**CSCE 855 Programming Assignment 3**

**Design Document**

Dongpu Jin

4/20/2013

The goal of this project is build a transaction-based banking system that supports multiple ATMs (clients) to operate on a single Bank server. The assumption made about this banking system is that one client can only access one account during each transaction. A client is not allowed to access a different account during the same transaction. In addition, other clients are blocked if they try to access the accounts that are being accessed. Correctness and consistency of these requirements are ensured by strict two-phase locking algorithm and private workspace algorithm. This banking system is implemented with Java RMI on Amazon EC2 services.

The client side is mainly responsible for interfacing with the customers, establishing the RMI connection, and issuing RPCs to the server. Most of the core algorithms are implemented on the server side. The server is designed with the Object Oriented concept. The bank class contains several inner classes including a class for account, a class for customer, and a class for private work space.

Each account object contains fields for account id, account balance, and a semaphore. Recall that semaphore is a type of data structure that provides mutual exclusion by acquiring and releasing permits. Each account initializes a semaphore to have just one permit to make sure that the account can be accessed by customers one at a time. The customer that first accesses an account acquires the permit of accessing the account. Other customers who also want to access this account are put into a waiting queue of the semaphore. Once the first customer finishes the transaction and releases the permit, the next customer waiting at the head of the queue can acquire the permit to access the account. Other customers remain blocked until their turn to acquire the permit. As a result, the strict two-phase locking is enforced such that only one customer is able to access a particular account within a single transaction. In addition, the FIFO ordering of waiting customers is also enforced by the semaphore.

The private work place object has fields to store account id and balance. It acts as a temporary sandbox for a customer's operations. Customers will first make a copy of the actual account to the private work space, and then perform all the subsequent operations on the private work space such that the actual account remains intact. If the customer aborts the transaction, the private work space is discarded and actual account is kept unchanged. If the customer commits the transaction, the changes made in the private work space are copied back to the actual account.

Each customer object contains fields for customer id, the id of the account this customer is accessing, and a private workspace object. Before performing an operation, a customer always checks if he is still accessing the same account. Operations will be canceled if accessing a different account. In addition, all the operations of a customer are performed on the private workspace object. The private workspace is not written to the actual account until customer commits this transaction.

The bank server maintains a list of currently active customers and a list of available accounts. Customers are added to the list when starts the transaction and removed from the list when either commit or abort the transaction. Accounts are created when a customer makes a deposit operation on a non-exist account.