

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 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 4 classes and an account object class. These 5 classes consist of BackOffice, TxnProcess, Utility, ErrorHandler and account object. The BackOffice class is responsible for collecting the information regarding the input and out files name and location, then BackOffice class call certain methods from TxnProcess based on the transaction codes read from the transaction summary file. The TxnProcess calls have the implement description for each transaction code, TxnProcess method configures the transaction code commands constraint and behavior. Logic that is required in multiple places, like filling the account object with 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 ow the classes interact with one another. Lastly Account object is just a data structure that keeps track of the account balance, newly created or deleted accounts based on the master account file.

# Solution Structure:

## Classes:

Log Output

BackOffice

MasterAccounts.txt

NewMasterAccounts.txt

TransactionSummary.txt

ValidAccountsFile.txt

Account Object(s)

TxnProcess

Utility

## 

ErrorHandler

## Methods:

|  |  |  |
| --- | --- | --- |
| Class | Function | Intentions |
| BackOffice | Main | Reads the input and out file names and then call correct TxnProcess functions based on the transaction summary file given. |
| TxnProcess | txn\_new | Function to process a “NEW” transaction code.  Create a new account is account object data structure |
| txn\_del | Function to process a “DEL” transaction code.  Delete an account is account object data structure |
| txn\_dep | Function to process a “DEP” transaction code.  Deposit money to an account is account object data structure (increment balance) |
| txn\_wdr | Function to process a “WDR” transaction code.  Withdraw money from an account is account object data structure (decrement balance) |
| txn\_xfr | Function to process a “XFR” transaction code.  Transfer money from one account to another account is account object data structure |
| txn\_eos | Function to process the “EOS” line from transaction file to ensure 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. |