|  |  |
| --- | --- |
| sait_icon_wordmark_horiz_text_black | **CPRG 251**  **Assignment 2 (Modules 4-8) Bank on It Frontend** |

**Name:**

**Marks:**  / 100

## **D2L Submission Instructions**

1. One ZIP file needs to submit to D2L with the following naming convention **CPRG251\_A2\_Firstname\_Lastname.zip** using your first and last name.

If working in a group of two (2), only one team member needs to submit to D2L (both can if you so wish). Both members will receive the same feedback. The file should have the following naming convention: **CPRG251\_A2\_Lastname of member 1\_Lastname of member 2.zip**

1. The ZIP file must contain the following:   
   1. The following directory structure:

* bin/ – Compiled Java files.
* src/ – Java source code files:
  + sait/bankonit/application/
  + sait/bankonit/gui/
* doc/ – Generated Javadoc files.
  + Ensure the *private* option is checked and everything is included in the generated documentation.
* lib/ – Any third-party libraries. This folder can be empty.
* res/ – Any resource or data files.
* test/ – Unit test cases. This folder can be empty.

1. A text file named **Readme.txt** in the root folder of the ZIP archive and contain:

* A project title.
* What the program does.
* The date.
* The author
* How to run the program.

1. A runnable JAR file in the root folder of the ZIP archive.
   1. Use the naming convention: **FirstInitialLastname2.jar** (i.e.: JBlow2.jar).
   2. It is to be built using only Eclipse IDE and JDK 1.8x.
2. You maybe required to provide a pre-recorded video demonstrating your completed assignment.

## **Assignment Instructions**

The goal of this assignment is to build a simple front-end that acts like an ATM (Automated Teller Machine). You may use the provided skeleton as a starting point for the login and account windows.

You are supplied with the backend (as a JAR file) as well as the documentation. The backend provides the following functionality:

* Loads accounts and transactions.
* Login using a card number and PIN.
* Perform a withdraw.
* Perform a deposit.
* Persist accounts and transactions to random access files.

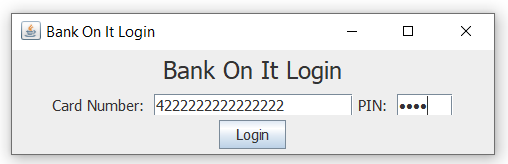
### **Backend Notes**

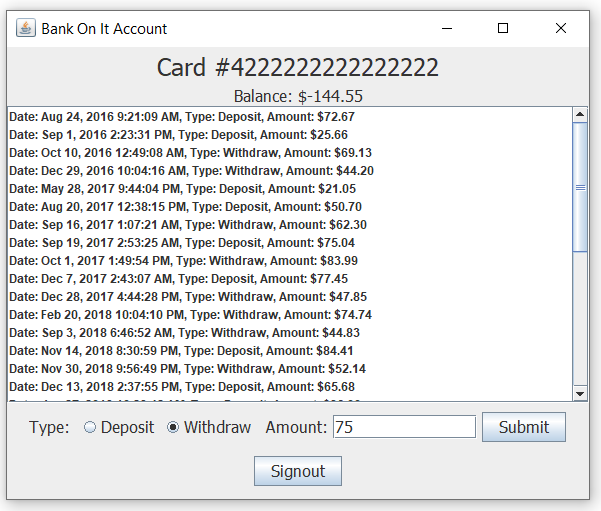
* It’s recommended you use the Singleton instance of the bank management system to avoid having multiple instances, file input/output errors, etc. The getInstance() static method in the ca.bankonit.manager.BankManagerBroker class provides this functionality.
* You will need to place the provided accounts.csv and transactions.csv files in the res/ folder of your Eclipse project.
* The provided JAR file should be placed in the lib/ folder of your Eclipse project.
* The transactions are sorted in ascending order by date/time.
* If a problem occurs with the backend, try deleting the random-access files (accounts.bin and transactions.bin in the res/ directory).
* The following card numbers and PINs are accepted by the backend:

|  |  |
| --- | --- |
| Card Number | PIN |
| 5555555555554444 | 1010 |
| 5105105105105100 | 9999 |
| 4111111111111111 | 4321 |
| 4012888888881881 | 1111 |
| 4222222222222222 | 1212 |
| 4444111122223333 | 4444 |
| 1234123412341234 | 1234 |

## **High Fidelity Mockups**

Use the provided high-fidelity designs to create the front-end:





## **Notes**

* Treat this as if it were a real-world project. Try to make your front-end as close to the mockups as possible.
* You may use radio buttons or a combo box for the transaction type. If using radio buttons, they will need to be added to a button group so only one can be selected.
* A list or table can be used to display transactions.
* The transaction list or table should be scrollable in case they overflow outside the size of the window. This can be done by wrapping the list/table in a JScrollPane.
* The transactions and balance should be updated after a deposit or withdraw is made.

## **Assignment Guidelines**

1. You will use only Eclipse IDE. You may use the WindowBuilder plug-in to help design the GUI.
2. If you wish to use JavaFx instead of Swing, please ask your instructor.
3. The due date for this assignment is posted in D2L in the assignment submission area and in the provided calendar located in the *Course Information* area**.** Any assignment submitted after the due date will receive a mark of zero, but feedback maybe given.
4. It’s important you start this assignment as soon as possible because it can be time consuming.
5. Submissions must be student’s original work. Refer to the Academic Misconduct (AC.3.4) policies and procedures.

## **Incremental Build Model**

This assignment can be completed using an incremental build model. In an incremental build, the system is designed and developed in small pieces in separate iterations. This makes it easier to build and test the entire program.

### **Version 1.0**

The account window is rendered with the transactions and current balance. The account that is logged in by the programmatically by you (the programmer). It’s recommended you come up with the user interface before adding any functionality.

### **Version 2.0**

A user can perform a withdraw and deposit in the account. The transactions, balance is updated, and form is reset if the amount is valid. If the amount is invalid (not a number, empty, etc.), an error is displayed.

### **Version 3.0**

The user signs out of the account. After signing out, the transactions are persisted to the random-access file. A message is displayed that says “Goodbye” then the account window is closed programmatically.

### **Version 4.0**

User can login with a valid card number and PIN. If the card number or PIN is invalid, a message box is shown (using JOptionPane). Otherwise, the account window is displayed with the correct balance and transactions.