# Back End Testing Methodology and Report for New Account Transactions

For the back-end testing on assignment 5 we are to practice partial white box testing of the Back-Office portion of Quinterac we programmed in Assignment #4.

For the Testing of our back Office we implemented white box testing for the new account creation transactions as well as the withdrawal transactions. For these two sections of the program we are to use two separate white box testing methodologies in order to test out program. For the new account transactions, we chose to perform the white box testing methodology of Path coverage. For the withdrawal transactions, we chose to perform the white box testing methodology of decision coverage. Both of our testing procedures only covered the section of code needed to perform these actions.

Down below outlines the breakdown of how we derived our test cases we have to write in order to perform the tests for the New Account transaction. For our test cases we are only testing the method/ section of code that performs the actual transaction and verifies that these sections of code are actually functioning as intended.

The two main files we are testing for this section of our code are BackendObj (in specific the processTrans() as well as processNEW() methods inside this file) and the MasterFileObj file (in specific the addToHash() method in this file).

**Creation of NEW Account (Path Coverage)**

- Line that contains NEW at the start
    - Account **IS** already in HASH
    - Account **IS NOT** already in HASH
- Line that contains DOG at the start
    - Goes Nowhere (throws exception)

Since we are performing path coverage for just our method/section of code in order to create a new account in the master accounts file we only need 3 test cases as that is the total number of distinct paths we can take. We have arrived at this number as in the outline above we have 3 different scenarios our code can run through when creating a new account to add to the newly created account to the Master Accounts File. In the section of code that we use to perform the action of creating a new account we have two decision statements that the code can run through in order to perform the task. The first decision statement our code runs through is a case statement which determines if the transaction code of the line in the TransactionSummaryFile contains NEW or if it contains something other than NEW. The Second decision statement out code will run through is when the line of the TransactionSummaryFile does contain NEW and this statement checks if the account that is intended to be added to the MasterAccountsFile is already in the File. These two decision statements that occur cause our code to have three distinct paths that it can go through when adding a new account to out MasterAccountsFile.

**All of our testing files for the back Office can be found at the following directory path in our repository on GitHub:**

**\Quinterac\Assignment_4_5\src\test\resources\R4**

## Chart of Test Cases for New Account Creation

| Requirement | Test Num | Test Name | Purpose | Input Files | Output | Output Files |
|---|---|---|---|---|---|---|
| Test the Functionality of the **New Account Creation** transaction<br><br>Performed **PATH** coverage on this portion of our Back-Office Program | back R1T1 | Creat eT1 | Working with account that already exists | BackR1T1Master.txt TransactionSummayrFiles/ backR1T1/ TransactionSummary.txt | Error during processNEW | backR1T1Expected.txt |
| | back R1T2 | Creat eT2 | Working with account that does not exist | BackR1T2Master.txt TransactionSummaryFiles/ backR1T2/ TransactionSummary.txt | Done | backR1T2Expected.txt |
| | back R1T3 | Creat eT3 | Using an unknows transaction code in the transactionSummary File | BackR1T3Master.txt TransactionSummayrFiles/ backR1T3/ TransactionSummary.txt | Unreadable line encountered in merged transaction file. | backR1T3Expected.txt |

## Test and Failure Report

| Test Number | Issue | Resolution |
|---|---|---|
| backR1T1 | [PROGRAM] Null pointer exception generated when feeding in a non-empty master accounts file | [Solved] Moved instantiation of allAccounts to a point before it is needed |
| | [PROGRAM] Another null pointer exception generated when feeding in a non-empty master accounts file | [Solved] Moved instantiation of allAccNums to a point before it is called |
| | [PROGRAM] Merged transaction file tagged EOS onto the end of the first line causing issues reading the file | [Solved] included a new line character where it was omitted previously |
| | [PROGRAM] Generates new master accounts file with without any new lines | [Solved] Added new line character to the file output |
| backR1T2 | [PROGRAM] Generated master file was missing new entries | [Solved] Accounts that were being created were not properly being added to the account list |
| | [PROGRAM] Generated master file was including an extra line at the end of the file | [Solved] improved logic for printing new lines so that they will only be printed if the account list is not empty |