COSC 2P05 Assignment 1

Winter 2015/16

Due: Feb. 8 @ 10:00 am (*Late Date: Feb. 11 @10:00 am*).

In preparation for this assignment, create a folder called `Assign_1` for the assignment. The objective of this assignment is to apply inheritance in a problem solution.

St. Kitts Bank

Part A

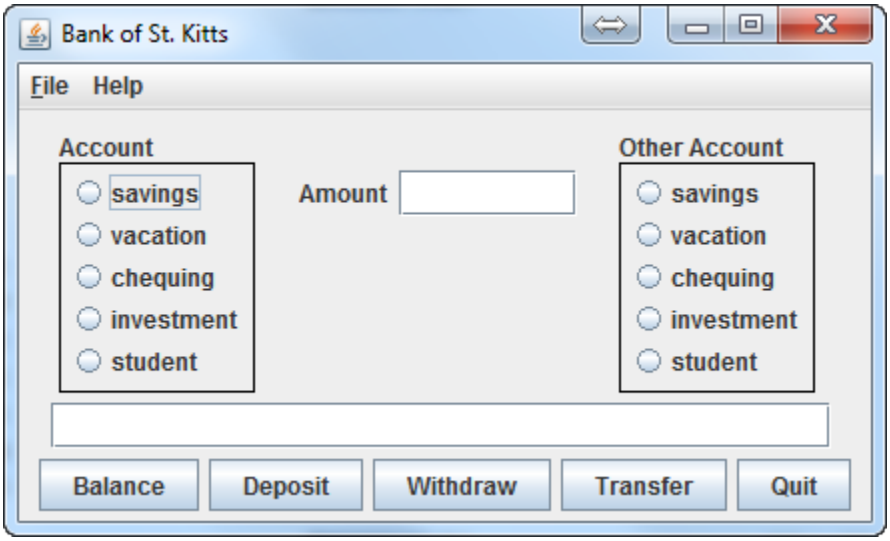
The Bank of St. Kitts is planning to install a new automated teller system whose software is to be written in Java. This assignment is to develop a prototype of the software that will run an ATM.

The bank has a number of clients. Each client has a client number, a personal identification number (PIN) and may hold a number (up to 5) of accounts that they may access using the ATM. An account has an account number, an account description and a balance.

When a client begins to use an ATM, they first insert their bank card. They are then prompted to enter their PIN. Once the PIN has been verified, they can perform transactions until they are finished.

For this prototype there will only be one client and the client won't log in (however there should still be a `Client` object that retains the information above.)

The client is presented with a form such as below. Note the names on the accounts are the account descriptions (a client could have a number of each kind of account such as below where `savings` and `vacation` are both savings accounts):



The possible transactions are: inquire balance, deposit, withdraw and transfer. For each type of transaction they select one (or two, in the case of a transfer—`Other Account` is only used for transfer) of their accounts and then (except for balance) they fill in the amount and press the button indicating their chosen transaction. If the transaction is acceptable, the transaction is performed. If the transaction is rejected an appropriate message is displayed in the text area

above the buttons and the form is presented with only one button labeled `OK`. For the `Balance` operation, the balance is displayed in the `Amount` field and the form is displayed with only one button (`OK`). In both cases, after OK is pressed, the form is presented with the four buttons for the next transaction. The program continues presenting the form for transactions until the user presses `Quit`.

In the production version, the ATM would then go back to the Enter Bank Card prompt.

When the program is about to shut down (i.e. after the user presses `Quit`), it should apply interest to each account simulating the end of day with the account interest computed daily.

Develop the appropriate classes to support such a system. The accounts should support the operations for deposit and withdrawal (a transfer is a withdrawal from one account and a deposit into another) and balance inquiry. The account should be able to reject a transaction if it considers it invalid in some way. At the current time (and this may change), transactions with negative amounts are rejected. Care should be taken to avoid the problem of one of the accounts rejecting the sub-operations of a transfer while the other accepts it. Neither operation should actually be performed if either account will reject it. This requires some sort of two-phase commit (i.e. each operation changing the balance would have a validate part (which records the validated amount) and a confirmation part (which applies the amount recorded).

The bank currently (and this may change) supports two types of accounts: savings and chequing as described below:

Savings: A savings account has a minimum balance of $1000.00 (based on balance before the transaction) or a charge of $0.50 is levied per withdrawal. A savings account may not be overdrawn and receives interest at the rate of 0.013% per day.

Chequing: A chequing account has a transaction has a fee of $1.00 for each withdrawal (regardless of balance). A chequing account pays no interest but may be overdrawn (in which case interest is charged at a rate of 0.018% per day).

Care should be used in the design of the class structure to support accounts. At the least, there should be a superclass (`Account`) which describes general properties of accounts and the savings and chequing account classes should be subclasses, inheriting as much as possible and overriding where necessary. The list of accounts held by a customer should be based on the `Account` class to provide polymorphism.

A file (`ASCIIIDataFile`) of the client information should be read by the progran at startup to initialize the client. This file contains data for only one client in the prototype, including: the client number (string), the PIN (string), the number of accounts (integer) and then, for each account, the account type (char, `C` for chequing, `S` for savings), the account number (string), account description (string) and the current balance (double). When the program shuts down, it should grant (or charge) interest to interest bearing accounts.

For auditing purposes, regulators require that the result of every transaction be logged to a log file (`ASCIIOutputFile`). Build this functionality into your program. A balance inquiry isn't logged but all other transactions are. The log entry for each transaction should include: the transaction type (e.g. `Deposit`), the amount, the account number, the prior balance (i.e. before the

transaction), the new balance (after the transaction) and an indication whether the transaction succeeded or was rejected. For a transfer the same information is required both for the "from" account and the "to" account. The log lines might look like:

```
Deposit 200.0 To: 1113 orig: 500.0      new: 700.0: Successful
Deposit 1000.0 To: 1111 orig: 800.0     new: 1800.0: Successful
Transfer 200.0 From: 1111 orig: 1800.0   To: 1113 orig: 700.0   From new: 1600.0 To new: 900.0: Successsful
```

Part B

News Flash

The Bank of St. Kitts announced that they have successfully completed a merger with Hamilton Dominion Bank. "Account holders should expect some exciting new developments in access and products" said B. Hogg, chairman of the bank. The merger is expected to involve a reduction in workforce as duplication between the two banks is reduced. Shares in the Bank of St. Kitts and Hamilton Dominion Bank both rose considerably in early trading today on the TSX.

As a result of the merger, the Bank of St. Kitts has introduced a number of different account types to suit different client needs. They must now upgrade their automated teller system to allow for these new account types. The following account types (with their rules) have been introduced:

Investment Savings: An investment savings account has a minimum balance of $2000 (based on balance before the transaction) or a charge of $1.00 is levied per withdrawal. Each transaction (deposit or withdrawal) must be at least $500.00 or it is rejected. An investment savings account may not be overdrawn and receives interest at the rate of 0.015% per day.

Student Savings: A student savings account has no charges, may not be overdrawn and receives interest at a rate of 0.013% per day.

Update your program to accommodate these new accounts. Strive for minimal changes in the existing classes.

The account type codes will be augmented to: `s` (savings), `c` (chequing), `i` (investment savings) and `x` (student savings). The rest of the client information file will remain the same.

Submission

Run your program using the script for transactions given in the file: `script.txt` and the client data file: `client.txt`. Your submission will be in two parts: electronic and paper. The paper submission should include the listings of all Java files you have written and the contents of the log file.

In addition to your paper submission, you must make an electronic submission of the assignment. You should have a folder for the assignment (`Assign_1`). This folder should include what is needed for the marker to run your program (as indicated below). Zip this folder as `Assign_1.zip`. Log on to Sakai and select the COSC 2P05 site. On the `Assignments` page select `Assignment 1`. Attach your .zip file (e.g. `Assign_1.zip`) to the assignment submission (use the `Add Attachments` button and select `Browse`. Navigate to where you stored your assignment and select the .zip file (e.g. `Assign_1.zip`). The file will be added to

your submission. Be sure to read and check the `Honor Pledge` checkbox. Press `Submit` to submit the assignment. You should receive a confirmation email.

The complete paper submission should be placed in an envelope to which a completed coversheet (`http://www.cosc.brocku.ca/coverpage`) is attached. The submission should be made to the COSC 2P05 submission box outside J328 in accordance with the assignment guidelines posted on the 2P05 Sakai site and not in violation of the regulations on plagiarism (`http://www.brocku.ca/webcal/2015/undergrad/areg.html#sec67`, `http://www.cosc.brocku.ca/about/policies/plagiarism`). **Note**: Assignments not including a coversheet will **not** be marked.

DrJava

The `.zip` folder you submit should contain the project folder including all files relevant to the project—the `.drjava`, `.java` and `.class` files for the assignment.

Other Platforms

If you are using an IDE other than DrJava to prepare your assignment, you must include the `.java` source files and the `.pdf` files described above as well as a file (likely `.class` or `.jar`) that will execute on the lab machines. It is your responsibility to ensure that the marker can execute your program.