|  |  |  |  |
| --- | --- | --- | --- |
| 1 | To test the print buttons. There are three print buttons: No Order, Last Name, and Date Open. No Order will simply print the accounts and relevant information in no specific order, Last Name and Date Open will print statements for each account sorted by account holder’s last name and date the account was created respectively. Possible error states include:   * Empty database | Empty Database->No Order  Empty Database->Last Name  Empty Database->Date Open  Account Database from importing database.txt->No Order  Account Database from importing database.txt->Last Name  Account Database from importing database.txt->Date Open | Database is empty.  Database is empty.  Database is empty.  List of Accounts including account types, names, dates opened, and balance in no order  Account statements for each account currently in the database ordered by Last Name lexicographically  Account statements for each account currently in the database ordered by Date Opened from oldest to youngest |
| 2 | To Test the Deposit and Withdrawal buttons. The deposit button will deposit a specified amount into a specified account via first name, last name, and account type. The withdrawal will use the same parameters except it will withdrawal the amount from the account. Possible error states include:   * Account does not exist * Holder name matches but type does not * Non-numerical amount specified * Negative amount specified * For withdrawal, withdrawing more than current balance | Using Database imported from database.txt:  (type,firstname,lastname,amount)  (Checking,“a”,”b”,500)->deposit  (Checking,“Tom”,”Moore”,500)->deposit  (Savings,”Tom”,”Moore”,”a”)->deposit  (Savings,”Tom”,”Moore”,-500)->deposit  (Savings,”Tom”,”Moore”,500)->deposit  (checking,“a”,”b”,500)->withdrawal  (Checking,“Tom”,”Moore”,500) ->withdrawal  (Savings,”Tom”,”Moore”,”a”) ->withdrawal  (Savings,”Tom”,”Moore”,-500) ->withdrawal  (Savings,”Tom”,”Moore”,1001.41) ->withdrawal  (Savings,”Tom”,”Moore”,500) ->withdrawal | Account does not exist.  Account does not exist.  Please enter a valid amount  Please enter a valid amount  500.00 deposited to account.  Account does not exist.  Account does not exist.  Please enter a valid amount  Please enter a valid amount  Insufficient funds.  500.00 withdrawn from account. |
| 3 | File IO:  Test importing a text file database.txt by manually entering path, and by using the file explorer to get a path. Print the accounts to verify the data is imported  To test export, we will again try manual and file explorer generated paths, then export after import. We will also check modifying the in-memory account database and writing back. Error states include:   * Invalid path * Null or blank path | (“database.txt”)->import  (“database.txt”)->export  (“database.txt”)->import    (“database.txt”)->import  (“[full path]/database.txt”)->import  (“[full path]/database2.txt”)->export  (“”) -> import  (null)->import  (“”) -> import  (null)->import  ([invalid path])->import  ([invalid path])->export | Imported File (verified with print)  Exported File (manually verified with text editor)  Imported File (verified with print it is identical to the first import hence no data loss)  Imported File (no duplicate accounts added)  Imported File (using file explorer to generate the path, verified by print)  File not found, creating new file.  Exported File (creates new file, verified with text editor, using file explorer to generate path)  File not found, check the file name  File not found, check the file name (null via cancel in file explorer)  No file name, please fill in the adjacent textbox.  File not found, check the file name  File not found, creating new file.  Cannot export, the filename or file path is probably invalid. |
| 4 | Open and Close:  To test we will enter correct data and attempt to add accounts of various types. We will also try to close these accounts either after opening or at a later time. We use print to verify the open and close operations. We will also test error states, including:   * Opening duplicate accounts * Closing closed and invalid accounts * Using invalid data like negative balances * Using invalid dates * Using alphabetical characters when numerical ones are expected * Account data mismatch when trying to close * Any other invalid fields | [imported database.txt]  (“Jane”, “Doe”, 500, checking, 2/29/2008)->open  (“Jane”, “Doe”, 500, checking, 2/29/2008)->close  (“Jane”, “Doe”, 500, checking, 2/29/2008)->close  (“Jane”, “Doe”, 500, checking, 2/29/2008)->open  (“Jane”, “Doe”, 500, moneymarket, 2/29/2008)->open  (“Jane”, “Doe”, 500, savings, 11/1/2019, loyalty)->close  (“Jane”, “Doe”, 500, savings, 11/1/2019, loyalty)->open  Changing any parameter (amount, first, last, date)->close  (“first”, “last”, -90, savings, 11/11/2011)->open  (“first”, “last”, 100, savings, 13/11/2011)->open  (“first”, “last”, 100, savings, 11/11/abc)->open  (“first”, “last”, t, savings, 11/11/2011)->open | [successful operations verified with print]  Account already added, nothing to do.  Account successfully deleted.  Account not found.  Successfully added the checking account.  Successfully added the money market account.  Account successfully deleted.  Successfully added the savings account.  Account not found.  The balance must be non negative.  Please ensure the date is correct.  Please ensure the date is correct.  Please enter a nonnegative numeric amount. |

Test Case # Purpose/error states Input->action Expected Output