CMPE327: Software Quality Assurance

Assignment # 5: Back Office Unit Testing

Stuart Bourne (10091057) | Patrick Dang-Ho (10100190) | Matthew Sims (10099516) | Brent Lommen (10142201) |

**The test case inputs are text files in the folder named Create and Withdraw Tests with file format {test\_case\_name}\_MTSF.txt.**

**And**

**{test\_case\_name}\_MAF.txt.**

**(e.g. testCreate\_invalidAccountName\_in.txt)**

# Create Transaction – Decision Coverage Training

## Code to be covered

...} **else if** (parsedInput[0].equals(**"CR"**)) {  
 *// Create* String accountNo = parsedInput[1];  
 String accountName = parsedInput[4];  
 **if** (*validateAccountNo*(accountNo) && *validateAccountName*(accountName)) {  
 *processCreate*(Integer.*parseInt*(accountNo), accountName);  
 } **else** {  
 System.***out***.println(String.*format*(**"Invalid transaction entry: %s"**, line));  
 }  
}...

*/\*\*  
 \* Processes a create transaction.  
 \*  
 \** ***@param accountNo*** *Account number of the account to create.  
 \** ***@param accountName*** *Account name of the account to create.  
 \*/***private static void** processCreate(**int** accountNo, String accountName) {  
 **if** (!*accountsMap*.containsKey(accountNo)) {  
 *accountsMap*.put(accountNo, **new** AccountInfo(0, accountName));  
 } **else** {  
 System.***out***.println(  
 String.*format*(**"Creation failed: account with account number %d already exists"**, accountNo));  
 }  
}

## Description

We have decided to use decision/condition coverage testing for the create transaction. We analyzed the code and determined how many major (i.e. relevant to the create transaction) conditions/decisions are encountered, highlighted above. We concluded that only three test cases were needed to cover all the decision statements. We decided to add another test case to ensure that the first decision gets full coverage. We found this appropriate because this is condition coverage, which is a parent of decision coverage. Although all the test cases passed successfully, these test cases would not be able to validate the proper handling of erroneous input.

## Test Cases

MAF = Master Accounts File | MTSF = Merged Transaction Summary File

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Expected Outputs** | **Pass/Fail** |
| testCreate\_invalidAccountName | Invalid account name  Invalid transaction entry: CR 10101010 000 000 me | Pass |
| testCreate\_invalidAccountNo | Invalid account number: 1010010.  Invalid transaction entry: CR 1010010 000 000 test | Pass |
| testCreate\_accountExists | Creation failed: account with account number 10091057 already exists | Pass |
| testCreate\_happyPath |  | Pass |

# Withdraw Transaction – Path Coverage Testing

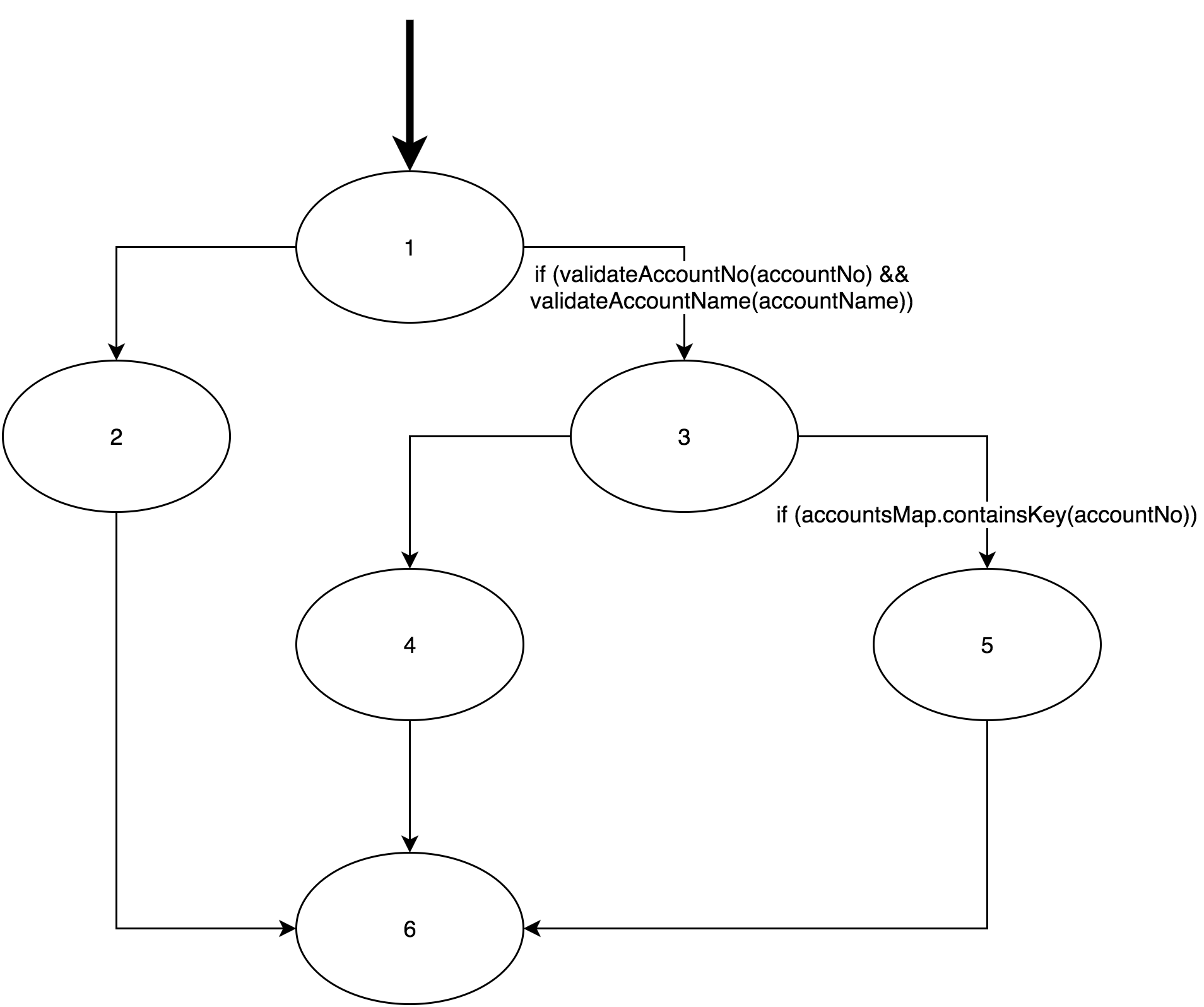
## Code to be covered

...} **else if** (parsedInput[0].equals(**"WD"**)) {  
 *// Withdraw* String accountNo = parsedInput[1];  
 String withdrawalAmount = parsedInput[3];  
 **if** (*validateAccountNo*(accountNo) && *validateTransactionMoneyAmount*(withdrawalAmount)) {  
 *processWithdraw*(Integer.*parseInt*(accountNo), Integer.*parseInt*(withdrawalAmount));  
 } **else** {  
 System.***out***.println(String.*format*(**"Invalid transaction entry: %s"**, line));  
 }  
}...

*/\*\*  
 \* Processes a withdraw transaction.  
 \*  
 \** ***@param accountNo*** *Account number to withdraw from  
 \** ***@param withdrawAmount*** *Amoount to withdraw  
 \*/***private static void** processWithdraw(**int** accountNo, **int** withdrawAmount){  
 AccountInfo accountInfo = *accountsMap*.get(accountNo);  
 **if** (accountInfo != **null** && accountInfo.isTransactionAllowed(withdrawAmount)) {  
 accountInfo.adjustBalance(-withdrawAmount);  
 } **else** {  
 System.***out***.println(String.*format*(**"Withdrawal failed: %d has insufficient funds."**, accountNo));  
 }  
}

## Description

We have decided to use path testing for the withdraw transaction. After analyzing the code for a withdraw transaction, we drew a flow chart of the path, which can be seen below. As shown in the diagram below, it was determined that only 3 independent paths were present, requiring only 3 test cases. Although all the tests pass successfully, these test cases would not be able to validate the proper handling of erroneous input.



## Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Expected Outputs** | **Pass/Fail** |
| testWithdraw\_invalidAccountNo\_invalidWithdrawAmount | Invalid account number: 00000000.  Invalid transaction entry: WD 00000000 20000000 -1 MattSims | Pass |
| testWithdraw\_notEnoughMoneyInAccountBalance | Withdrawal failed: 20000000 has insufficient funds. | Pass |
| testWithdraw\_happyPath |  | Pass |