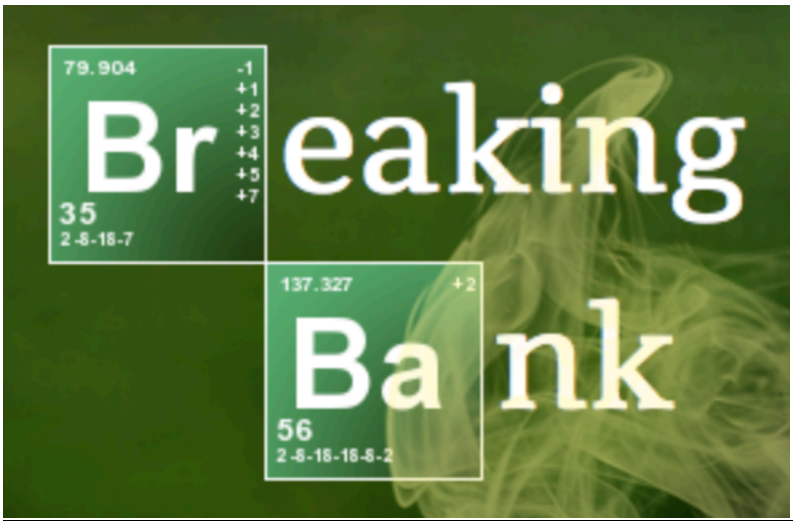


CISC 327 Assignment 4
Back End Prototype

Scott Wallace 10051890
Brad Guner 10059112

Inputs and Outputs

Inputs:

- **Old Master Accounts File**
- **Merged Transactions Summary File**

Outputs:

- **New Master Accounts File**
- **New Valid Accounts File**

Back-end Methods

Main Program

Our main program is designed to handle a few different operations in our back end. Its first task is to open the input files and put them into usable a data structure for the back end to edit. The 2 input files are the old master accounts file and the merged transactions summary file. Both input files will be put inside a list, to make editing and looping throw them easier. The next task for our main program to perform is looping through the merged transactions summary file's list. Since each index of the list will be a transaction, we will use it and call our transaction() method. This will loop for every value in the list and thus updating the master accounts file for every transaction performed. The last thing our main program does is it will call our 2 file writing methods to produce our 2 output files.

transaction()

The transaction method in the back end, takes in a transaction string from the main program and uses the information to update the old master accounts file list with the transaction information. If at anytime there is an error with the input, the method will just throw the ThrowError() method to handle it.

When creating a new account, the transaction method will add the account in the correct order in the master accounts list.

WriteNewValidAccounts()

This function gets called at the end of the main program. It will take the master accounts file list and iterate through while creating the new validaccounts.txt file.

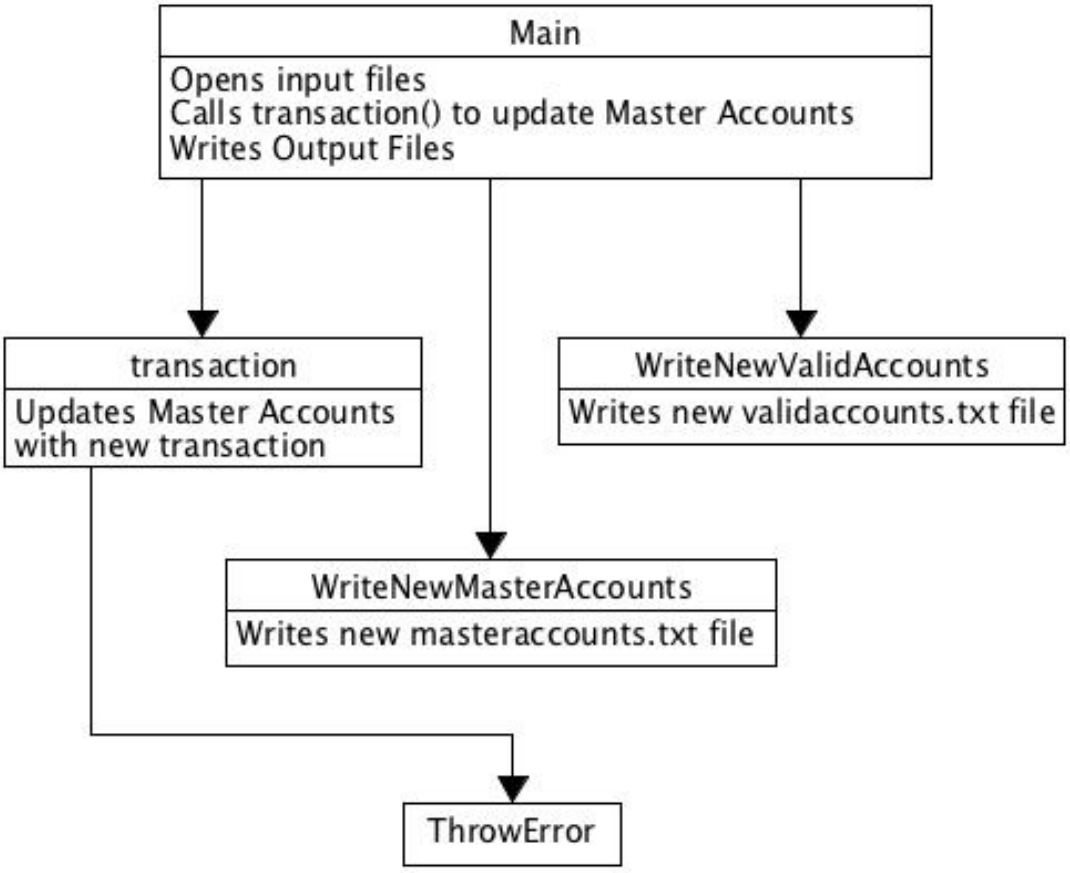
WriteNewMasterAccounts()

This function will take the master accounts list from main and use it to create the new updated master accounts file.

ThrowError()

This is a simple error handling method for the back end.

UML Diagram



Breakingbank-backend.py

```
import sys

def transaction(masterAccts,trans):
	#take trans, split by _ into list
	transCopy = trans.split('_') #[CC, AAAAAA, BBBBBB, MMMMMMMM, NNNNNNNNNNNNNN]
	master = []
	for i in range(len(masterAccts)):
		master.append(masterAccts[i])
	for i in range(len(master)):
		master[i] = master[i].split('_')

	if (transCopy[0] == '01'):	#deposit
		for acct in range(len(masterAccts)):
			if (master[acct][0] == transCopy[1]):
				acctBalance = int(master[acct][1])
				depAmount = int(transCopy[3])
				acctBalance += depAmount
				master[acct][1] = str(master[acct][1])
				master[acct][1] = str(acctBalance)
				newStr = format(master[acct][0], master[acct][1], master[acct][2])
				masterAccts[acct] = newStr
		return masterAccts

	elif (transCopy[0] == '02'):	#withdraw
		for acct in range(len(masterAccts)):
			if (master[acct][0] == transCopy[1]):
				acctBalance = int(master[acct][1])
				depAmount = int(transCopy[3])
				acctBalance -= depAmount
				master[acct][1] = str(master[acct][1])
				master[acct][1] = str(acctBalance)
				newStr = format(master[acct][0], master[acct][1], master[acct][2])
				masterAccts[acct] = newStr
		return masterAccts

	elif (transCopy[0] == '03'):	#transfer
		for acct in range(len(masterAccts)):
			if (master[acct][0] == transCopy[1]):
				for anotherAcct in range(len(masterAccts)):
					if (master[anotherAcct][0] == transCopy[2]):
						recAcctBalance = int(master[acct][1])
						transAcctBalance = int(master[anotherAcct][1])
						transAmt = int(transCopy[3])
						recAcctBalance += transAmt
						transAcctBalance -= transAmt
						master[acct][1] = str(master[acct][1])
						master[anotherAcct][1] = str(master[acct][1])
						newStrFirstAcct = format(master[acct][0],
master[acct][1], master[acct][2])
						masterAccts[acct] = newStr
						newStr = format(master[anotherAcct][0],
master[anotherAcct][1], master[anotherAcct][2])
						masterAccts[anotherAcct] = newStr
		return masterAccts

	elif (transCopy[0] == '04'):	#create
		temp = 0
		acctNum = int(trans[1])
```

```
                newStr = format(trans[1], trans[3], trans[4])
                for acct in range(master):
                        if (accct[0] != acctNum):
                                first = int(master[acct][0])
                                if (acct + 1 <= range(master)):
                                        second = int(master[acct + 1][0])
                                else:
                                        second = 'None'
                                if (accNum < first):
                                        masterAccts.insert(acct - 1, newStr)
                                elif (accNum > first and accNum < second):
                                        masterAccts.insert(acct, newStr)
                                elif (accNum > first and second == 'None'):
                                        masterAccts.insert(acct, newStr)
                        else:
                                throwError()
                return masterAccts

        elif (transCopy[0] == '05'):  #delete
                acctNum = int(trans[1])
                transAcctName = trans[4]
                for acct in range(master):
                        if (acctNum == acct[0]):
                                acctBalance = acct[1]
                                if (acctBalance == 0):
                                        acctName = acct[2]
                                        if (transAcctName == acctName):
                                                masterAccts.remove(masterAccts[acct])
                                        else:
                                                throwError()
                                else:
                                        throwError()
                return masterAccts

        elif (transCopy[0] == '00'):
                return masterAccts

def format(num, balance, name):
        string = "111111_1_NNNNNNNNNNNNNN"
        return string

def writeNewMasterAccounts(list):
        f = open('./masteraccounts.txt','w')
        for i in list:
                f.write(i + "\n")
        f.close()
        return 0

def writeNewValidAccounts(list):
        f = open('./validaccounts.txt','w')
        for i in list:
                #wrong doesnt write it correctly
                f.write(i + "\n")
        f.close()
        return 0

def throwError():
        sys.exit('Fatal Error')

def main_program():
        #open master accounts
```

```
        masteraccts = []
        f = open('./masteraccounts.txt')
        masteraccts = f.readlines()
        for x in range(len(masteraccts)):
                masteraccts[x] = masteraccts[x].strip()
        f.close()
        #open merged transaction file
        mergedtrans= []
        f = open('./mergedtransactions.txt')
        mergedtrans = f.readlines()
        for x in range(len(mergedtrans)):
                mergedtrans[x] = mergedtrans[x].strip()
        f.close()
        #for all transactions update the master accounts file
        for i in mergedtrans:
                masteraccts = transaction(masteraccts,i)
        #writes output files
        writeNewValidAccounts(masteraccts)
        writeNewMasterAccounts(masteraccts)
        return 0

main_program()
```