CT414 Assignment 1

James Quaife – 14100104

Andrew East – 16280042

4BCT

8 February 2019

Distributed Banking Application using RMI

# Methodology

## Application Setup

This Java Remote Method Invocation (RMI) application was created as two distinct components, client and server, linked only over the abstracted network communication provided by RMI. A set of Interfaces are compiled into a single shared JAR, which must be made available to both client and server at compile time. Conversely, the implementation for these interfaces need only be provided at run time, through the codebase mechanism of RMI. The server is compiled into a JAR, and the bulk of its code is kept secure on the server. The shared implementation of its interfaces, such as Statement, are served up as a codebase that the client can read. Similarly, the client has its own JAR and shared implementations.

Maven is used to tie all compilation together, automating the multiple-JAR creation. Maven is also used to run unit tests on several classes, with mocks of server objects used for the client tests. The application setup was completed by writing several shells scripts to run all components of the app. Two scripts exist to run the registry and server in the background, both configured to connect to the registry over port 7777. The third script to run the client is pre-configured with the same connection details, and the script additionally accepts arguments to perform ATM operations. These scripts do not manage the lifecycle of the registry or server, and the user must stop them when finished. They have been written in both PowerShell and Bash versions.

## Testing Methodology

During a planning session, a variety of error and non-error scenarios were identified that will provide coverage for normal and edge cases. These both guided what input validation was required and provided motivation for executing tests. To automate the process, a shell script was written that runs these scenarios in sequence. The output of this script can be seen below, along with the corresponding server logging.

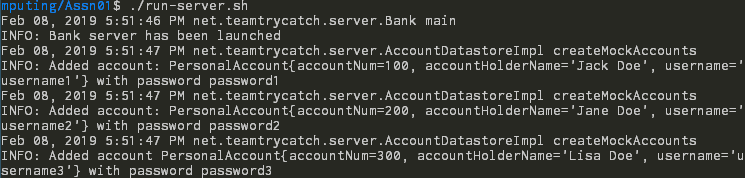
# Repository Link

<https://github.com/reideast/distributed-banking-system>

GitHub was used to manage remote development workflow between the authors of this project, working on different git branches and integrating work through pull requests. The online repository also shows a readme with detailed running instruction.

# Testing Results

## Server Initialisation



## Operation Before Login Fails

### ATM Output

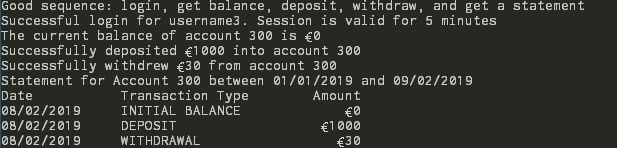


### Server Logs

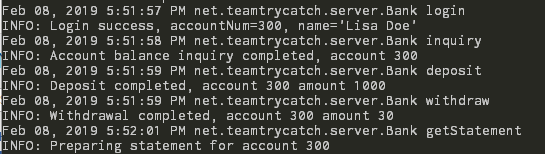


## Normal Sequence of Withdrawals, Deposits, and Statements

### ATM Output

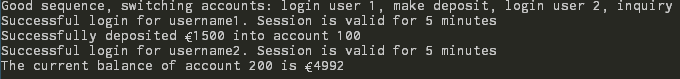


### Server Logs

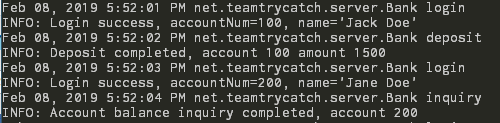


## Normal Sequence Switching Between Two Users

### ATM Output



### Server Logs

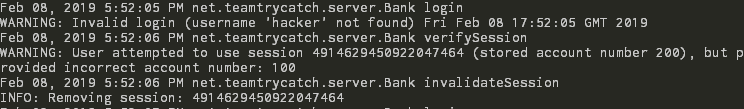


## Invalid Login Prevents Any Operations

### ATM Output

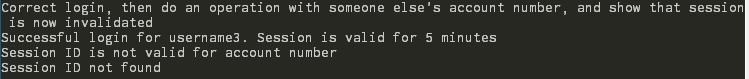


### Server Logs

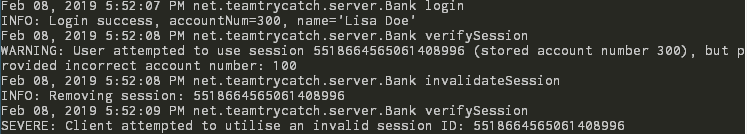


## Incorrect Account Number for Current Login

### ATM Output

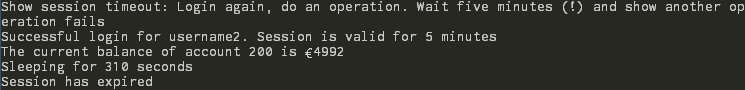


### Server Logs

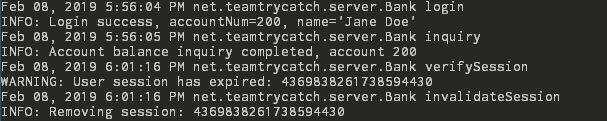


## Session Times Out and Prevents Further Operations

### ATM Output



### Server Logs





# Appendix: Planning Diagram

