QUINTERAC

Loaft Inc.

December 7, 2019

Samir Mekhdi – 10191295

Luvit Chumber – 10190467

Jessica Wong – 10181646

Tayyab Ahmad – 10197212

1) Source Listing:

CISC327-LoaftInc

|--QUINTERAC/Testing/

| |---integration/integrationTest.java

| |---backend/backendTest.java

| |---frontend/frontendTest.java

|--QUINTERAC/src/backend/

| |---backend.java

| |---Account.java

|--QUINTERAC/src/frontend/

| |---frontend/frontend.java

| |---frontend/Terminal.java

|--QUINTERAC/src/transactionClasses/

| |---CreateAcct.java

| |---DeleteAcct.java

| |---Deposit.java

| |---Login.java

| |---Logout.java

| |---Transaction.java

| |---Transfer.java

| |---Withdraw.java

Daily scripts:

Both daily and weekly scripts were implemented via integrationTest.java, which can be found in the testing directory. 5 days of transactions were created, with each set running 3 separate transaction sessions (frontends). The loop runs 5 of these daily scripts in succession. The daily commands are named as t_in_day1, t_in_day2, etc. This structure contains a list of strings to be passed in as terminal input. The contents of each daily test are as follows:

- t_in_day1:
  - Create accounts
- t_in_day2:
  - Deposit into these accounts
- t_in_day3:
  - Transfer between these accounts
- t_in_day4:
  - Withdraw from these accounts
- t_in_day5:
  - Delete accounts
  - Withdraw accounts
  - Transfer

```
String t_in_day4[][] = {
    { "login agent",          //session 1
      "withdraw 9999999 8180000",
      "logout"},
    { "login agent",                  //session 2
      "withdraw 1000001 101100",
      "withdraw 5000001 400000",
      "withdraw 1000003 61820",
      "withdraw 1000002 11500",
      "withdraw 1000000 98000",
      "withdraw 7777777 7900000",
      "logout"},
    { "login agent",                  //session 3
      "withdraw 3030303 1064000",
      "withdraw 8300781 107000",
      "withdraw 1000000 150000",
      "logout"} };                   //END DAY 4

String t_in_day5[][] = {
    { "login agent",            //session 1
      "deleteacct 5000001 JaneDee69",
      "transfer 9999999 1000000 100",
      "transfer 9999999 1000001 100",
      "withdraw 9999999 1349600",
      "logout"},
    { "login agent",                      //session 2
      "deleteacct 7777777 BethanyWhite212",
      "transfer 9999999 1000003 100",
      "deleteacct 3030303 Walter101Grey",
      "transfer 9999999 1000002 100",
      "logout"},
    { "login agent",                  //session 3
      "deleteacct 8300781 747JonathanBlue",
      "deleteacct 9999999 JohnDoe420",
      "logout"} };                   //END DAY 5

String t_in[][][] = {t_in_day1,t_in_day2,t_in_day3,t_in_day4,t_in_day5};
```

Figure 1. More In-depth Sample of Commands and Structure

2) Daily script inputs:

Day 2:

> Transactions being issued:
>> FrontEnd #1
login agent
deposit 1000000 50000
deposit 1000001 100
deposit 9999999 200000
logout
>> FrontEnd #2
login agent
deposit 9999999 9500000
deposit 5000001 320
deposit 8300781 100000
deposit 3030303 671000
deposit 1000002 500
logout
>> FrontEnd #3
login agent
deposit 7777777 8000000
deposit 3030303 500000
deposit 1000003 1500
deposit 5000001 400000
logout

3) Corresponding Merged TSF:

Day 2:

DEP 1000000 50000 0000000 ***
DEP 1000001 100 0000000 ***
DEP 9999999 200000 0000000 ***
EOS
DEP 9999999 9500000 0000000 ***
DEP 5000001 320 0000000 ***
DEP 8300781 100000 0000000 ***
DEP 3030303 671000 0000000 ***
DEP 1000002 500 0000000 ***
EOS
DEP 7777777 8000000 0000000 ***
DEP 3030303 500000 0000000 ***
DEP 1000003 1500 0000000 ***
DEP 5000001 400000 0000000 ***
EOS
EOS

4) Master Accounts File:

Day 0:

// empty file

Day 1:

9999999 0 JohnDoe420
8300781 0 747JonathanBlue
7777777 0 BethanyWhite212
5000001 0 JaneDee69
3030303 0 Walter101Grey
1000003 0 JessicaW
1000002 0 SamirM
1000001 0 LuvitC
1000000 0 TayyabA

Day 2:

9999999 9700000 JohnDoe420
8300781 100000 747JonathanBlue
7777777 8000000 BethanyWhite212
5000001 400320 JaneDee69
3030303 1171000 Walter101Grey
1000003 1500 JessicaW
1000002 500 SamirM
1000001 100 LuvitC
1000000 50000 TayyabA

Day 3:

9999999 9530000 JohnDoe420
8300781 107000 747JonathanBlue
7777777 7900000 BethanyWhite212
5000001 400000 JaneDee69
3030303 1064000 Walter101Grey
1000003 61820 JessicaW
1000002 11500 SamirM
1000001 101100 LuvitC
1000000 248000 TayyabA

Day 4:

9999999 1350000 JohnDoe420
8300781 0 747JonathanBlue
7777777 0 BethanyWhite212
5000001 0 JaneDee69
3030303 0 Walter101Grey
1000003 0 JessicaW
1000002 0 SamirM
1000001 0 LuvitC
1000000 0 TayyabA

Day 5:

1000003 100 JessicaW
1000002 100 SamirM
1000001 100 LuvitC
1000000 100 TayyabA

5) Integration Defect Report:

The following errors were found during testing, with an explanation of how it was fixed and updated.

Detected Defects	Solution

Lowercase convert: Converted input to lowercase in frontend before splitting for transaction code but affected name as well.
 	
Before:
System.out.println("Enter next transaction: ");
inputRaw = in.nextLine().toLowerCase();
input = inputRaw.split(" ");
	Updated the convert case to be done after splitting the name and the trans code, essentially changing the order of operations.
	
	After:
	System.out.println("Enter next transaction: ");
	inputRaw = in.nextLine();
	input = inputRaw.split(" ");
	input[0] = input[0].toLowerCase();

String split: inputs were not being split correctly while validating master accounts containing a new account. In the case where a name had a number in it, the string would split at these points if the number was the same as the amount in the account (0)
 	
Before:
String[] in = inRaw.split(" "); // cannot use this method for names, names can have spaces
String type = in[0];
String acctTo = in[1];
String amountStr = in[2];
String acctFrom = in[3];
int startIDX = inRaw.lastIndexOf(acctFrom);
String name = inRaw.substring(startIDX+acctFrom.length());
name = name.trim(); // removes space from front of string;
	Since the format of the input was known, we accommodated this by looking for a more specific match, in this case a " " (space) before and after the amount field
	
	After:
	String[] in = inRaw.split(" "); // cannot use this method for names, names can have spaces
	String type = in[0];
	String acctTo = in[1];
	String amountStr = in[2];
	String acctFrom = in[3];
	int startIDX = inRaw.lastIndexOf(" " + acctFrom + " ") + 1;
	String name = inRaw.substring(startIDX+acctFrom.length());
	name = name.trim(); // removes space from front of string;

Account Balance cannot be 0: Account information was flagged as incorrect during the validation step for master accounts due to the account balance being 0 (which all new accounts have). Initially the code checked for numbers that are at least 3 digits, no greater than 99999999, and not a negative number Before: if ((in.length() < 3) || num > 99999999 || num < 0)	Checks that the account balance is under the same conditions as previously written except for a value of 0 After: if ((in.length() < 3 && num != 0) || num > 99999999 || num < 0)
Withdraw amount from account: User is not allowed to withdraw the exact balance of the account Before: if (amount < currentAmount)	Allow the user to withdraw the amount in the account or less After: if (amount <= currentAmount)