***Title      Homework 7: Automated Teller Machine***

***Author     Kalpan Bhatt***

***Date         01/012/2012***

***Overview***

**Purpose**

Develop the back end of a distributed Automated Teller Machine (ATM) system using basic Java networking.

**Brief description**

We are developing a distributed Automated Teller Machine (ATM). The ATM will be hosted in a different process than the client application. The client will connect to the server using Java networking and communicate with the server using a protocol.

**Functional Requirements**

The banking system consists of an ATM, a Bank, Accounts and a Security service

The ATM will support the following operations:

deposit: add some dollar amount to a specified account's balance

withdrawal: deduct some dollar amount from a specified account's balance

balance inquiry: get current balance of a specified account

transfer: deduct some dollar amount from a specified account and deposit that amount into another specified account

**Design**

ATM interface will be implemented by both the real ATM and a client side proxy for the ATM, called a stub. The client will use a stub that will proxy all the operations of the ATM and delegate them an implementation object on the server.

The ATM will now include several individual Accounts and each of the ATM methods must now include a parameter that allows the account number to be specified. For simplicity we'll assume the account identifier type is int. In this system, we will create an ATM factory that the client will use to get a reference to a remote ATM.

The ATM factory is a server side object with a remote interface just like the ATM. We will create an ATMFactory interface that has a single method getATM() that returns a remote reference to the serverside ATM instance.

The server process will start up, create an ATMFactoryImpl instance, and then register it with the RMI naming service or registry using java.rmi.Naming. The client will then be able to lookup the ATM factory and connect to it. Then the client will use the getATM() method of the factory to get a remote reference to an ATM instance.

**Goals**

1. All classes should be in package cscie160.project
2. Enhance ATM interface to be an RMI remote interface.
3. Create an RMI remote implementation ATMImpl that implements the remote ATM interface.
4. Create an Account class. It will hold information about its balance.
5. The Security component is a remote object that has methods for authenticating AccountInfo objects and for authorizing specific operations on individual Accounts.
6. The AccountInfo object is a data class that includes an account number and a PIN. Instances are passed by value between processes.
7. Create an ATMFactory interface that is RMI ready and supports a single getATM() method.
8. The TransactionNotification object is a data class that includes pertinent information about a particular transaction including target accounts and amounts.
9. The ATMListener interface includes a method to receive TransactionNotification messages from the ATM.
10. Create the RMI ready implementation of the factory interface. It should return a remote reference to an ATM instance.
11. Create a Bankserver, ATMServer and Client class.
12. Client class should have main method for the demonstration.

**See Also**

[**http://courses.dce.harvard.edu/~cscie160/project.htm**](http://courses.dce.harvard.edu/~cscie160/project.htm)

**Procedure**

1. The BankServer process creates two remote objects and registers them for RMI access: a Bank object and a Security object.
2. When the Bank is created it creates some Account objects. These Accounts are remote objects, but they are not RMI registered. Remote access to an Account object is through a Bank.
3. The ATMServer process creates an ATMFactory and registers it for RMI access. The ATMFactory creates ATM objects on request. Upon creation the implementation ATM object looks up the Bank in the rmiregistry.
4. The ATM gets a reference to an Account through a remote call to the Bank, after checking with Security. The Client never gets direct access to an Account.
5. The Client process calls to the rmiregistry to get a reference to the ATMFactory. It is that object that creates the ATM and returns a reference to it for the Client to use.

**How to run the assignment**

*Unzip the contents of the jar file Project.jar*

*Run the Ant tool at the folder level of build.xml*

*To demonstrate your system you will need to start three distinct processes.*

*First startup the rmi registry:*

*C:\> rmiregistry*

*Then startup the Bank server and then ATM server:*

*C:\> java cscie160.project.BankServer*

*C:\> java cscie160.project.ATMServer*

*Once the servers are up and running, start up the client:*

*C:\> java cscie160.project.Client*

*You should see the output as mentioned in the output.txt file*