

Back-Office Design document

Team: Zero Defects



|  |  |
| --- | --- |
| Name | Student Number |
| Muhammad Usman Majeed | 10086980 |
| Jessica Nahulan | 10029341 |
| Johan Cornelissen | 10098176 |

Contents

[QBasic Application Usage: 3](#_Toc496037025)

[Architecture Description: 3](#_Toc496037026)

[Solution Structure: 4](#_Toc496037027)

[Classes: 4](#_Toc496037028)

[Methods: 5](#_Toc496037029)

# QBasic Back-Office Application Usage:

The following provides a brief overview of how to use the QBasic Back-Office python application.

The Back-Office takes four arguments: the previous day Master Accounts File, the Merged Transaction Summary File, and file to use for the new Master Accounts File created, and the file to use for the new Valid Accounts File created.

The previous day Master Accounts File is created by the last run of the back-office application. The Merged Transaction Summary File will be provided by an intermediate script that concatenates the Front End’s Transaction Summary Files. Lastly the new Master Accounts File created by the Back-Office application will be used by the next day’s Back-Office, and the new Valid Accounts File will be used by the next day’s Front-End sessions.

**How to run the Back-Office application:**

./BackOffice.py masteraccount.txt transactionsummary.txt newmasteraccount.txt newvalidaccount.txt

# Architecture Description:

The Qbasic back end follows an object-oriented approach made up of 5 classes. These 5 classes consist of BackOffice, TxnProcess, Utility, ErrorHandler and an Account object class. The BackOffice class is responsible for interacting with the input and output files to the system (master/valid accounts files, and merged transaction summary file). The BackOffice class reads in the contents of the master accounts file, and the merged transaction summary file and calls the appropriate methods from TxnProcess based on the transaction codes read through the transaction summary file. Account objects are created in a dictionary using the account number as the dictionary key, allowing for easy Account object lookup. The Account object is a basic data structure that keeps track of the account balance, account name, and account number. Making use of Account objects stored in a dictionary allows for simplicity when updating an account’s balance (using mutator methods) or deleting/creating an account all together. The TxnProcess class implements the logic required for each transaction code to ensure command constraints are met and the proper Account objects are modified. For example, the TxnProcess method txn\_dep ensures constraints are met for the “to” account and value being deposited before updating the balance for the desired Account object. Logic that is required in multiple places, like filling the account object with the master account file, is done through methods in the Utility class. The Utility class members are used by multiple TxnProcess methods and ensures the architecture is modular. The ErrorHandler class is used throughout all the classes to allow for clean and concise error reporting. A simple ErrorHandler method can be invoked using a unique error code to allow for a more detailed error message to be displayed to user. A class relationship diagram is provided below in the “Solution Structure” section to illustrate how the classes interact with one another.

# Solution Structure:

## Classes:

Log Output

BackOffice

MasterAccounts.txt

NewMasterAccounts.txt

TransactionSummary.txt

ValidAccountsFile.txt

Account

\*Multiple Account Obj’s exist

Utility

TxnProcess

## 

ErrorHandler

## Methods:

|  |  |  |
| --- | --- | --- |
| Class | Function | Intentions |
| BackOffice | Main | Reads the input and output file names and then calls appropriate TxnProcess functions based on the transaction summary file given. |
| TxnProcess | txn\_new | Function to process a “NEW” (createacc) transaction code.  Creates a new Account object in accounts\_dic. |
| txn\_del | Function to process a “DEL” (deleteacct) transaction code.  Deletes an Account object from accounts\_dic. |
| txn\_dep | Function to process a “DEP” (deposit) transaction code.  Adds deposit amount to account balance in desired Account object found using account number key in accounts\_dic. |
| txn\_wdr | Function to process a “WDR” (withdraw) transaction code.  Subtracts withdraw amount from account balance in desired Account object found using account number key in accounts\_dic. |
| txn\_xfr | Function to process a “XFR” (transfer) transaction code.  Transfer money from one Account object to another Account object, accessing each object using account number key in accounts\_dic. |
| txn\_eos | Function to process the “EOS” line from transaction file to ensure transaction summary file validity. |
| Account | \_\_init\_\_ | Initializes the Account object attributes. |
| get\_account\_num | Accessor to return an Account's account number. |
| get\_account\_name | Accessor to return an Account's account name. |
| get\_account\_balance | Accessor to return an Account's account balance. |
| set\_account\_balance | Mutator function to set a new account balance for an Account object. |
| Utility | process\_master\_account | Function to create a dictionary of Account objects from the master accounts file.  This allows for a cached version of the master accounts file for any future transactions. |
| create\_master\_valid\_account\_files | Function to write all cached dictionary items (account objects) to master account file and valid account file. |
| ErrorHandler | process\_error | Function to send error string to stdout. |
| \_\_init\_\_ | Initialize error code to error string mapping. |