and Transactions and most of these cases were used to make sure that our Database worked right and that edge cases gave the proper values back.

- Account Case

**** These are just some of the methods that we tested ****

```
@Test
public void testCreateAccount() throws SQLException {
    System.out.println("createAccount");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Documents\\NetBeansProjects\\BankTest2\\");
    Accounts instance = new Accounts(4444, 1221, "Checkings");
    boolean expResult = true;
    boolean result = instance.createAccount(con);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of viewAccounts method, of class Accounts.
 */
@Test
public void testViewAccounts() throws SQLException {
    System.out.println("viewAccounts");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Documents\\NetBeansProjects\\BankTest2\\");
    Accounts instance = new Accounts();
    ResultSet result = instance.viewAccounts(con);
    assertNotNull(result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

```

@Test
public void testMatchClients() throws SQLException {
    System.out.println("matchClients");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Documents\\NetBeansProjects\\BankTest2\\db\\bankTest2.db");
    Accounts instance = new Accounts();
    boolean expResult = true;
    boolean result = instance.matchClients(con, 12);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of DeActivate method, of class Accounts.
 */
@Test
public void testDeActivate() throws SQLException {
    System.out.println("DeActivate");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Documents\\NetBeansProjects\\BankTest2\\db\\bankTest2.db");
    Accounts instance = new Accounts();
    boolean expResult = true;
    boolean result = instance.DeActivate(con, 4444);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

```

```






@Test
public void testReActivate() throws SQLException {
    System.out.println("ReActivate");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Documents\\NetBeansProjects\\BankTest2\\db\\bankTest2.db");
    Accounts instance = new Accounts();
    instance.ReActivate(con, 4444);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of viewSpecificAccount method, of class Accounts.
 */
@Test
public void testViewSpecificAccount() throws SQLException {
    System.out.println("viewSpecificAccount");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Documents\\NetBeansProjects\\BankTest2\\db\\bankTest2.db");
    Accounts instance = new Accounts();
    ResultSet result = instance.viewSpecificAccount(con, 4444);
    assertNotNull(result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

```

Test Results

finalproject2.AccountsTest.testViewAccounts X

Test Results	Test Name
<p>Tests passed: 100.00 %</p> <p>The test passed. (0.397 s)</p> <p>      </p>	viewAccounts

- Client Case

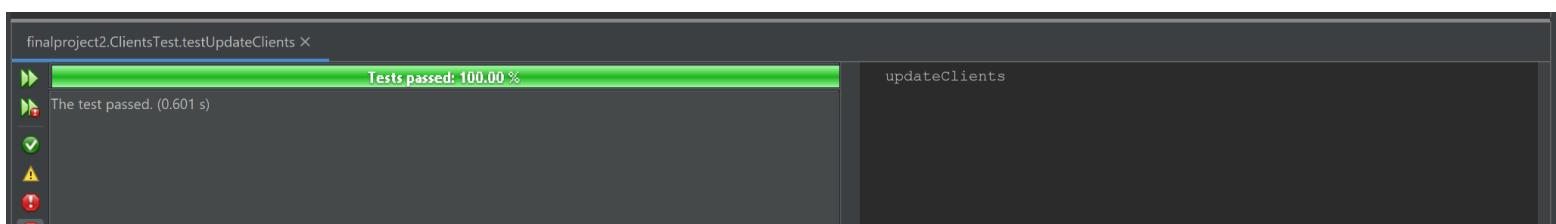
**** These are just some of the methods that we tested ****

```
@Test
public void testCreateClient() throws ClassNotFoundException, SQLException {
    System.out.println("createClient test");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Clients instance = new Clients(1111, "Jimmy", "Newtron", "Drivers", "MTLS");
    boolean expResult = true;
    boolean result = instance.createClient(con);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

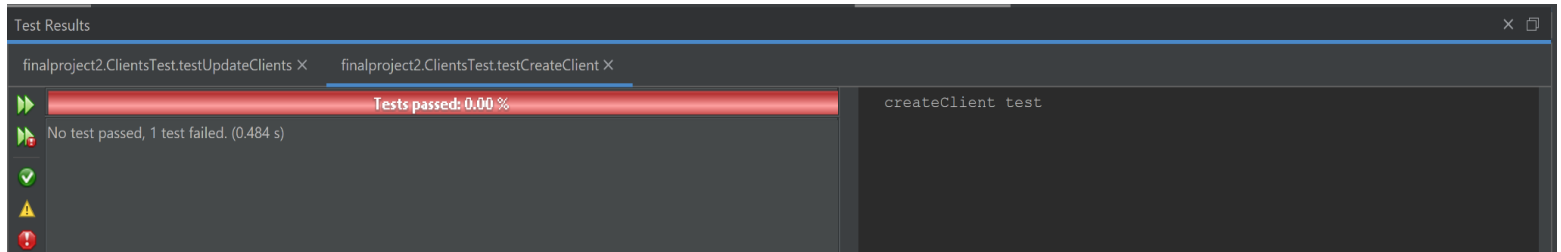
/**
 * Test of viewClient method, of class Clients.
 * @throws java.sql.SQLException
 */
@Test
public void testViewClient() throws SQLException {
    System.out.println("viewClient test");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Clients instance = new Clients();
    ResultSet result = instance.viewClient(con);
    assertNotNull(result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

```
@Test
public void testViewSpecificClient() throws SQLException {
    System.out.println("viewSpecificClient");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Clients instance = new Clients();
    int id = 1221;
    ResultSet result = instance.viewSpecificClient(con, id);
    assertNotNull(result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of updateClients method, of class Clients.
 * @throws java.sql.SQLException
 */
@Test
public void testUpdateClients() throws SQLException {
    System.out.println("updateClients");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Clients instance = new Clients();
    instance.updateClients(con, 1221, "Max", "Newt", "Driv", "CHI");
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```



This test case failed correctly as we purposely tried to add a user into the database that already had the same ID.



- Transaction

**** These are just some of the methods that we tested ****

```
@Test
public void testMakeTranBetween() throws SQLException {
    System.out.println("makeTranBetween");
    Connection con = DriverManager.getConnection("jdbc:sqlite:\\C:\\Users\\natan\\OneDrive - Vanier College\\Doc
    Transactions instance = new Transactions(1111, 2222, 500);
    instance.makeTranBetween(con, 0);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of reverseTransaction method, of class Transactions.
 */
@Test
public void testReverseTransaction() throws SQLException {
    System.out.println("reverseTransaction Test when corrcet");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Transactions instance = new Transactions();
    boolean result = instance.reverseTransaction(con, 2222);
    assertNotNull(result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

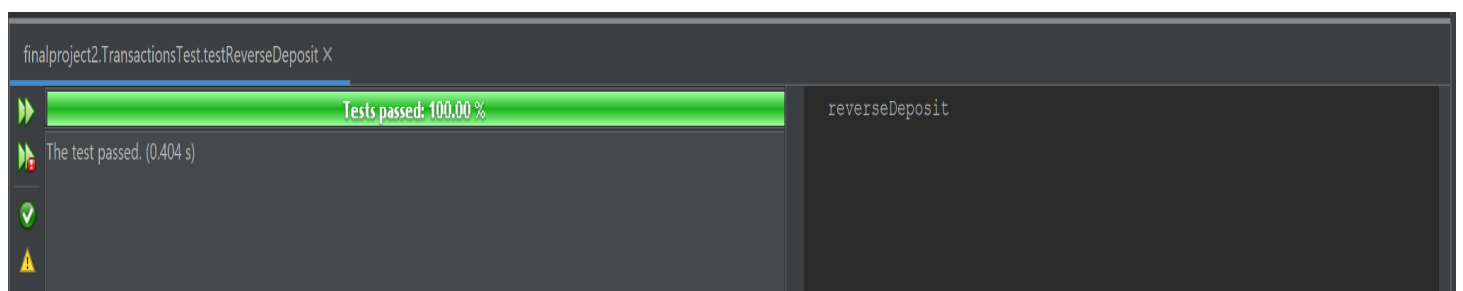
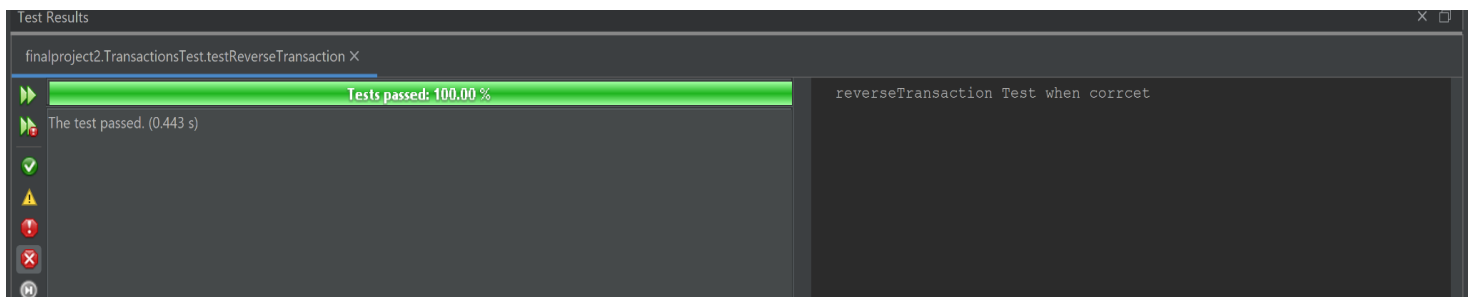
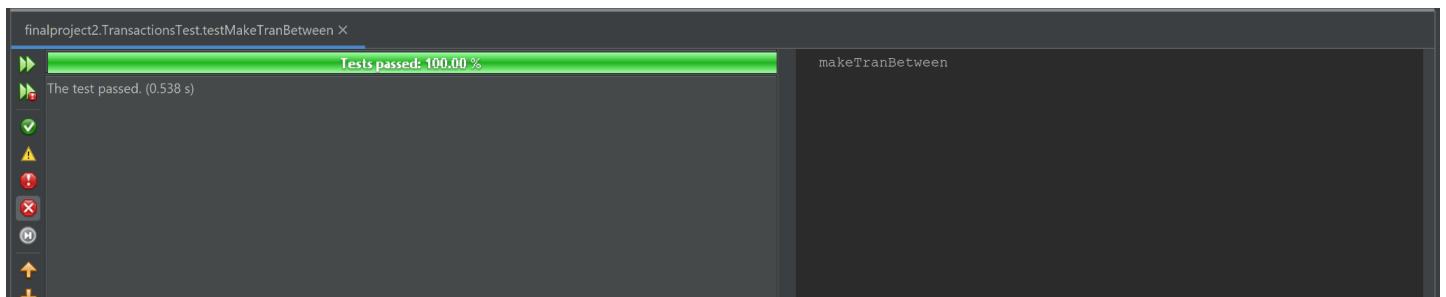


```

@Test
public void testReverseWithdraw() throws SQLException {
    System.out.println("reverseWithdraw");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Transactions instance = new Transactions();
    instance.reverseWithdraw(con, 2222);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of calculateTransfer method, of class Transactions.
 * @throws java.sql.SQLException
 */
@Test
public void testCalculateTransfer() throws SQLException {
    System.out.println("calculateTransfer");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Transactions instance = new Transactions(1111, 2222, 500);
    boolean result = instance.calculateTransfer(con);
    assertNotNull(result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

```



finalproject2.TransactionsTest.testReverseWithdraw X

Tests passed: 100.00 %

The test passed. (0.562 s)

▶▶

▶

✓

⚠

❗

✖

⌂

↑

reverseWithdraw

finalproject2.TransactionsTest.testCalculateTransfer X

Tests passed: 100.00 %

The test passed. (0.323 s)

▶▶

▶

✓

⚠

❗

✖

⌂

↑

calculateTransfer

finalproject2.TransactionsTest.testDeposit X

Tests passed: 100.00 %

The test passed. (0.591 s)

▶▶

▶

✓

⚠

❗

✖

⌂

Deposit

finalproject2.TransactionsTest.testWithdraw X

Tests passed: 100.00 %

The test passed. (0.394 s)

▶▶

▶

✓

⚠

❗

✖

⌂

Withdraw

finalproject2.TransactionsTest.testViewTransactions X

Tests passed: 100.00 %

The test passed. (0.413 s)

▶▶

▶

✓

⚠

❗

✖

⌂

viewTransactions

- Teller

**** These are just some of the methods that we tested ****

```
@Test
public void testSignUp() throws SQLException, ClassNotFoundException {
    System.out.println("signUp test");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Teller instance = new Teller("1234", "9999", "9999", "Kyle");
    assertEquals(true, instance.signUp(con));
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of Login method, of class Teller.
 */
@Test
public void testLogin() throws ClassNotFoundException, SQLException {
    System.out.println("Login test");
    Connection con = DriverManager.getConnection("jdbc:sqlite://C://Users//ben_1//Desktop//BankTest.db");
    Teller instance = new Teller("1234", "9999", "9999", "Kyle");
    String expResult = "Kyle";
    String result = instance.Login(con);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

