CSCI2020-01 | Algorithm Analysis

Final Project

4/30/18

Restaurant Management System Documentation

Created By:

Michael McGaffick

2018

Table of Contents

[I. Problem Description (Requirement 1) 2](#_Toc512785564)

[II. Reason for Data Structures (Requirement 2) 4](#_Toc512785565)

[Hash Map 4](#_Toc512785566)

[Array List 4](#_Toc512785567)

[III. Test Case Scenarios (Requirement 10) 5](#_Toc512785568)

[Case 1: Removal of Save Files 6](#_Toc512785569)

[Case 2: Successful Inventory Management 8](#_Toc512785570)

[Case 3: Inventory Incorrect Form 10](#_Toc512785571)

[Case 4: Successful Table Management 12](#_Toc512785572)

[Case 5: Incorrect Table Management Usage 15](#_Toc512785573)

[Case 6: Successful Receipt Management 17](#_Toc512785574)

[Case 7: Incorrect Restaurant Management Usage 20](#_Toc512785575)

[Case 8: About Form and Form to Form Buttons 22](#_Toc512785576)

[Case 9: Full Implementation with Command Prompt 26](#_Toc512785577)

[Case 10: Full Implementation with Java IDE Blue J 31](#_Toc512785578)

[IV. User Manual (Requirement 11) 39](#_Toc512785579)

[Startup 39](#_Toc512785580)

[Home Form 39](#_Toc512785581)

[About Form 40](#_Toc512785582)

[Inventory Management Form 41](#_Toc512785583)

[Table Management Form 42](#_Toc512785584)

[Receipt Management 43](#_Toc512785585)

[Save Files 44](#_Toc512785586)

[Error Reference 44](#_Toc512785587)

[Practice Implementation (Demo) 45](#_Toc512785588)

[V. Class Participation (Requirement 13) 46](#_Toc512785589)

[Question 1 46](#_Toc512785590)

[Question 2 46](#_Toc512785591)

[Question 3 46](#_Toc512785592)

# I. Problem Description (Requirement 1)

Write a use case in a few paragraphs clarifying what is the requirement of the problem, what assumptions are made and constraints if any.

For my project, I have decided to do a restaurant management system where users can modify table layout data, inventory data, and keep track of their receipts. My program is designed for small restaurants such as coffee shops and pizzeria where they would only need to have one instance of it running. My project attempts to provide a solution to the problems that arise when managing a small restaurant such as keeping an accurate count of inventory, keeping a precise record of purchase orders to see which items are being purchased, and keeping track of the current status of tables. It also allows for user modification of table layout files and the inventory files, so that items could be managed without needing to run the program for every instance of a change. With the inventory management, an accurate record of current items is kept along with prices. Users would add in restocks or subtract items depending on changes with the menu. With the table management, the current size and names of all tables are saved and the status of them can be viewed at any time. Whenever customers are seated, leave, or a bus boy cleans the table, the user of this program could update the records to reflect that. It will allow for users to see the status of tables in order to seat customers appropriately. Receipt management allows for the user to keep a listing of all items purchased by a customer, the total cost, and they can even save the list for further viewing. This could be used to see what items are selling the most and which are not selling at all. The user could then modify the menu/inventory accordingly. In other words, the data from this project could be used in charts and datasheets to effectively run the business. The name on these receipts do not necessary have to refer to the customer, they could be used as times or table ids as the user sees fit.

My program will have a few assumptions that may not reflect what could potentially happen in real life. The first is that tables have a set maximum in my project. In real life, they could potential be moved or chairs added to them if there is one more guest. My program will have a way to manipulate the table information, but it would not be practically to do this every time that a chair is moved. My program also assumes that it would be working in conjunction with a cashier register as there is only a way to store the current receipts, not actually pay for the items. It also assumes that when an item is removed from a receipt, it was either offered for free or is not salvageable, so it is not added back to the inventory’s quantity without the user manually going back to inventory management.

I have also limited my program with a few constraints. Inventory data is only saved to one specified file. The file can be edited or copied, but there is only one source of a file to be loaded. It is assumed that the inventory of a restaurant is not going to change on a large scale. This limitation does not exist for tables, as any file with proper format can be read. The receipts are also limited in the fact that they cannot be read in. It is assumed that once you are done with the current session, you no longer need to use access the receipts and there will be discarded. However, if you do desire the receipt data, there is an option to save it to a file based on the current system time. The final constraint is that you cannot add items to the inventory unless the quantity is one or more and you cannot edit the quantity without manually going to the file. The quantity will change if the item is bought or added back in as a new instance, however. Items will also remain if the quantity reaches zero and a restock is needed.

# II. Reason for Data Structures (Requirement 2)

Pick an appropriate data structure for your problem – linked list, array, queue, stacks, tree etc. Explain what are the advantages and disadvantages of using this data structure and why you have selected to use the same for your problem (5).

For my project, I have utilized two of the java structures; Maps and Lists. For the map, I used a Hash Map and for the lists, I used Array Lists. In regards to step 2 of the requirements, I will discuss everything separately for both of these structures below.

## Hash Map

I utilized the Hash Map structure for the table map exclusively. The advantage of this was that I could use a nickname for the key while the actual data was the table itself. If I had used a list, I would have had to call the table object to determine the name or have the user remember an index. With the map, I could make it easy for the user to remember a simple nickname. I chose a map for this specific reason within my program. The disadvantage though was keeping track of all the keys, so that the user was not expected to memorize all of them. This required me to utilize a second data structure, which was the Array List.

## Array List

I used the Array List for multiple separate classes in my program including the key list in table map class, the list for the inventory class, and finally the list of all the receipts. The advantage of this data structure was that it provided an easy way to manage a sorted version of the data. The disadvantage of this data structure was that you had to either use a binary search or iterate through all of the elements to find the one you wanted while searching whereas in a map you could just call the key and see if it is present in the map. I chose this data structure for its advantages and because it would be easier to implement rather than a Linked List, which would take unnecessary steps to create since it would require Node manipulation when adding or removing elements along with sorting.

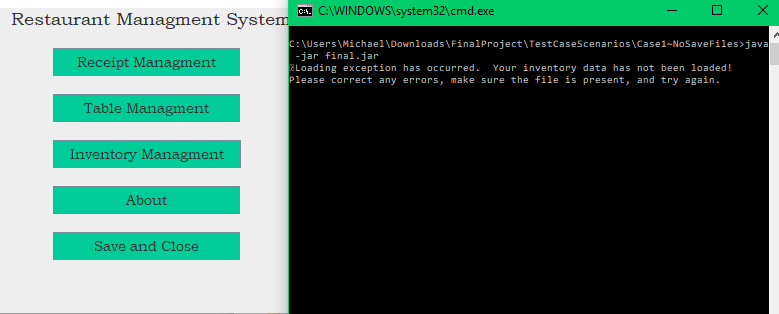
# III. Test Case Scenarios (Requirement 10)

Write a minimum of 10 test case scenarios to test the problem thoroughly including both positive and negative results and make sure correct errors are thrown (10).

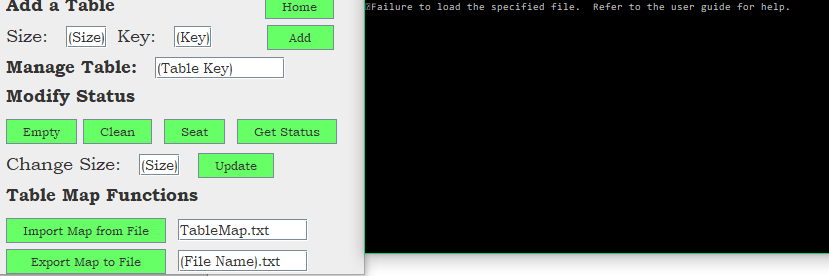
**Please note that I use the run.bat file to run the jar file which uses command prompt in these scenarios. It does not allow for the escape \f and it will appear as a small strange symbol; however, I feel that since it shows previous steps, it will provide better screenshots for test case scenarios. My final case will show the differences with the java IDE blueJ’s output interface. Formatting in these cases will differ from final product due to these conditions.**

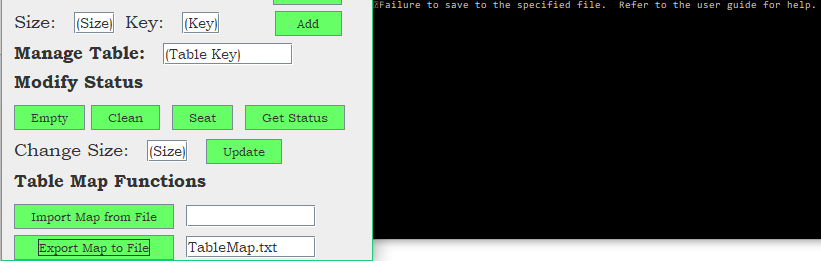
## Case 1: Removal of Save Files

For this case, I simply removed the save files folder and ran the program. I then attempted to try to load non-existent files and save to the folder which did not exist. Inventory would fail to load on startup. Inventory, Tables, and Receipt would fail to save later in the program.

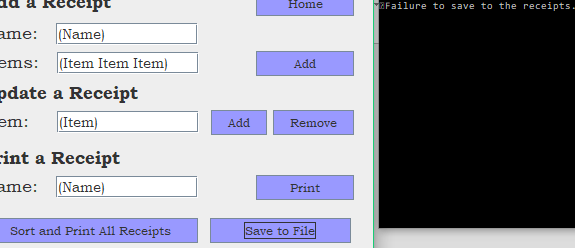
Startup (Inventory Loading): 

Tables Load and Save:

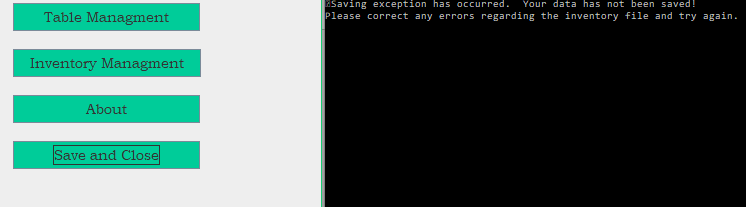




Receipt Saving:



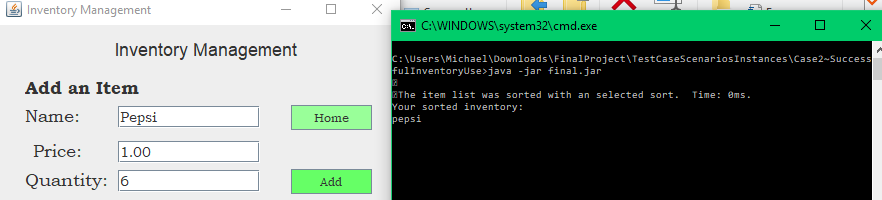
Program Close (Inventory Saving):



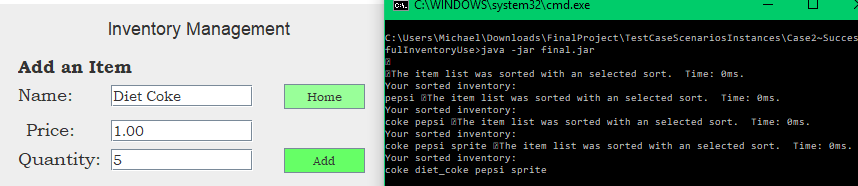
## Case 2: Successful Inventory Management

For this case, I run the inventory management successful. I add a few items, remove one, search for one, and save the files upon exit.

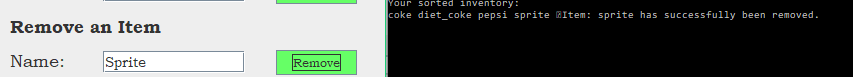
Adding One Item:



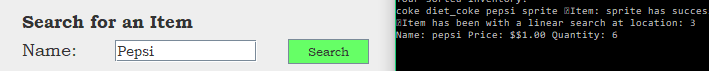
After Adding Three Items:



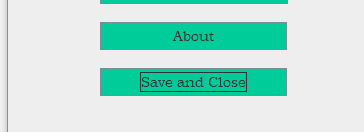
Removing Sprite:

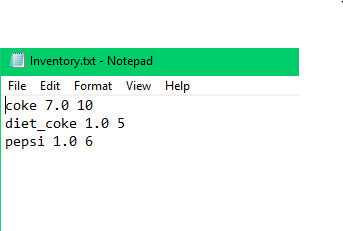


Searching For Pepsi (Note Location Does Not Refer to Index):



Saved File (Saves Upon Home Form Exit):



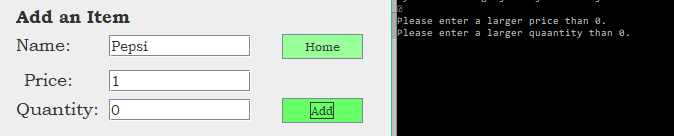


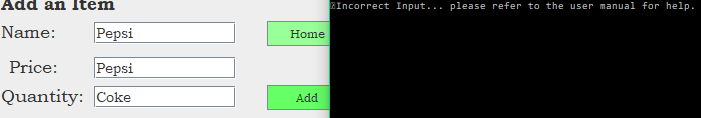
## Case 3: Inventory Incorrect Form

For this case, I use the incorrect format for input throughout the Inventory Management form. I show that the correct errors are thrown in each of the situations.

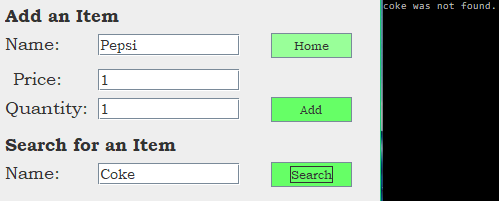
Incorrect Price and Quantity Usage:



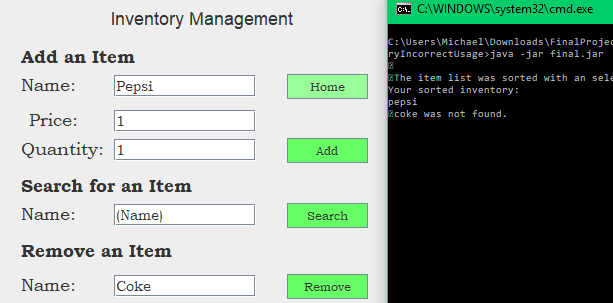




Search Item Not Found:



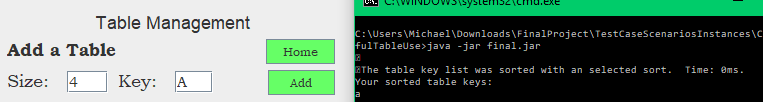
Remove Item Not Found:



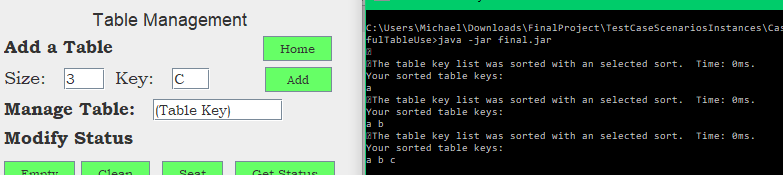
## Case 4: Successful Table Management

For this case, I use the table management form to show a successful run, which includes adding tables, getting their status, modifying their Boolean properties, and importing/exporting to files.

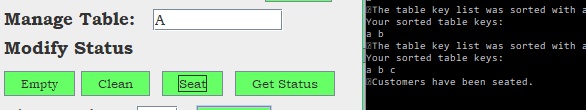
Adding One Table:

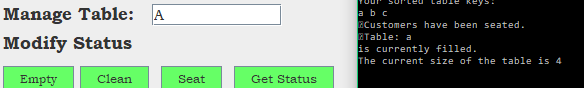


After Adding Two More Tables:

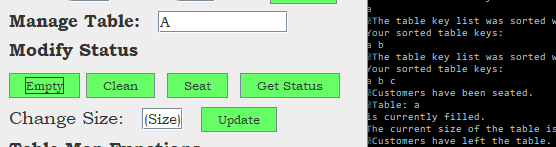


Modifying Table A Now. Seating Someone in Table A:

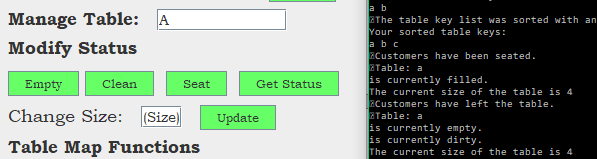


Getting Status of Table A:

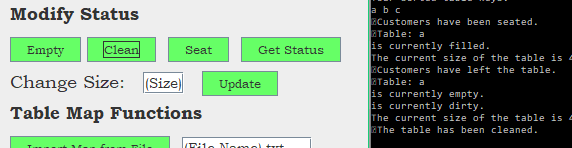
Empty Table A:



Get Status of Table A Again:



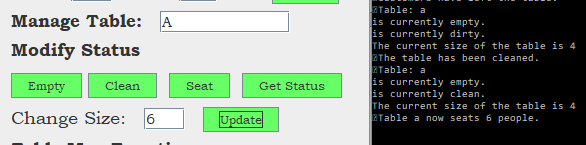
Clean Table A:



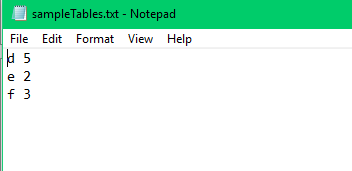
Get Status of Table A for Final Time:



Change Size of Table A from 4 to 6:

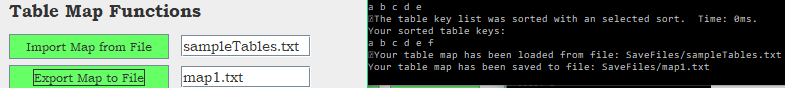


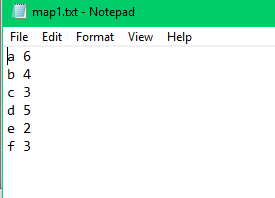
Import Map:





Export tables to File:

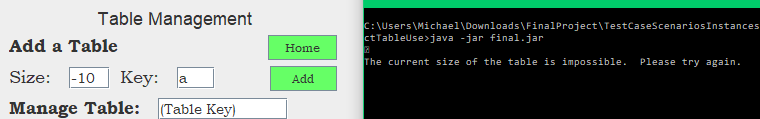


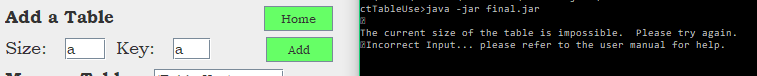


## Case 5: Incorrect Table Management Usage

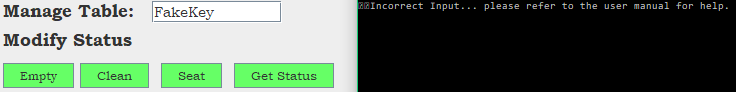
In this case, I run through the table management form again, but this time I use incorrect inputs and usage of the buttons. I ignore saving and loading since that was covered in the first case.

Incorrect Size Format:

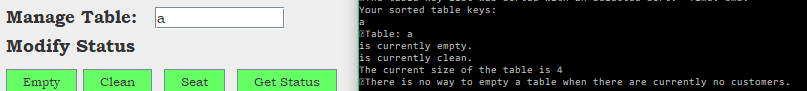




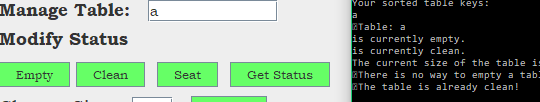
Key Not Found in Modify Table:



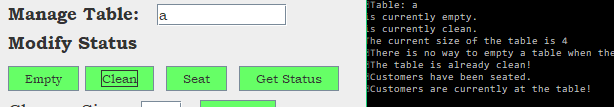
Created a Table with Key of a. Tried to Empty an Empty Table:



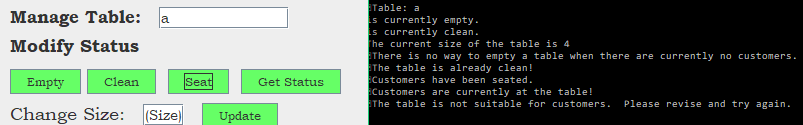
Tried to Clean a Clean Table:



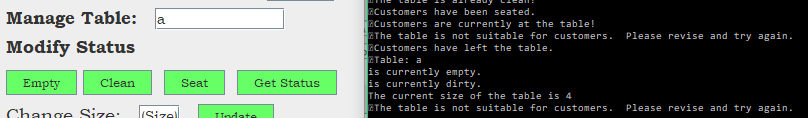
Tried to Clean a Table with Customers:



Tried to Seat Customers at a Table with Customers Already:



Tried to Seat Customers at a Dirty Table:



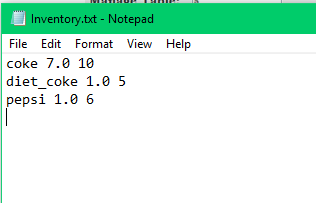
Tried to Use Change Size with Incorrect Size:



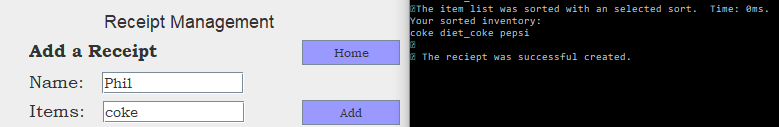
## Case 6: Successful Receipt Management

This case will use the receipt management form successfully. It will show how to add receipts, search for them, remove and add items to them, print all receipts, and export data to a file.

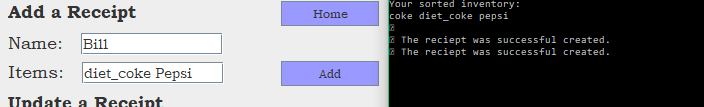
Please Take Note that I am Using the Following Inventory:



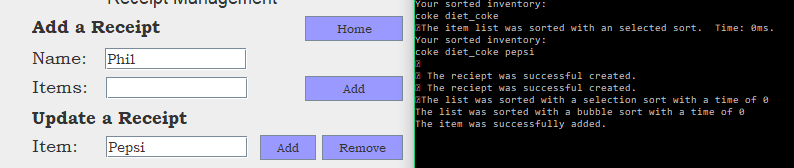
Add a Receipt:



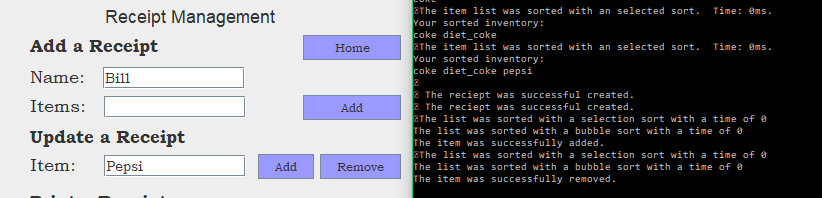
Add Another Receipt with Two Items:



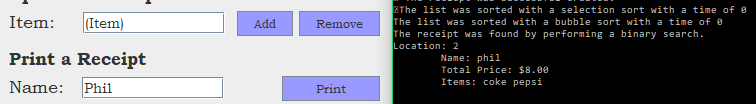
Give a Customer Another Item (Give Phil a Pepsi):

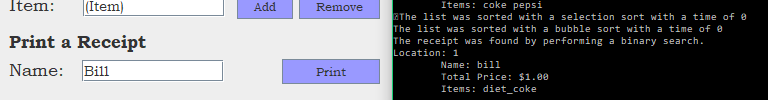


Remove an Item from Customer Receipt (Remove Bill’s Pepsi):

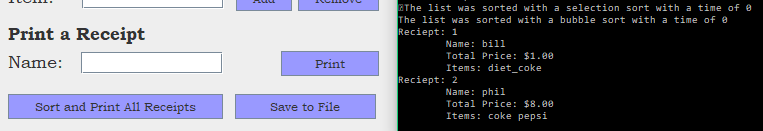


Print A Receipt (First Phil, then Bill):



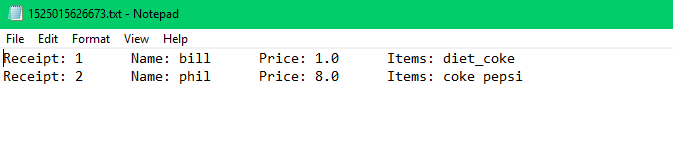


Print All Receipts:



Save to File Based on Current Time in Milliseconds:

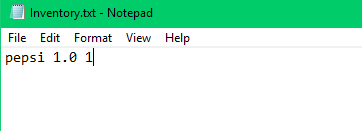




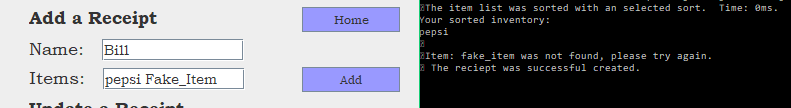
## Case 7: Incorrect Restaurant Management Usage

This case will show the incorrect usage of the input boxes within receipt management. This is specifically referring to using items that have either run out or that do not exist in the inventory. I also try referring to a receipt that does not exist in this instance of the program.

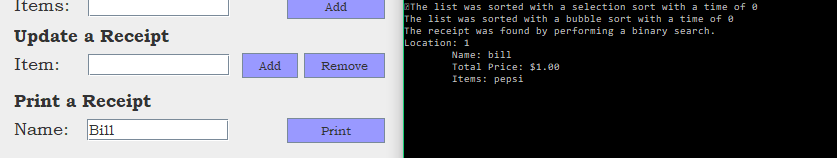
Please notice the new inventory for this case:



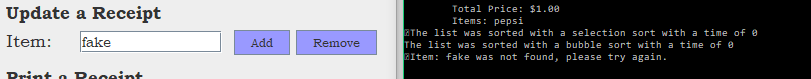
Add a Customer with a Fake Item:



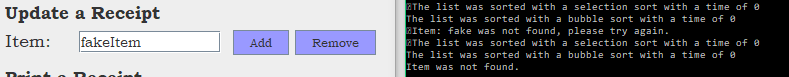
Prints Receipt, Notice Only Pepsi was Included:



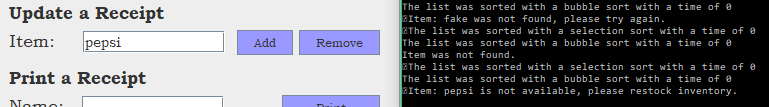
Attempt to Add Fake Item Under Update:



Attempt to Remove Fake Item Under Update:



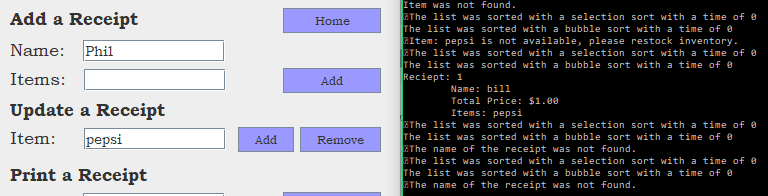
Try to Add Pepsi, but Inventory is Out:



Try Printing a Receipt that Does Not Exist:



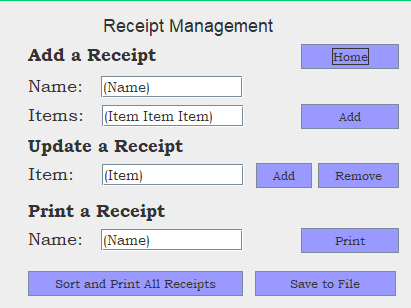
Try Adding Items to a Receipt that Does Not Exist:

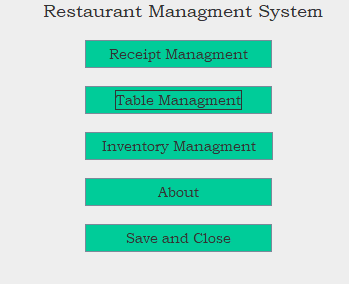


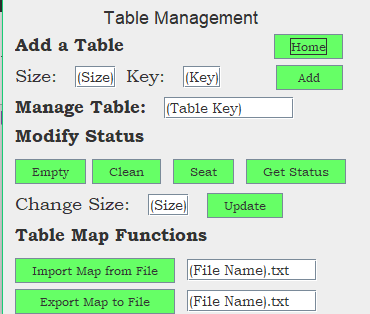
## Case 8: About Form and Form to Form Buttons

For this case, I simply tested the navigation buttons of the forms. This includes about, inventory, receipt, and table to the home form and vice versa. The path I took was home to receipt to home to table to home to inventory to home to about to home.

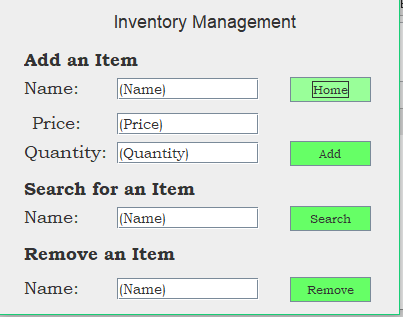




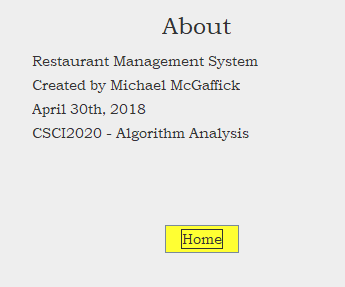


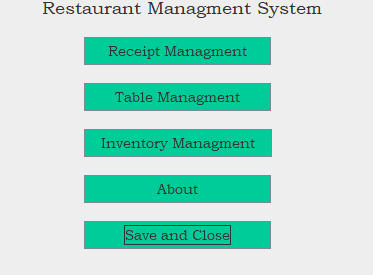








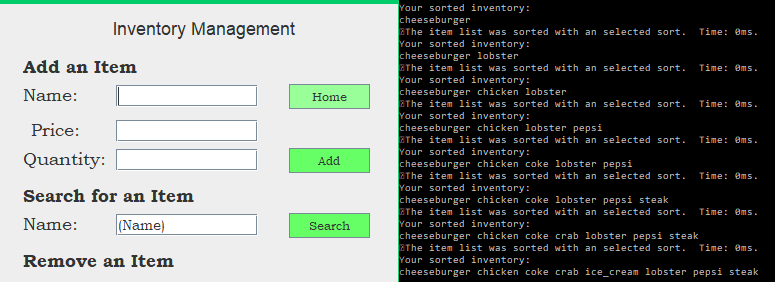




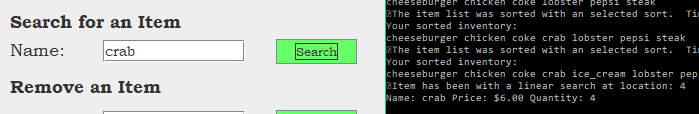
## Case 9: Full Implementation with Command Prompt

This case will use everything with the program successfully. It will show how the system could actually be implemented and used in a small restaurant. This implementation will start with nothing in the save files and it will create the files that case 10 will use.

Items Added to Inventory:



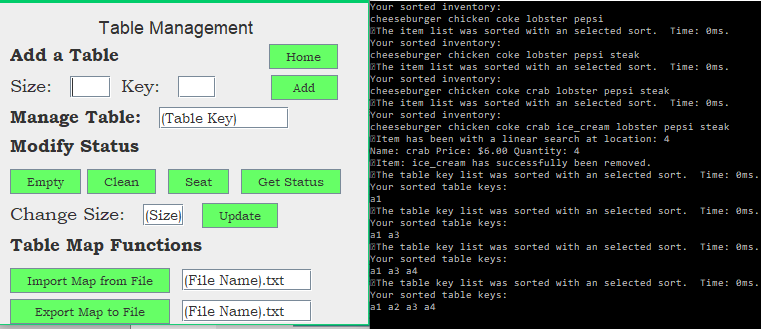
Search for Crab:



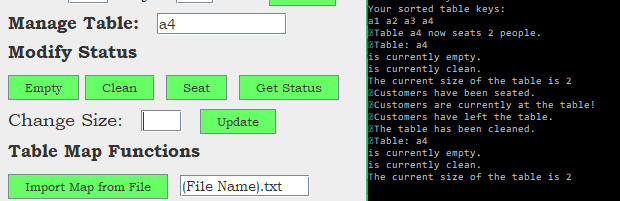
Remove Ice Cream:



Added Tables to the Map:



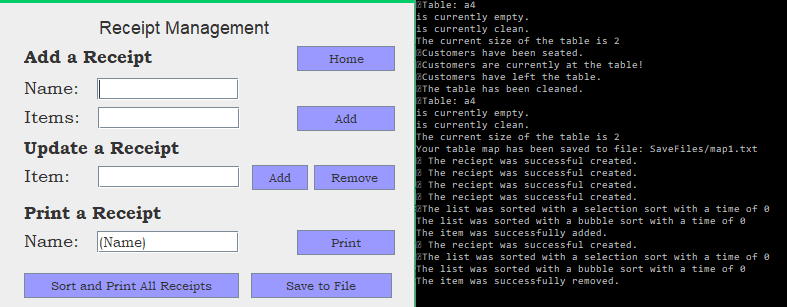
Modified Table A4 to be Size 2, Seated Customers, Tried to Clean Table, Customers Left, and Finally Cleaned Table:



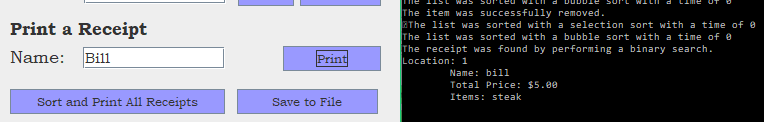
Export Tables to File map1.txt:



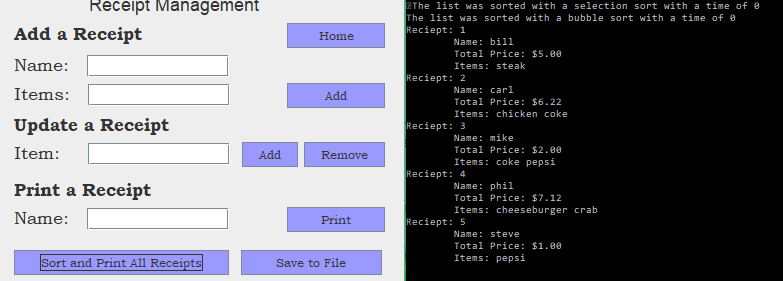
Created Five Receipts And Used Update to Remove and Add an Item:



Prints A Receipt:



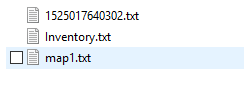
Prints All Receipts:

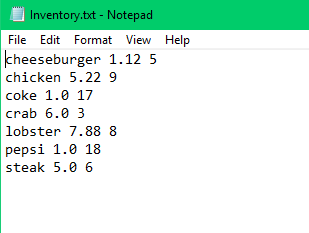


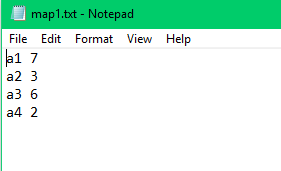
Saves to File:

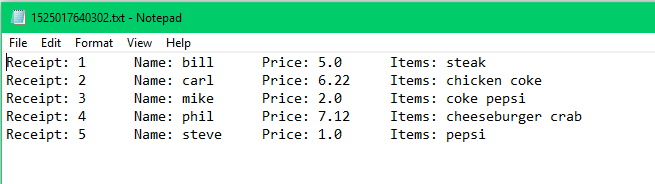


Program now saved and closed. Saved Files:





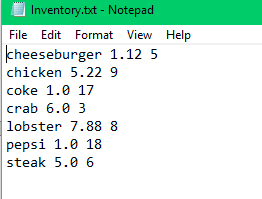


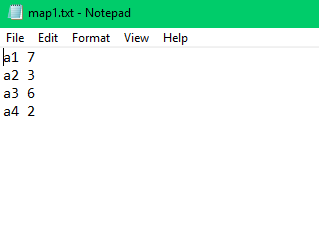


## Case 10: Full Implementation with Java IDE Blue J

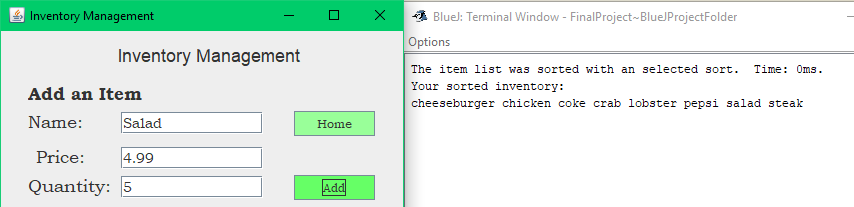
This implementation utilizes the data that was created from the previous case. This case uses the Blue J interface to show how it differs from Command Prompt. I will also be using the steps from this case as my in class demo, with the steps also being listed as the practice implementation in the short user guide.

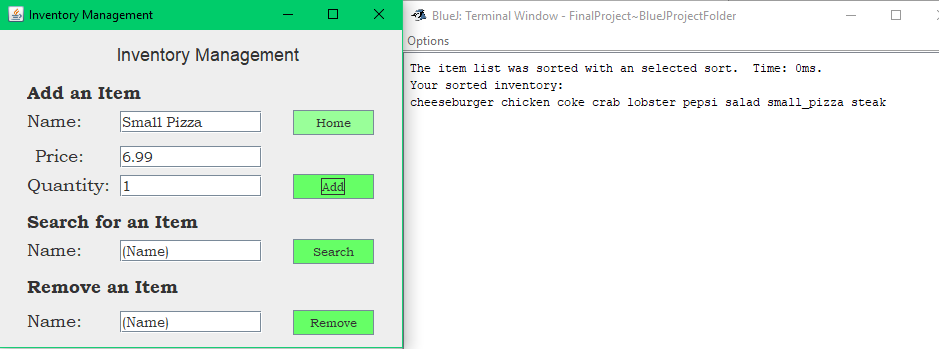
Note the Current Inventory and Map files:



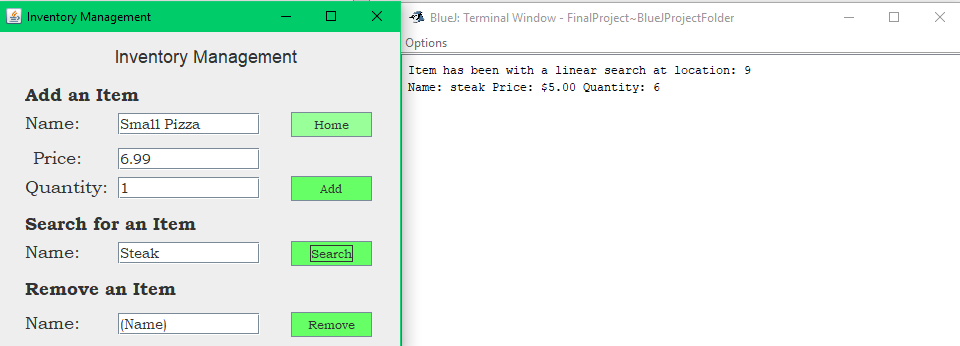


Add an Item and Add a Second Item, but with a Space in the Name:

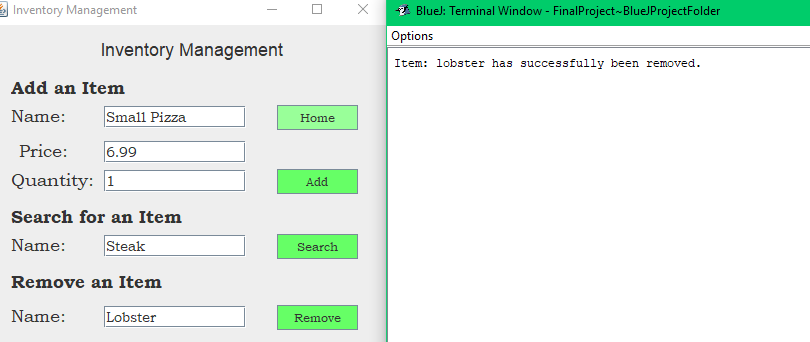




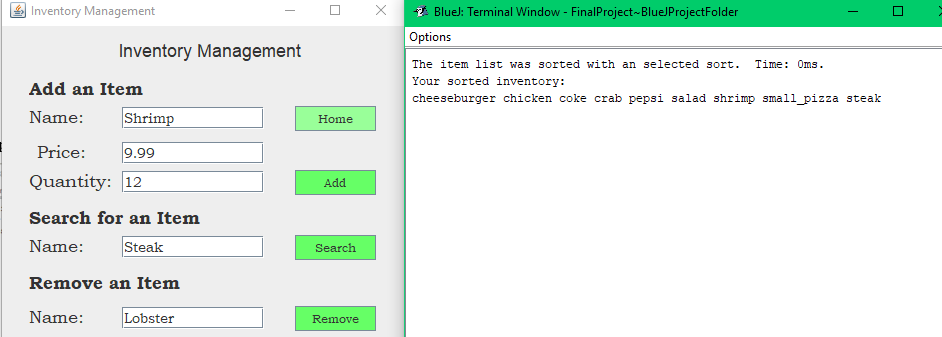
Search For An Item (Steak):



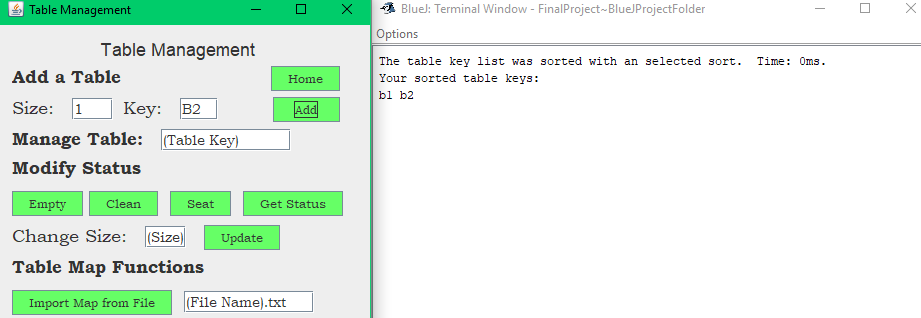
Remove an Item (Lobster):



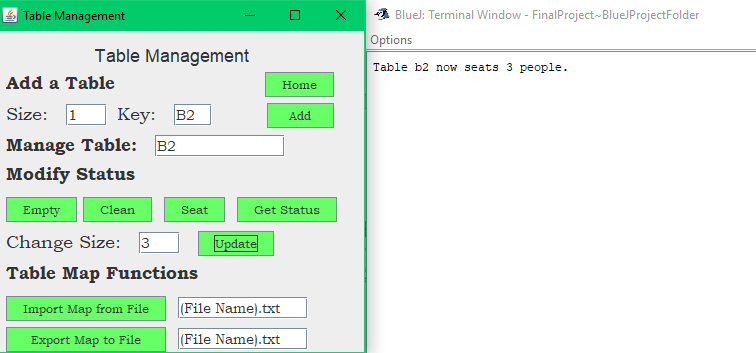
Add an Item (Shrimp):

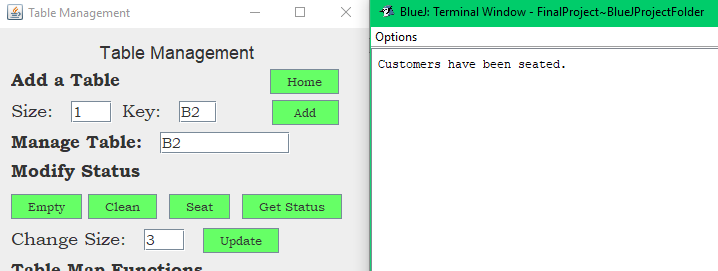


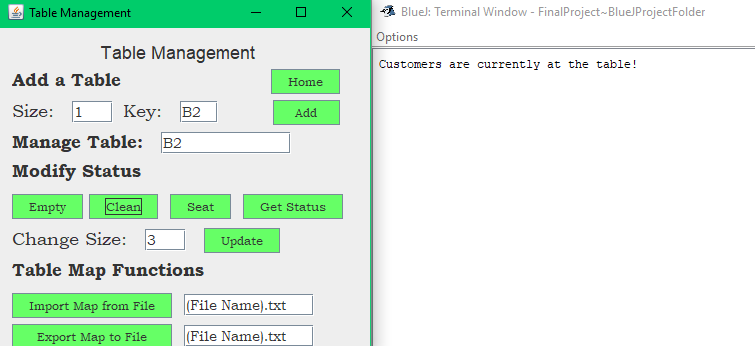
Add Two Tables:

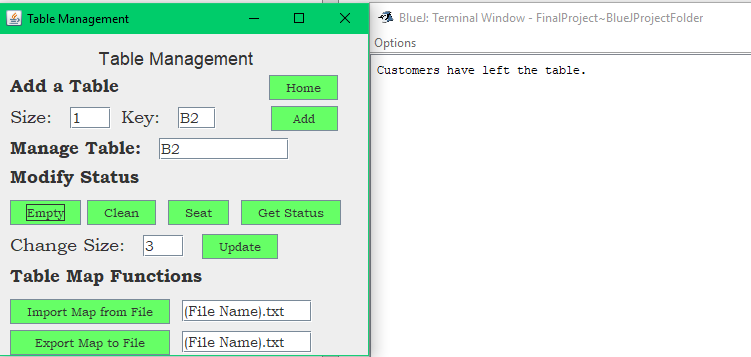


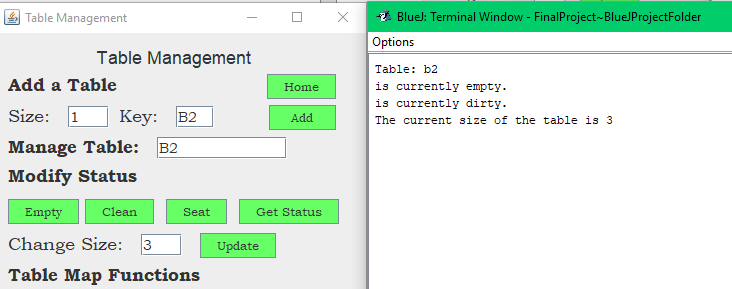
Change Size of Table B2 to 3, Seat Customers, Try to Clean, Empty, Get Status:



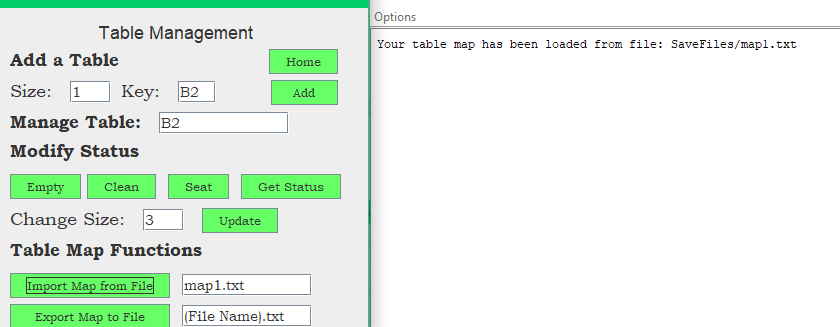


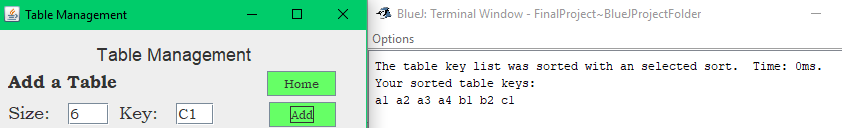




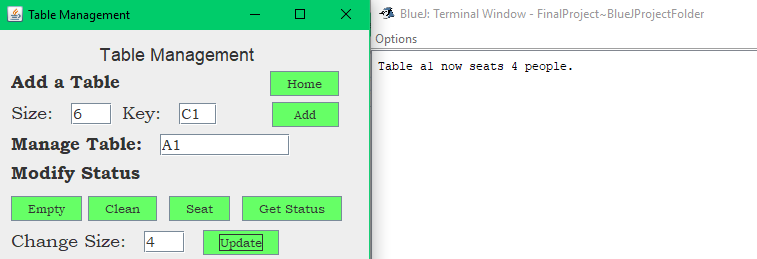


Import map1.txt and Add New Table:

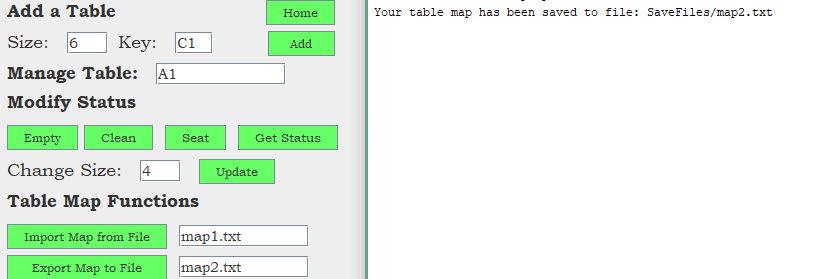




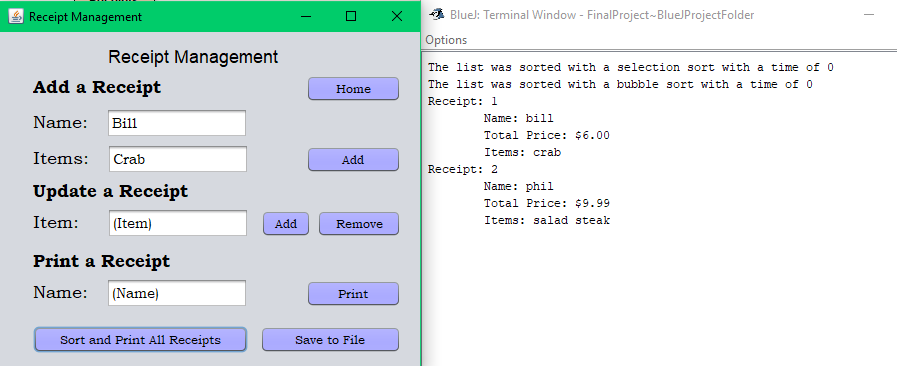
Change Size of A1 from 7 to 4:



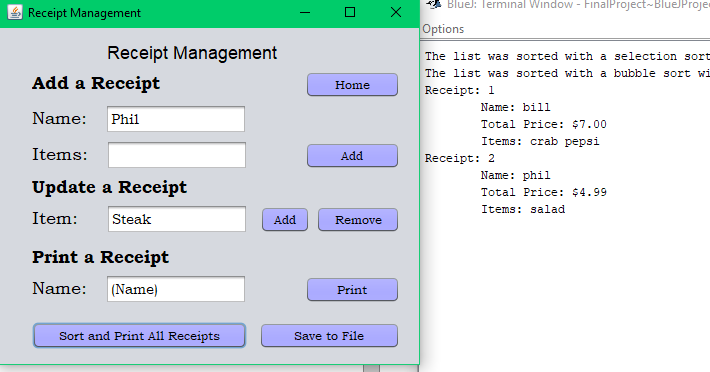
Export to File map2.txt:



Create Two Receipts and Print All:

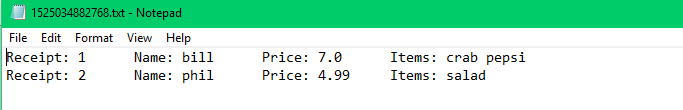


Add Item to First Receipt, Remove Item From the Second, and Print All:

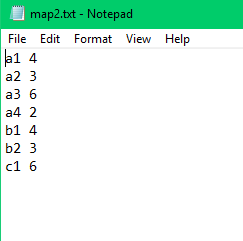


Save To File, Exit Program, and View File:

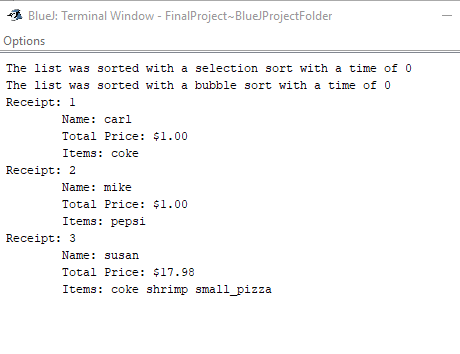




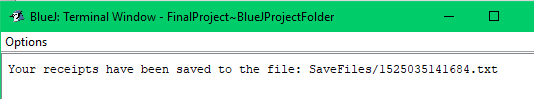
View Newly Created Map File:



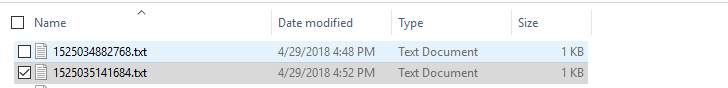
Open Program Again and Create Three More Receipts then Print All:

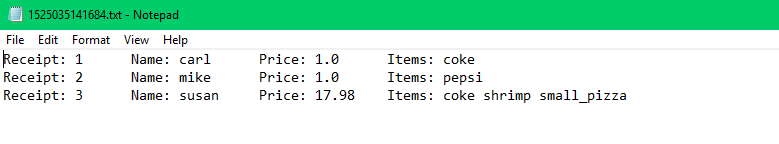


Save to File, Exit Program, and View File:



Note the Order of the Files:



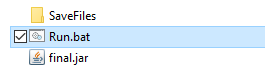


# IV. User Manual (Requirement 11)

Write a short user guide. It can be as simple as 1 page with an intro to your project and how to use it (10).

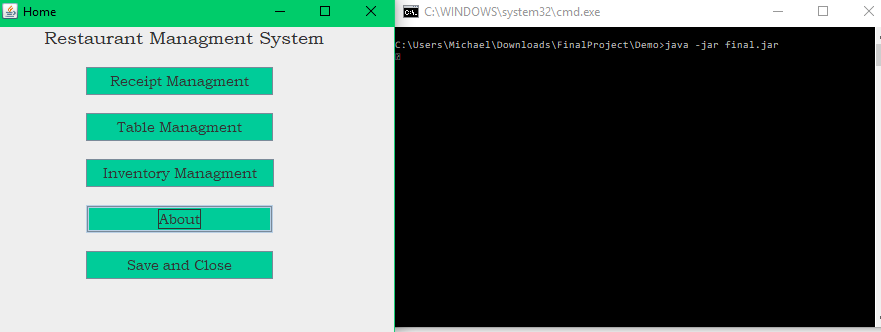
## Startup

There are two options to start this program. The first is to use the java class files and open the project through an IDE such as Blue J or Eclipse. You would then run the Main class’s main method and the program would start at the home menu. The second way is to run the program using the jar file. However, to get the output window, you must use a batch file to run the jar with the code: java -jar final.jar. An alternative to making this batch file is to use the one that is already created called run.bat. Note that if you choose to use the jar file, a small symbol where appear whenever \f is used since Command Prompt does not recognize this symbol. In an IDE, this is where the form would reset to clear the previous information. You will need to have the SaveFiles folder present in order to save and load files (See below for example).



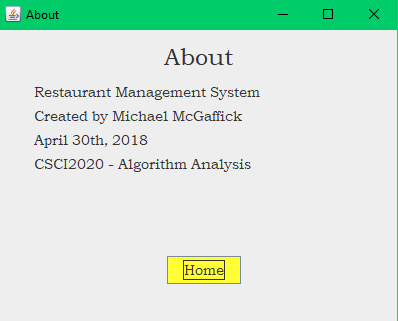
## Home Form

Upon start-up of the program, you will be greeted with the home form and the output window. Please note that this user guide will be referring to the command prompt window as the output console. From the Home Form, you have access to all four of the other forms. New users should first click the about form to learn about the project and how navigation works. The output console will also show when selection sorts, bubble sorts, binary searches, and linear searches are used for your items. The sorts will also be timed and output how long they took. The Save and Close button should be pressed whenever exiting the program (It will be discussed in the Save Files Section).



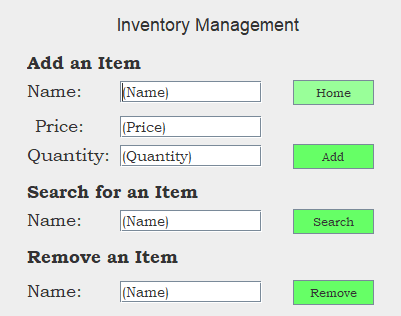
## About Form

The about form show information regarding the program. In practical use, this form should only be visited to see information regarding the program and in most runs, it will not be visited. New Users can visit Table Management or Inventory Management in any order, however, it is imperative to use Inventory Management before Receipt Management.



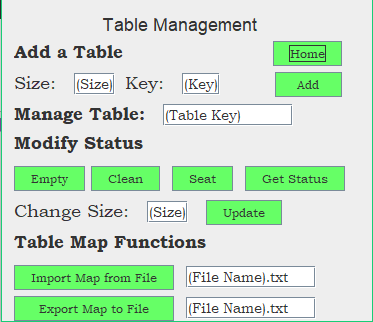
## Inventory Management Form

This form allows for the users to create and store information about items being offered at their restaurant. The name field can either be the name of the item or an item code based on the user’s preference. The price must be a number greater than 0 and the same goes for the Quantity. Spaces can only be used in the name fields on this form, but take note that all spaces will be converted to underscores within the data. Using the Add button, users can add their desired item to the current inventory. Using the search button, you can find the price and quantity of the given item. Removing an item with remove all instances of it… meaning not one from the quantity, but the entire data for the item. Upon start of the program, all previous inventory data will be loaded in and upon saving and closing of the program, all inventory data will be saved. For more on saving and loading, visit the Save Files Section.



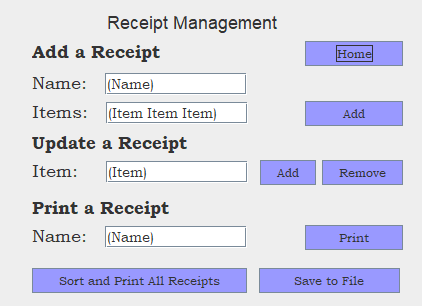
## Table Management Form

The table management form allows for users to manage the table layout of their restaurant. You can add a new table simply by clicking add with a number in size and a nickname/code for the key. The size field must be greater than 0 in both cases. The key can be anything the user desires to nickname their table, but it should be noted that the user should be responsible for maintaining a list of the keys. The program stores the list, but there is no way to check the current list without exporting the map to a file or adding a new table. To modify the status or size of a table, place the key within the manage table field. You can then seat customers, empty the table if the customers leave, and clean the table as you see fit. You can also check the status of current tables. By importing a previous table map, you can add an old layout of tables to the current layout. This will **not** override your current layout, it will add the tables together. Exporting the maps will save the current table layout to the specified file, so that you may either manually edit the file or import it for later usage. Your table data will not be deleted when exporting… it is only cleared upon exiting the program and there is no automatic saving for table layouts. For formatting of import and export statements, please visit the Save Files Section.



## Receipt Management

This form is to manage and store an accurate record of your receipts for further inspection. Please note that this is meant to accompany a cash register as there is no way to pay with this program, it only stores the data. To add a receipt, simply give it a name (It can be a code if you do not wish to use customer’s names) and create a list of items. The item list should have spaces between each of the items and match the names of the Inventory. Please use underscores if you item has a space within the name such as Diet Pop being Diet\_Pop. To update a receipt, enter the name in the name field and then list the item you wish to remove. This item must match what is present on the customer’s receipt otherwise it will not remove the item. Added items must match the inventory and have a quantity greater than 0 to be added to a receipt. To print a receipt, specify the name on the receipt and it will be outputted in the command window. Sorting and print all receipts will output the entire list of receipts with name, price, and items to the output console. Saving receipts to file will create a text file based on the current system time in milliseconds. For further information regarding saving, please visit save files.



## Save Files

Before attempting to start the program, please make sure that the SaveFiles folder is listed in the directory and properly named as shown in the first image of the user guide. All input and output files must be located within this folder. An empty Inventory.txt should also be present within every implementation, otherwise an error will occur during start up. This is error will not stop the program, it will just alert the user that the inventory data has failed to load.

Inventory.txt is the name of the file that inventory data is loaded from and saved to. Upon starting up the program, the file will be read and all data will be added to the inventory. Whenever the program is closed from the main menu, the save and close button, the data will be stored into Inventory.txt once again. If the user desires, this file can be manipulated to add items while the program is not running. The format is name price quantity and names must contain an underscore rather than spaces when appropriate.

Table have import and export too, however, it is done by user manipulation rather than upon program start. Please note that it is recommended to use .txt for table files; however, other untested file types may work successfully. When importing a table, please note that the format should be the key and then the size of the table with a space in between. Spaces should not be used for key names and could potential cause errors to occur when importing the data. These errors will not stop the program, but they will prevent from importing all of your data. To export the data, simply name the file with the extension and the data will be properly formatted. Note that if you used spaces when creating the table names, the export file will not work as an import file.

Receipt has only an export function. Upon clicking save, a new file will be created based on the current system time in milliseconds. The receipts will be listed in alphabetical order and include the name, price, and items ordered. This is for user inspection only, the program does not utilize these files after saving them. There is also no automatic saving for receipts, so it is up to the user to remember to save them.

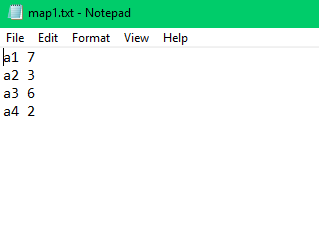
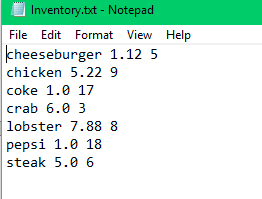
## Error Reference

Sometimes you will receive an error that suggests viewing the user guide for proper usage. For errors that occur outside of saving and loading, please refer to the section that matches the form you were on for proper formatting. For saving and loading, please visit the Save Files section for proper format and usage of the documents. Please note that if there is failure to load a specified file due to improper format of some of the data in the file, it is recommended to close the program with the exit button in the window to prevent the saving of the partial incorrect data.

## Practice Implementation (Demo)

This is an example implementation to get users trained and ready to use the software. This will be my tenth test case. To get started, please locate the Demo folder within the program. Copy and paste the folder somewhere else if you wish to use it again from the default settings.

1. First, navigate to the SaveFiles folder and view the two text documents. Note that the Inventory.txt and the map1.txt should reflect the information below:



1. Use the run.bat file to start the program from the demo directory.
2. Navigate to the about form. Return to the home form. Head to the inventory form.
3. Add Salad Item with whichever price and quantity you wish.
4. Add Small Pizza Item with a space with whichever price and quantity you wish.
5. Search for the item, Steak
6. Remove the item, Lobster
7. Add the item Shrimp with whichever price and quantity you wish
8. Navigate back to home menu and then to table management
9. Add two tables; b1 and b2 with size of 4 and 1.
10. Change b2 to seat 3 people. Click seat, clean, empty, and get status while noting the results of each
11. Import map1.txt and add a new table, c1 with size of 6
12. Change Size of A1 from 7 to 4
13. Export to file map2.txt
14. Navigate back to the home menu and then to receipt management
15. Create two receipts (Bill with crab and Phil with salad and steak)
16. Add Pepsi to Bill’s receipt and remove steak from Phil’s
17. Save to File, navigate back to home and click exit and save.
18. Reopen the program
19. Create three more receipts (Carl with coke, Mike with Pepsi, and Susan with Coke, shrimp, and small\_pizza). Note the underscore with small pizza. Search for Mike and Print All.
20. Save to file, return to home, and click exit and save.
21. Navigate to the Save Files folder
22. View the inventory and note the changes.
23. View the map1 and map2 documents. Note the changes.
24. View the two receipts, noting that they are in order from the time. Note the results.

This concludes the sample implementation of the project.

# V. Class Participation (Requirement 13)

Participation – Actively interact with other students as they present their projects in class. At least 3 questions or suggestions to different projects need to be asked/recommended. This can be per team if you are with a partner. Note this down with obtained answers or with suggestions (5).

## Question 1

Questions Asked: Does the airline ticket save, or does it only print out after the user submits it?

Student Who I Asked the Question to: Bili and Mohammad

Answer Received: Yes, however, it only works in Blue J or a java IDE.

## Question 2

Questions Asked: Does checking the box and clicking confirm changes populated the completed ticket form?

Student Who I Asked the Question to: Johnathon

Answer Received: No, you must click the remove to populate the completed tickets forms.

## Question 3

Questions Asked: When does the program save the data to the file?

Student Who I Asked the Question to: Noah

Answer Received: The program currently does not save. The data loaded is used for the demo

only and is manually coded in.