CISC 327 FRONT END REQUIREMENTS TESTING

(Cloud Programming)

Ly Sung 10150760
Tyler Mizuyabu 10141746
Calvin Che 10158050

Table of Contents

Assignment Changes:

As a group decision, we have decided to switch from Windows to Linux since we wanted to automate the test cases by using bash. But just to clarify we are still using Java as our main programming language.

Additionally, we have added two more test cases to fix some of the requirement errors from assignment1 as stated by the TA. These two test cases are inserted as test #22 and test #23 in order to maintain the quality of the test such that the next test assumes that the previous has been tested. This means that all test cases starting from number 22 originally in asignment1 are shifted two down (Original test #22 is now test #24 and so on).

Another side note text documents need to be opened on sublime because when opened on notepad new line character is made. As a result when comparing two identical files, they would not be matched because of the new '\n'.

Directory Structure

/327project
- Agent.java
- Atm.java
- FrontEnd.java
- Session.java
- createExec.sh
- MANIFEST.MF
- Simbank.exe
- Stub.sh
- **/testsuite**
  - **/front-end**
    - **/cout**
    - **/expected**
    - **/expected-cout**
    - **/input**
    - **/output**
    - console.txt
    - Errors.txt
    - Summary.txt
    - testScript.sh
    - Validaccounts.txt

Important Files

createExec.sh - creates the execuable called simbank
/cout - this directory contains all the console output files when executing the program
/expected-cout - this directory contains all the expected console output files
/input - this directory contains all the test case inputs files
/output - this directory contains all the test output (transaction summary) files
/expected - this directory contains all the test expected output files
Console.txt - temporary stores the console output of each test case
Summary.txt - temporary stores the test output (transaction summary) of each test case
Errors.txt - contains all the test cases that did not have a matching expected output or console expected output
validaccounts.txt - contains all the valid accounts
testScript.sh - automates each test input file and checks that the output/console output file matches its expected output/console output file

How to run the executable

Since the executable has been created, proceed to **/327project/testsuit/frontend** and call testScript.sh to execute the program. testScript.sh will call simbank and supply validaccounts.txt, summary.txt as well as the input file. In addition testScript.sh will automate all the testcases in **/input** and will compare the result of each test case to its expected output in **/expected**, and that the console output matches its expected console output in **/expected-cout**. If during the comparison the output/console output file is different from its expected output/console output file, the script will save the test name into **errors.txt**.

If testScript.sh does not run, proceed to **/327project** and call createExec.sh again. This will ensure that the executable is properly built for the current platform.

Test Case Number Failure	Purpose	Input file	Expected Output file (transaction summary file)	Actual Output File	Error in code/Input file	How we fixed it
19	Checks that new account name must be between 3-30 alphanumeric characters, and doesn't start or end with white spaces	login agent create 98765432 ab create 98765432 thisnameismorethanthirtycharacters create 97865432 hasspace create 97865432 hasspaceatend logout	ES 00000000 00000000 ***	CR 98765432 00000000 000 hasspace ES 00000000 00000000 000 ***	We did not have empty space handler in our Agent class for names that start or end with an empty space.	Added: !(args[1].startsWith(" ") || args[1].charAt(args[1].length() -1) == ' ') In addition to checking that the length of the name is valid, we extended the if statement to make sure that no names can begin or end with an empty space.
34	Checks that withdraw doesn't have a limit in agent sessions (agent)	login agent withdraw 12345678 50000 withdraw 12345678 60000 logout	WD 12345678 00000000 50000 *** WD 12345678 00000000 60000 *** ES 00000000 00000000 000 ***	WD 12345678 00000000 50000 *** ES 00000000 00000000 000 ***	For this particular error, when we tried to run the same transaction on the same account twice, we would get an invalid account number. In our FrontEnd class we missed curly brackets sounding an if statement that checks if an account is valid.	We simply fixed the problem by surrounding line 298 to 306 with the curly brackets.
30 & 32 & 38	Checks that an Agent cannot withdraw/deposit/transfer an amount greater than 99999999	login agent deposit 12345678 -1 deposit 12345678 10001 logout --- simnliar format for case 32 and 38	ES 00000000 00000000 000 ***	WD 12345678 00000000 10001 *** ES 00000000 00000000 000 ***	There was originally an error in Assignment1 where we did not insert the withdrawn account to be greater than 99999999 to check that this amount would exceed the maximum withdrawn amount an Agent can perform.	We fixed the withdrawn amount in test30.txt , test32.txt, and test38.txt to be 10000000001