Banking

By: Aaron Garten, Huan Le, Dolu Odumosu

What is it?

- Multithreaded Server with ATM and Teller Client which interact with a local database to represent a Bank System
    - ATM - Limited Client, assumed number pad
    - Teller - Full keyboard Client
    - Server - Only Console messages

ATM

- ATMGUI
  - This is where the main class lies that drives the entire ATM module.
  - This is what prompts the ATM User for necessary variables that will be passed to the ATM for operations to be carried out.
  - In my own implementation this is also where the Output Streams, Socket and Input Streams are declared and initialized.
  - The ATM GUI class also passes these object output streams to the ATM.
- ATM
  - The ATM class receives important variables that will be accessed in the ATM class and sent to the server for feedback.
  - These variables may include Object input and output streams or strings and integers for login or other operations
  - This class is where messages are sent to the server and is received from the server.
  - It is where all the ATM operations are done.

Teller

- TellerGUI with ActionListener for User’s inputs
- Input messages will be sent to server
- Server will process messages and send back success or failure
- Proceed with success messages

Server

- Multithreaded
- “offline” (but still listens) if error loading data from file
- DataBase
    - loaded from csv file
    - ArrayLists: Employee, Customers
    - HashMap lookup tables: login to password - Employee, card to customer id - ATM
    - all structures Synchrnized to prevent race conditions (except card to customer ID, external system)

Message Solution

- Message Parent Class
    - Child Message for every operation (ATMLogin, Logout, ATMTransfer, TellerLogin, TelleTransfer, ..etc)
    - Different messages can carry different type of objects (Customer, Account, Employee)
    - Always send receive Message, then cast to correct child
- Each message has enum Process for switch inside Server
- Some child Messages provide encapsulation via their constructors i.e. pass in Customer / Account but Message only holds id for either
    - Multiple nested classes Customer -> Account -> LastTransaction & Fee

Junit & Lessons Learned

Unimplemented Functions

- Supervisor’s actions
- Add/Remove anything (Customer, Account, Employee)
    - messages/methods available in DataBase and Teller but lacking Server implementation
- Online/Shutdown Server
    - messages but no Server implementation.
- Employee file loading
    - Currently hard coded in DataBase
    - Object implemented but untested

Unimplemented Tests

- Tests for unimplemented functions
- Tests for functions which required server communications
  - Unable to access objects and send over to server
- Any file handling, including the DataBase
- Any Server required communication
- Many Message class constructor Junits did not get implemented for time

Demo