Design Document
for the
Back Office of Simple Interactive Banking System
(SimBank)

Submitted By:

Tech Testers

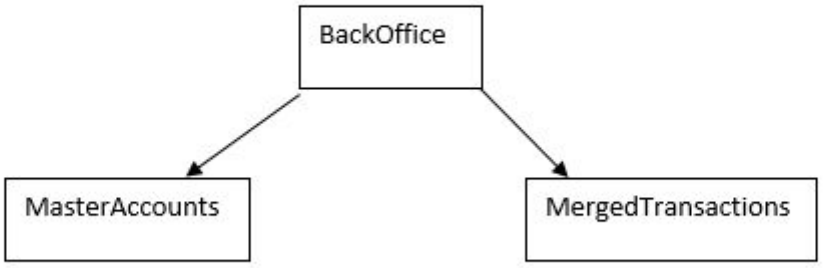
Zainab Bello - 10147946

Wanyu Zhang - 10141510

Jia Yue Sun (Selena) - 10152968

1. OVERVIEW OF THE CLASS HIERARCHY

Diagram showing the class/object hierarchy of Tech Tester's design of the Back Office of SimBank



The BackOffice class is the main class for the Back Office program. The MasterAccounts and MergedTransactions classes both extend it in order to gain access to its attributes.

2. DESCRIPTION OF CLASSES AND METHODS

Class BackOffice

---

public abstract class **BackOffice**

This is the "main" class for the Back Office program. It is an abstract class that takes in the master accounts file and the merged transaction summary file and stores the contents into an arrayList of string arrays. It then creates an instance of the MergedTransactions class in order to update the master accounts file.

At the end of the program, a new master accounts file and a valid accounts file is created.

The program is intended to run by supplying the input files as parameter to this class.

Constructor Summary

Constructor and Description
BackOffice() **Description:** Creates an instance of the BackOffice class with no parameters

Method Detail

Method and Description
protected void addMaster(String[] trans) **Description:** Adds a new line to the arrayList containing the master accounts file **Parameters:** trans - a newline to be added to the arrayList
public ArrayList<String[]> getMaster() **Description:** Returns the arrayList containing the master accounts file **Return:** The arrayList containing the master accounts file
public static Boolean checkAccountNo(String inputNo) **Description:** Checks the format of the account number **Parameters:** inputNo - account number whose format is to be checked **Return:** Returns true if the account number format is valid, false otherwise.
public static Boolean checkBalance(String amount) **Description:** Checks if the balance is in the valid format **Parameters:** amount - balance to be checked **Return:** Returns true if amount is in the valid format, false otherwise
public static Boolean checkAccountName(String userName) **Description:** checks if the account name format is valid **Parameters:** userName - the account name of the User

Return: Returns true if account name is in the valid format, false otherwise
private static void checkMasterFormat() **Description:** Check if the master accounts file is in the valid format
private static void checkMergedFormat() Check if the merged transaction summary file is in the valid format
public static void readFile(File filename, ArrayList<String[]> List) **Description:** Reads a file and stores the data into an arrayList **Parameters:** filename: the name of the file list: the arrayList in which the data from the file is stored
public static void main(String[] args) **Description:** Reads in the master accounts file and the merged transaction summary File into the program **Parameters:** args - the master accounts file and the merged transaction summary file

Class MergedTransactions

---

class **MergedTransactions**

extends **BackOffice**

This class obtains the contents of the arrayList containing the merged transactions file and updates the arrayList of master accounts with it. If there's an invalid value in any of the fields, it stops the program immediately and logs a fatal error onto the terminal.

Method Detail

Method and Description
public void createMaster(String[] line) **Description:** Adds a newly create account to the arrayList with the contents of the master accounts file

Parameters:
line - the transaction summary line to be added to the master accounts file

Public int getIndex(String acctNum)
Description:
Gets the index of the specified account number from the arrayList containing the master accounts file

Parameters:
acctNum - the account number being searched for

Return:
Returns the index of the specified account number if it exists, if not exit the program.

public void deleteMaster(String[] line)
Description:
Checks if the account balance is 0 and if the account name given is associate with the account number before deleting the account number from the master accounts arrayList

Parameters:
line - the transaction summary line stating the account number to be deleted

public int withdrawMaster(String[] line)
Description:
Check if the account balance is positive after withdrawing a certain amount. If it is, it updates the master accounts arrayList

Parameters:
line - the transaction summary line stating the account number a certain amount is to be withdrawn from

Return:
Returns 0 if the withdraw transaction is successful, otherwise returns -1

public void depositMaster(String[] line)
Description:
Adds the deposit transaction summary line to the arrayList containing the master accounts file

Parameters:
line - the transaction summary line stating the account number a certain amount is to be deposited to

public void transferMaster(String[] line) **Description:** Checks that there are enough money in the account the user is transferring from before updating the master accounts arrayList **Parameters:** line - the transaction summary line stating the account numbers a certain amount should be withdrawn from and deposited to.
public void mergeStart() **Description:** Indicates the start of the MergedTransactions class. Depending on what transaction should be processed, it filters each line of the merged transaction summary file arrayList to the appropriate function

Class MasterAccounts

---

class **MasterAccounts**

extends **BackOffice**

This class deals with the creation of a new master accounts file and a valid accounts file. It takes the contents of the arrayList containing the master accounts file, extracts the account numbers from it and writes the account numbers into a new valid accounts file. It also writes the contents of the master accounts arrayList into a new file.

Method Detail

Method and Description
private void writeNewMaster(ArrayList<String[]> masterList) **Description:** Writes a new Master Accounts file **Parameters:** masterList - the arrayList containing the master Accounts file
private void writeValidAcct(ArrayList<String> validList) **Description:** Writes a new valid account file **Parameters:**

validList - the arrayList that contains the valid Accounts

Private ArrayList<String> extractValidAccounts(ArrayList<String[]> masterAcct)
Description:
Extracts the new valid accounts list from the list containing the master accounts
Parameters:
masterAcct - the arrayList containing the master accounts
Return:
the arrayList containing valid accounts

Public int compareTo (Object obj)
Description:
This method overrides the java compareTo method

Parameters: obj - the object to be compared

public void masterStart()
Description:
Indicates the start of the MasterAccounts class