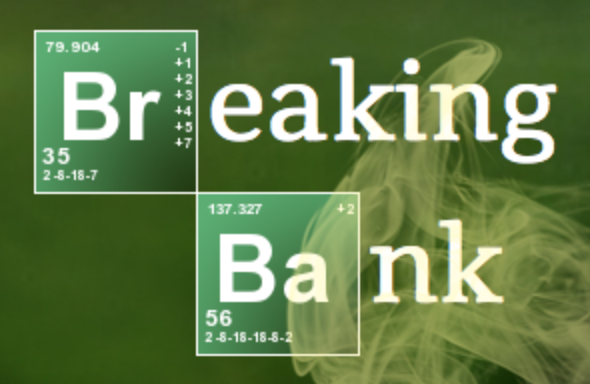
****

**CISC 327 Assignment 3**

**Scott Wallace 10051890**

**Brad Guner 10059112**

**Part 1**

This is the linux shell script we ran to do all our testing for assignment 3.

--------------------------------------------------------------------------------------------

#!/bin/bash

cd inputs

for i in \*.txt

do

echo "running $i"

FILE="$i"

cat $FILE | while read line

do

echo $line

done | python ../breakingbank.py > ../outputs/$i.log

cp ./tempsummfile.txt ../outputs/$i.sum

rm ./tempsummfile.txt

done

for i in \*.txt

do

echo "checking outputs for $i"

diff ../outputs/$i.sum ../expected/$i.sum

diff ../outputs/$i.log ../expected/$i.log

done

--------------------------------------------------------------------------------------------

**Part 2 – Failure Log**

This is a log of all our reported failures during testing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Name** | **What it was testing** | **How it’s input was wrong** | **Error in the code** | **How program was fixed** |
| test\_create\_  bad\_accnum | creating an account with a bad account name | It won’t accept any type of account number as valid | In the agent class, the create function had some errors in the if statements for validating the account number input | rewrote the if statements inside the function |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Part 3**

This is our source code for our final front end program that passes all our test. It is in python

breakingbank.py

--------------------------------------------------------------------------------------------

"""

CISC 327

Breaking Bank

Assignment #2

Scott Wallace 10051890

Brad Guner 10059112

"""

import datetime

import time

import os.path

############################################ RETAIL #################################################

class retail(object):

def \_\_init\_\_(self, type,dailylimit):

self.type = type

self.dailylimit = dailylimit

def withdraw(self):

accNumInput = True

while (accNumInput):

accNum = int(raw\_input('Account Number: '))

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Withdrawal Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 100000):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

elif (self.dailylimit + amount > 100000):

print "This amount exceeds your daily limit."

else:

self.dailylimit += amount

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '02\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('02', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def deposit(self):

accNumInput = True

while (accNumInput):

accNum = raw\_input('Account Number: ')

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Deposit Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 100000):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

else:

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '01\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('01', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def transfer(self):

accNumInput = True

accNumInput2 = True

while(accNumInput):

accNumTo = raw\_input('To Account Number: ')

print str(accNumTo) + "\n"

#CHECK to SEE IF FIRST ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumTo)):

while (accNumInput2):

accNumFrom = raw\_input('From Account Number: ')

print str(accNumFrom) + "\n"

#CHECK TO SEE IF SECOND ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumFrom)):

accNumInput = False

accNumInput2 = False

amt = True

while (amt):

amount = int(input('Transfer Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 100000):

print "Please enter a valid transfer amount."

elif (amount < 0):

print "Please enter a valid transfer amount."

else:

amt = False

#create string for write file

accNumTo = str(accNumTo)

accNumFrom = str(accNumFrom)

amount = str(amount)

#transactionInfo = '03\_' + accNumTo + '\_' + accNumFrom + '\_' + amount

transactionInfo = formatFileLine('01', accNumTo, accNumFrom, amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

return transactionInfo

#METHOD WHICH RUNS ANY TRANSACTIONS FOR A RETAIL DAY

#WILL WRITE ANY TRANSACTIONS TO FILE

#LOGOUT IS ACCEPTED AT THIS STAGE

def runRetailDay(self):

running = True

"""

#CREATES TRANSACTION SUMMARY FILE

ts = time.time()

st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

save\_path = './TransactionSummaryFiles/'

file = 'Transaction\_Summary\_File\_\_' + st + '.txt'

filename = file.replace(":", "\_")

completeName = os.path.join(save\_path, filename)

f = open(completeName,'w')

"""

file1 = 'tempsummfile.txt'

f = open(file1,'w')

while (running):

#STARTS ACCEPTING RETAIL TRANSACTIONS

transaction = raw\_input('Perform a transaction: ')

transaction.lower()

print str(transaction) + "\n"

#TESTS INPUT FOR WHICH TRANSACTION TYPE TO PERFORM

if (transaction == "withdraw"):

newTrans = self.withdraw()

f.write(newTrans + '\n')

elif (transaction == "deposit"):

newTrans = self.deposit()

f.write(newTrans + '\n')

elif (transaction == "transfer"):

newTrans = self.transfer()

f.write(newTrans + '\n')

elif (transaction == "logout"):

f.close()

running = False

else:

print "Please enter a valid transaction type."

return False

###########################################################################################################

############################################ AGENT #################################################

class agent(object):

def \_\_init\_\_(self, type):

self.type = type

def withdraw(self):

accNumInput = True

while (accNumInput):

accNum = raw\_input('Account Number: ')

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Withdrawal Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 999999):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

else:

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '02\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('02', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def deposit(self):

accNumInput = True

while (accNumInput):

accNum = raw\_input('Account Number: ')

print str(accNum) + "\n"

#CHECK TO SEE IF VALID ACCOUNT NUMBER

if (acctNumExist(accNum)): #if account num is valid

amt = True

accNumInput = False

while (amt):

amount = int(input('Deposit Amount (Cents): '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 999999):

print "Please enter a valid amount."

elif (amount < 0):

print "Please enter a valid amount."

else:

amt = False

#CREATE STRING TO WRITE TO FILE

accNum = str(accNum)

amount = str(amount)

#transactionInfo = '01\_' + accNum + '\_' + amount #NEEDS PROPER FORMATTING STILL

transactionInfo = formatFileLine('01', accNum, 'BBBBBB', amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

return transactionInfo

def transfer(self):

accNumInput = True

accNumInput2 = True

while(accNumInput):

accNumTo = raw\_input('To Account Number: ')

print str(accNumTo) + "\n"

#CHECK to SEE IF FIRST ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumTo)):

while (accNumInput2):

accNumFrom = raw\_input('From Account Number: ')

print str(accNumFrom) + "\n"

#CHECK TO SEE IF SECOND ACCOUNT NUMBER IS VALID

if (acctNumExist(accNumFrom)):

accNumInput = False

accNumInput2 = False

amt = True

while (amt):

amount = int(raw\_input('Transfer Amount (Cents) : '))

print str(amount) + "\n"

#amount = amount\*100

if (amount > 999999):

print "Please enter a valid transfer amount."

elif (amount < 0):

print "Please enter a valid transfer amount."

else:

amt = False

#create string for write file

accNumTo = str(accNumTo)

accNumFrom = str(accNumFrom)

amount = str(amount)

#transactionInfo = '03\_' + accNumTo + '\_' + accNumFrom + '\_' + amount

transactionInfo = formatFileLine('03', accNumTo, accNumFrom, amount, 'NNNNNNNNNNNNNNN')

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

return transactionInfo

def create(self):

accNumInput = True

accNameInput = True

while (accNumInput):

accNum = int(input('Enter your desired account number: '))

print str(accNum) + "\n"

if ((len(str(accNum))) <= 6):

if (accNum <= 999999):

if (not acctNumExist(accNum)):

accNumInput = False

while (accNameInput):

accName = raw\_input('Enter your desired account name: ')

print str(accName) + "\n"

if (len(accName) > 15):

print "Please enter a valid account name."

elif (len(accName) == 0):

print "Please enter a valid account name."

else:

#create account number here

accNameInput = False

#create string for write file

accNum = str(accNum)

accName = str(accName)

#transactionInfo = '04\_' + accNum + "\_" + accName #proper formatting on end of string is needed

transactionInfo = formatFileLine('04', accNum, 'BBBBBB', 'MMMMMMMM', accName)

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

else:

print "Please enter a valid account number."

return transactionInfo

def delete(self):

accNumInput = True

accNameInput = True

while (accNumInput):

accNum = int(input('Enter the account number: '))

print str(accNum) + "\n"

#CHECK TO SEE IF INPUT ACCOUNT NUMBER EXISTS

if (acctNumExist(accNum)):

accNumInput = False

while (accNameInput):

accName = raw\_input('Enter the account name: ')

print str(accName) + "\n"

#CHECK TO SEE IF INPUT ACCOUNT NAME MATCHES ACCOUNT NUMBER

if (1 == 0): #backend thing

print "Please enter the proper account name for this account."

else:

#delete account now

accNameInput = False

#create string for write file

accNum = str(accNum)

accName = str(accName)

#transactionInfo = '05\_' + accNum + '\_' + accName #proper formatting on end of string is needed

transactionInfo = formatFileLine('05', accNum, 'BBBBBB', 'MMMMMMMM', accName)

else:

print "Please enter a valid account number."

return transactionInfo

#METHOD WHICH RUNS ANY TRANSACTIONS FOR A RETAIL DAY

#WILL WRITE ANY TRANSACTIONS TO FILE

#LOGOUT IS ACCEPTED AT THIS STAGE

def runAgentDay(self):

running = True

"""

#CREATES TRANSACTION SUMMARY FILE

ts = time.time()

st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')

save\_path = './TransactionSummaryFiles/'

file = 'Transaction\_Summary\_File\_\_' + st + '.txt'

filename = file.replace(":", "\_")

completeName = os.path.join(save\_path, filename)

f = open(completeName,'w')

"""

file1 = 'tempsummfile.txt'

f = open(file1,'w')

while (running):

#STARTS ACCEPTING RETAIL TRANSACTIONS

transaction = raw\_input('Perform a transaction: ')

transaction.lower()

print str(transaction) + "\n"

#TESTS INPUT FOR WHICH TRANSACTION TYPE TO PERFORM

if (transaction == "withdraw"):

newTrans = self.withdraw()

f.write(newTrans + '\n')

elif (transaction == "deposit"):

newTrans = self.deposit()

f.write(newTrans + '\n')

elif (transaction == "transfer"):

newTrans = self.transfer()

f.write(newTrans + '\n')

elif (transaction == "create"):

newTrans = self.create()

f.write(newTrans + '\n')

elif (transaction == "delete"):

newTrans = self.delete()

f.write(newTrans + '\n')

elif (transaction == "logout"):

f.close()

running = False

else:

print "Please enter a valid transaction type."

return False

###########################################################################################################

def formatFileLine(transCode, firstAcctNum, secondAcctNum, acctAmt, acctName):

transCode = str(transCode)

firstAcctNum = str(firstAcctNum)

secondAcctNum = str(secondAcctNum)

acctAmt = str(acctAmt)

acctName = str(acctName)

#transaction code

if (len(transCode) == 2):

fileLine = transCode + "\_" #line: CC\_

#first account number

if (len(firstAcctNum) == 6):

firstAcctNum += "\_"

fileLine += firstAcctNum #line: CC\_AAAAAA\_

elif (len(firstAcctNum) < 6 and len(firstAcctNum) > 0): #pads 0 to beginning of account numbers

acctLength = len(firstAcctNum)

diff = 6 - acctLength

for i in range(diff):

firstAcctNum = "0" + firstAcctNum

firstAcctNum += "\_"

fileLine += firstAcctNum

#second account number

if (len(secondAcctNum) == 6):

secondAcctNum += "\_"

fileLine += secondAcctNum #line: CC\_AAAAAA\_BBBBBB\_

elif (len(secondAcctNum) < 6 and len(secondAcctNum) > 0):

acctLength = len(secondAcctNum)

diff = 6 - acctLength

for i in range(diff):

secondAcctNum = "0" + secondAcctNum

secondAcctNum += "\_"

fileLine += secondAcctNum

#transaction amount

if (len(acctAmt) == 8):

acctAmt += "\_"

fileLine += acctAmt #line: CC\_AAAAAA\_BBBBBB\_MMMMMMMM\_

elif (len(acctAmt) < 8 and len(acctAmt) > 0):

amtLength = len(acctAmt)

diff = 8 - amtLength

for i in range(diff):

acctAmt = "0" + acctAmt

acctAmt += "\_"

fileLine += acctAmt

#account name

if (len(acctName) == 15):

fileLine += acctName #line: CC\_AAAAAA\_BBBBBB\_MMMMMMMM\_NNNNNNNNNNNNNNN

elif (len(acctName) > 15):

newAcctName = ""

while (len(newAcctName) < 15):

for char in acctName:

newAcctName += char

fileLine += newAcctName

else:

nameLength = len(acctName)

diff = 15 - nameLength

for i in range(diff):

acctName = "0" + acctName

acctName += "\_"

fileLine += acctName

return fileLine

def readAcctFile():

list = []

#f = open('./validaccts.txt')

f = open('../validaccts.txt') #this one is for a3 testing

list = f.readlines()

for x in range(len(list)):

list[x] = list[x].strip()

list[x] = int(list[x])

f.close()

return list

def acctNumExist(num):

num = int(num)

for x in accounts:

if (x == num):

return True

return False

def openBankingSystem():

loggedIn = True

while (loggedIn):

#GETS LOGIN TO START, STAGE 0

firstInput = raw\_input('Type "login" to login: ')

firstInput.lower()

firstInput = str(firstInput)

print firstInput + "\n"

if (firstInput == "login"):

pickDay = True

while (pickDay):

#ACCEPTS INPUT FOR AGENT OR RETAIL, STAGE 1

dayType = raw\_input('agent or retail: ')

dayType.lower()

dayType = str(dayType)

print dayType + "\n"

if (dayType == "retail"):

pickDay = False

retailDay = retail(dayType,0)

loggedIn = retailDay.runRetailDay()

elif (dayType == "agent"):

pickDay = False

agentDay = agent(dayType)

loggedIn = agentDay.runAgentDay()

else:

print "Please enter a valid input.\n"

elif (firstInput == "stop"):

#Entering stop will kill the program

return 0

else:

print "Please enter a valid input.\n"

#STARTS OVER AGAIN AFTER LOGOUT AT STAGE 0

return openBankingSystem()

###### MAIN PROGRAM ######

accounts = readAcctFile()

openBankingSystem()

--------------------------------------------------------------------------------------------