Restaurant Management System

Microsoft | [Company address]

technical support manual

Dhiraj Ranjit

Muath Aldhubayb

Chris Tiller

2014 Edition

Table of Contents

[Class Definitions 2](#_Toc400051163)

Class Definitions

[DataColumn 6](#_Toc402260934)

[DataRow 7](#_Toc402260935)

[Recordset 11](#_Toc402260936)

[Inventory 19](#_Toc402260937)

DataColumn

Remarks

Constructors 4

DataRow() 4

Variables 4

Methods 5

## Constructors

### DataColumn()

Sets the initial values for the DataColumn object.

|  |  |
| --- | --- |
| Variable Name | Default Value |
| m\_next | NULL |
| m\_prev | NULL |
| m\_name | “” |
| m\_firstRow | NULL |
| m\_currentRow | NULL |
| m\_lastRow | NULL |
| m\_rowLength | 0 |

## Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Access Modifier | Description |
| m\_next | DataColumn\* | Private | Pointer to the DataColumn’s object next DataColumn object |
| m\_prev | DataColumn\* | Private | Pointer to the DataColumn’s object previous DataColumn object |
| m\_name | std::string | Private | String that contains the DataColumn’s name |
| m\_firstRow | DataRow\* | Private | Pointer to the first DataRow in the object |
| m\_currentRow | DataRow\* | Private | Pointer to the current DataRow in the object |
| m\_lastRow | DataRow\* | Private | Pointer to the last DataRow in the object |
| m\_rowLength | Integer | Private | Length of the longest entry in the DataColumn |

## Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access Modifier | Description |
| [getNext](#_getNext()_1) | DataColumn\* | Public const | Gets the next DataColumn object |
| [getPrev](#_getPrev()_1) | DataColumn\* | Public const | Gets the previous DataColumn object |
| [getColumnName](#_getColumnName()) | std::string | Public const | Gets the name of the DataColumn object |
| [setNext](#_setNext()_1) | Void | Public | Sets the DataColumn object’s next variable |
| [setPrev](#_setPrev()_1) | Void | Public | Sets the DataColumn object’s previous variable |
| [setColumnName](#_setColumnName()) | Void | Public | Sets the DataColumn object’s name |
| operator=(string) | Void | Public | Sets the current row text to the value given |
| operator=(int) | Void | Public | Sets the current row test to the value given |
| operator=(double) | Void | Public | Sets the current row test to the value given |
| operator=(DataColumn&) | DataColumn& | Public | Sets the current row test to the value given |
| operator==(string) | Bool | Public | Determines if the current row text is equal to the argument given |
| operator!=(string) | Bool | Public | Determines if the current row test is not equal to the argument given |
| operator<< | ostream& | Public | Prints out the current row text in the DataColumn |
| addRow | Void | Public | Appends a DataRow to the end of the list |
| removeRow | Void | Public | Removes the current DataRow from the list |
| removeAll | Void | Public | Removes all DataRows from the list |
| moveFirst | Void | Public | Moves the current row pointer to the first row |
| moveNext | Void | Public | Sets the current row pointer to the next current row |
| movePrev | Void | Public | Sets the current row to the previous row |
| moveLast | Void | Public | Sets the current row to the last row |
| setColWidth | Void | Public | Sets the |
| getColWidth | Int | Public |  |
| recalculateRowLength | Void | Public |  |
| toInt | Int | Public |  |
| toDouble | Double | Public |  |
| getRowText | String | Public |  |
|  |  |  |  |

### getNext()

Prototype: DataColumn\* getNext() const

Description: Gets the current DataColumn object’s next DataColumn to link in a linked list.

### getPrev()

Prototype: DataColumn\* getPrev() const

Description: Gets the current DataColumn object’s previous DataColumn to link in a linked list.

### getColumnName()

Prototype: std::string getColumnName()

Description: Gets the current DataColumn object’s text field.

### setNext()

Prototype: void setNext(DataColumn \*)

Parameters: DataColumn \* - Pointer to the DataColumn you want to set as the next DataColumn to the current DataColumn

Description: Sets the current DataColumn object’s next DataColumn to link in a linked list.

### setPrev()

Prototype: void setPrev(DataColumn \*)

Parameters: DataColumn \* - Pointer to the DataColumn you want to set as the previous DataColumn to the current DataColumn

Description: Sets the current DataColumn object’s previous DataColumn to link in a linked list.

### setColumnName()

Prototype: void setColumnName(std::string)

Description: Sets the current DataColumn object’s next DataColumn to link in a linked list.

DataRow

Remarks

[Constructors 8](#_Toc402260233)

[DataRow() 8](#_Toc402260234)

[Variables 8](#_Toc402260235)

[Methods 8](#_Toc402260236)

## Constructors

### DataRow()

Sets the initial values for the DataRow object.

|  |  |
| --- | --- |
| Variable Name | Default Value |
| m\_next | NULL |
| m\_prev | NULL |
| m\_text | “” |

## Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Access Modifier | Description |
| m\_next | DataRow\* | Private | Pointer to the DataRow’s object next DataRow object |
| m\_prev | DataRow\* | Private | Pointer to the DataRow’s object previous DataRow object |
| m\_text | std::string | Private | String that contains the DataRow’s text |

## Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access Modifier | Description |
| [getNext](#_getNext()) | DataRow\* | Public const | Gets the next DataRow object |
| [getPrev](#_getPrev()) | DataRow\* | Public const | Gets the previous DataRow object |
| [getText](#_getText()) | std::string | Public const | Gets the text of the current DataRow object |
| [setNext](#_setNext()) | Void | Public | Sets the DataRow object’s next variable |
| [setPrev](#_setPrev()) | Void | Public | Sets the DataRow object’s previous variable |
| [setText](#_setText()) | Void | Public | Sets the DataRow object’s text |

### getNext()

Prototype: DataRow\* getNext() const

Description: Gets the current DataRow object’s next DataRow to link in a linked list.

### getPrev()

Prototype: DataRow\* getPrev() const

Description: Gets the current DataRow object’s previous DataRow to link in a linked list.

### getText()

Prototype: std::string getText()

Description: Gets the current DataRow object’s text field.

### setNext()

Prototype: void setNext(DataRow\*)

Parameters: DataRow\* - Pointer to the DataRow you want to set as the next DataRow to the current DataRow

Description: Sets the current DataRow object’s next DataRow to link in a linked list.

### setPrev()

Prototype: void setPrev(DataRow\*)

Parameters: DataRow\* - Pointer to the DataRow you want to set as the previous DataRow to the current DataRow

Description: Sets the current DataRow object’s previous DataRow to link in a linked list.

### setText()

Prototype: void setText(std::string)

Description: Sets the current DataRow object’s next DataRow to link in a linked list.

Recordset

Remarks

Recordset class requires the DataColumn and DataRow classes.

[Constructors 12](#_Toc402259495)

[Recordset() 12](#_Toc402259496)

[Variables 12](#_Toc402259497)

[Methods 13](#_Toc402259498)

## Constructors

### Recordset()

Sets the initial values of the variables in the Recordset Object.

|  |  |
| --- | --- |
| Variable Name | Default Value |
| m\_firstColumn | NULL |
| m\_currentColumn | NULL |
| m\_lastColumn | NULL |
| m\_numRows | 0 |
| m\_numColumns | 0 |
| m\_currentRow | 0 |

## Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Access Modifier | Description |
| m\_firstColumn | DataColumn\* | Private | Pointer to the first column in the Recordset object |
| m\_currentColumn | DataColumn\* | Private | Pointer to the current column in the Recordset object |
| m\_lastColumn | DataColumn\* | Private | Pointer to the last column in the Recordset object |
| m\_numRows | Integer | Private | Holds the number of rows in the Recordset object |
| m\_numColumns | Integer | Private | Holds the number of columns in the Recordset object |
| m\_currentRow | Integer | Private | Holds the number of the current row in the Recordset object |
| m\_fileName | String | Private | Holds the name of the text file that the Recordset will read from and write to |
| m\_delimiter | String | Private | Holds the character that will delimit all the data within the Recordset |
| m\_autoIncrement | Bool | Private | Holds whether or not the Recordset will auto increment the ID field when you add a row |

## Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access Modifier | Description |
| [addField](#_addField()) | Void | Public | Adds a field to the Recordset object |
| [addRow](#_addRow()) | Void | Public | Adds a row to the Recordset object |
| [columnExists](#_columnExists()) | Bool | Public | Determines if a column exists based on the name |
| [containsRow](#_containsRow()) | Bool | Public | Determines if a row matches all the specified fields in argument |
| [filter](#_filter()) | Recordset | Public | Filters the parent Recordset based on arguments given |
| [fields](#_fields()) | DataColumn& | Public | Sets the current column to the column with the name that matches the argument given |
| [getColumns](#_getColumns()) | Int | Public const | Gets the number of columns withing the Recordset object |
| [getColumnHeaders](#_getColumnHeaders()) | vector<ColumnRowIntersection> | Public | Returns a CRI with all the column headers within the Recordset object |
| [getRow](#_getRow()) | Int | Public const | Gets the number of the current row you are on |
| [getRows](#_getRows()) | Int | Public const | Gets the total number of rows within the Recordset object |
| [isAutoIncrement](#_isAutoIncrement()) | Bool | Public | Returns whether or not the Recordset is to auto increment a new row ID value |
| [isEmpty](#_isEmpty()) | Bool | Public | Determines if the Recordset object doesn’t have any rows |
| [load](#_load()) | Void | Public | Loads the Recordset with the data found in the specified text file |
| [moveFirst](#_moveFirst()) | Void | Public | Sets the current row in the Recordset object to the first row |
| [moveNext](#_moveNext()) | Void | Public | Sets the current row to the next row in the Recordset object |
| [movePrev](#_movePrev()) | Void | Public | Set the current row to the previous row in the Recordset object |
| [moveLast](#_moveLast()) | Void | Public | Sets the current row to the last row in the Recordset object |
| [moveTo](#_moveTo()) | Void | Public | Sets the current row to the row number passed in in the Recordset object |
| [print](#_print()) | Void | Public | Prints out the contents of the recordset with the desired columns and specified row values given |
| [removeAll](#_removeAll()) | Void | Public | Removes all the rows contained within the Recordset object |
| [removeRow](#_removeRow()) | Void | Public | Removes the current row you are in within the Recordset object |
| [setAutoIncrement](#_setAutoIncrement()) | Void | Public | Sets whether or not the Recordset will auto increment the ID field within the Recordset object |
| [write](#_write()) | Void | Public | Writes the data in the Recordset to the specified text file |

### addField()

Prototype: void addField(std::string)

Parameters: std::string – the name of the DataColumn you want to give

Description: Creates a new DataColumn object, gives is the name based on the string argument, and appends it to the current DataColumn collection the Recordset already has.

### addRow()

Prototype: void addRow(bool=false)

Parameters: bool – only needs used when reading from a text file

Description: Goes through each DataColumn in the Recordset and adds a row to each and sets the current row to the newly created row at the end of the Recordset

### columnExists()

Prototype: bool columnExists(std::string)

Parameters: std::string – name of the column in question

Description: Returns whether or not a DataColumn with the name given in the argument is found within the Recordset

### containsRow()

Prototype: bool containsRow(std::vector<ColumnRowIntersection>)

Parameters: std::vector<ColumnRowIntersection> - vector that contains the structure that holds specific columns and well as specific row values to the corresponding columns

Description: Returns whether or not a row exists that contains the required information in the vector and sets the current row to that row.

### containsRow()

Prototype: bool containsRow(std::string, std::string)

Parameters: std::string – Column Name to check

Std::string – Row value to check the column on

Description: Returns whether or not a row exists that contains the required information in the arguments and sets the current row to that row.

### filter()

Prototype: Recordset filter(std::string, std::string)

Parameters: std::string – the name of the column you want to filter on

Std::string – the value you want to filter on in the column

Description: Returns a Recordset that has been filtered on the given column and value.

### fields()

Prototype: DataColumn& fields(std::string)

Parameters: std::string – the name of the Column to use

Description: Finds the DataColumn whose name matches the argument and returns a reference to that DataColumn.

### getColumns()

Prototype: int getColumns() const

Description: Returns the number of columns in the Recordset

### getColumnHeaders()

Prototype: std::vector<ColumnRowIntersection> getColumnHeaders()

Description: Returns a vector containing all the columns names within the Recordset

### getRow()

Prototype: int getRow() const

Description: Returns the number of the current row you are on in the Recordset.

### getRows()

Prototype: int getRows() const

Description: Returns the number of rows in the Recordset

### isAutoIncrement()

Prototype: bool isAutoIncrement()

Description: Returns whether or not the Recordset will auto increment the ID field upon adding a new row

### isEmpty()

Prototype: bool isEmpty()

Description: Returns whether or not the Recordset has any rows.

### load()

Prototype: int getRow() const

Description: Returns the number of the current row you are on in the Recordset.

### moveFirst()

Prototype: void moveFirst()

Description: Sets the current row to be the first row in the Recordset.

### moveNext()

Prototype: void moveNext()

Description: Sets the current row to be the next row in the Recordset.

### movePrev()

Prototype: void movePrev()

Description: Sets the current row to be the previous row in the Recordset.

### moveLast()

Prototype: void moveLast()

Description: Sets the current row to be the last row in the Recordset.

### moveTo()

Prototype: void moveTo(int)

Parameters: int – number of row to move to

Description: Sets the current row to the row number in the argument in the Recordset.

### print()

Prototype: void print(std::string)

Description: Prints out the data in the Recordset in a grid-like manner with specified columns and row values for corresponding columns.

### removeAll()

Prototype: void removeAll()

Description: Removes all the rows in the Recordset.

### removeRow()

Prototype: void removeRow()

Description: Removes the current row in the Recordset.

### setAutoIncrement()

Prototype: void setAutoIncrement()

Description: Sets the auto increment variable to allow the Recordset object to auto increment values in the ID column.

### write()

Prototype: void write()

Description: writes all the data contained in the Recordset to the specified test file.

Inventory

Remarks

Constructors 20

Inventory() 20

Variables 20

Methods 21

## Constructors

### Inventory()

Sets the initial values for the Inventory object.

|  |  |
| --- | --- |
| Variable Name | Default Value |
| in\_file | NULL |
| qtyTot | 0 |
| Per\_Price\_Total | 0 |
| priceTot | 0 |
| listCount | NULL |

## Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Access Modifier | Description |
| in\_file | Integer | Private | Hold and save the number of item from the external file |
| qtyTot | Double | Private | Holds the value of the total Quantity of each item |
| Per\_Price\_Total | Double | Private | Hold the value of the total Price of the all items |
| priceTot | Double | Private | Hold the value of the total price of each item |
| listCount | Integer | Private | Counts the number of item in external file |

## Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Return Type | Access Modifier | Description |
| [menu\_Display](#_menu_Display()) | Void | Public | User Interface for a Main Menu |
| [readFile](#_readFile()) | Integer | Public | Function to read the items from external file |
| [viewInventory](#_viewInventory()) | Void | Public | Function for view info inventory feature |
| [sortItem\_code](#_sortItem_code_()) | Void | Public | Function to sort the item in the list by item code |
| [nameSearch](#_nameSearch_()) | Void | Public | Function to search item by the name |
| [addQty](#_addQty_()) | Void | Public | Function to edit item in current inventory |
| [addNewItem](#_addNewItem_()) | Void | Public | Function for adding new item in the inventory |
| [delItem](#_delItem_()) | Void | Public | Function to delete item in current inventory |
| [printInventory](#_printInventory_()) | Void | Public | Function to print the data in class after reading external file. |
| [printListing](#_printListing_()) | Void | Public | Function to print to new external file, the inventory listing (text version) |
| [help](#_help_()) | Void | Public | Instructions for users |
| [bar](#_bar_()) | Void | Public | List bar at the top of inventory for display with description |
| [welcomeMessage](#_welcomeMessage_()) | Void | Public | Header of the program with the name of restaurant |
| [invTempl](#_invTempl_()) | Void | Public | Function for displaying the inventory |
| [update\_file](#_update_file_()) | Void | Public | Functiom to write to an external file |

### menu\_Display()

Prototype: void menu\_Display()

Description: User Interface for Inventory page. Prompt user for a query.

### readFile()

Prototype: int readFile()

Description: Read the items from the external file and returns listcount value.

### viewInventory()

Prototype: void viewInventory()

Description: View the current item in the inventory.

### sortItem\_code ()

Prototype: void sortItem\_code()

Description: Sort the item in the inventory file by item code.

### nameSearch ()

Prototype: void nameSearch()

Description: Search the item in the inventory file by the item name.

### addQty ()

Prototype: addQty()

Description: Edit the item’s item code, quantity and price. Use + for adding and – for subtracting the item quantity

### addNewItem ()

Prototype: void addNewItem()

Description: Add the new inventory item with unique item code, quantity and price

### delItem ()

Prototype: void delItem()

Description: Delete the item from the inventory file.

### printInventory ()

Prototype: void printInventory()

Description: Print the current inventory items in the screen.

### printListing ()

Prototype: void printListing()

Description: Print the current inventory item in the external file.

### help ()

Prototype: void help()

Description: Displays the instructions manual for users .

### bar ()

Prototype: void bar()

Description: Header bar which prints Name, Item Code, Quantity, and Price as a header.

### welcomeMessage ()

Prototype: void welcomeMessage()

Description: Header with the name of Restaurant.

### invTempl ()

Prototype: void invTempl()

Description: Easy Display of the modified Inventory.

### update\_file ()

Prototype: void update\_file

Description: fstream used for writing the updated inventory external file.