Cafe Simulator

Lani, Jharell, Jacky, Mariia

Team 4

Team Member Descriptions

Lani

Team Coordinator

Coordinated group meetings and organized division of labor. Responsible for compiling & debugging all members' code to ensure functioning final product

- Assembled all Team Reports, Presentation Poster, and Final Project Files
- Contributed to project menu

Mariia

File I/O, Binary Search Tree

Designed Binary Search Tree, made some function so the traversal can be possible in the code

Jacky

Search & Display Manager

Designed search and display managers incorporating the use of multiple data structures.

- Assisted in creating test cases and performing tests on final build
- Debugged console displayed code and created sample outputs

Jharell

Hashing Methods & Resolution

Created methods involving designing techniques for mapping data elements into Hash Table.

- Assisted with Diagrams
- Debugged sections of the code
- Drafted the Presentation
- Integrated Data Structures together

Summary

Our cafe simulator creates a database of Drink items and offers a virtual platform that mimics a consumer and buyer transaction. Our program implements Hash Tables, Binary Search Trees, and Linear Probe to handle collisions in order to maximize code efficiency & reusability.

Sample Output

Adding Custom Drink

← Intro
← Menu Options

← Option **4** (Searching to see if drink already exists)

← Verifies drink is not yet in menu

← Option **1** (Adding drink and its characteristics)

← Successfully added custom drink

Sample Output

```
============================================================
Please choose option 1-9: 4
============================================================
Enter the name of the drink to search: Lemonade
Name: Lemonade
Type: Juice
Rating: 5
Dairy (Y/N): N
Price: $4.50


============================================================
Please choose option 1-9: 2
============================================================
Enter the name of the drink to delete: Lemonade
Lemonade has been deleted!
============================================================
Please choose option 1-9: 4
============================================================
Enter the name of the drink to search: Lemonade
Lemonade not found!
```

← Option **4** (Searching to see if drink exists)

← Search for “Lemonade”, it’s found, prints description

← Option **2** (Deleting a drink)

← Attempt to delete “Lemonade”, delete successful

← Option **4** (Searching to see if drink exists)

← Confirmed “Lemonade” was deleted

Deleting Custom Drink

Sample Output

```
============================================================
Please choose option 1-9: 3
============================================================
Lemonade added back to menu!
============================================================
Please choose option 1-9: 4
============================================================
Enter the name of the drink to search: Lemonade
Name: Lemonade
Type: Juice
Rating: 5.00
Dairy (Y/N): N
Price: $4.50
```

Undo Delete Custom Drink

← Option **3** (Undo delete previously deleted drink)

← Adds “Lemonade” back to menu

← Option **4** (Searching to see if drink already exists)

← Confirmed “Lemonade” is back in the menu

Sample Output

```
============================================================
Please choose option 1-9: 6
============================================================
Drinks successfully saved to file!
All drinks saved to file!
============================================================
Please choose option 1-9: 5
============================================================
DRINKS MENU
Apple Juice
Cappuccino
Cappuccino
Caramel Apple Spice
Dark Chocolate Mocha
Dragon Drink
Espresso
Flat White
Green Tea
Hot Chocolate
Iced Black Tea
Iced Brown Sugar Oatmilk Shaken Espresso
Iced Caffè Americano
Iced Shaken Espresso
Iced White Chocolate Mocha
Latte
Lemonade
Matcha Crème Frappuccino Blended Beverage
Mocha Frappuccino Blended Beverage
Nitro Cold Brew
Orange Juice
Paradise Drink
Pink Drink
Steamed Milk
Water
White Hot Chocolate
```

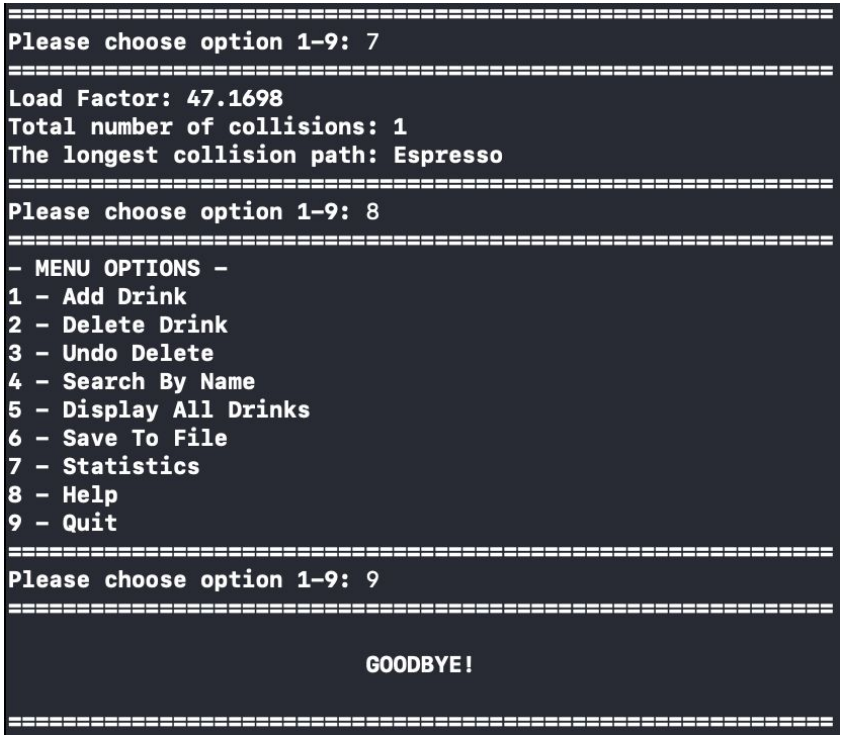
← Option **6** (Saves everything to file and empties stack)
← Confirmation save to file was successful

← Option **5** (Displays all drinks in menu)

← List of all drinks

Save File & View Full Menu

Sample Output



← Option **7** (Shows statistics of program)

← Displays Hash Table stats

← Option **8** (Help shows menu options)

← Menu Options

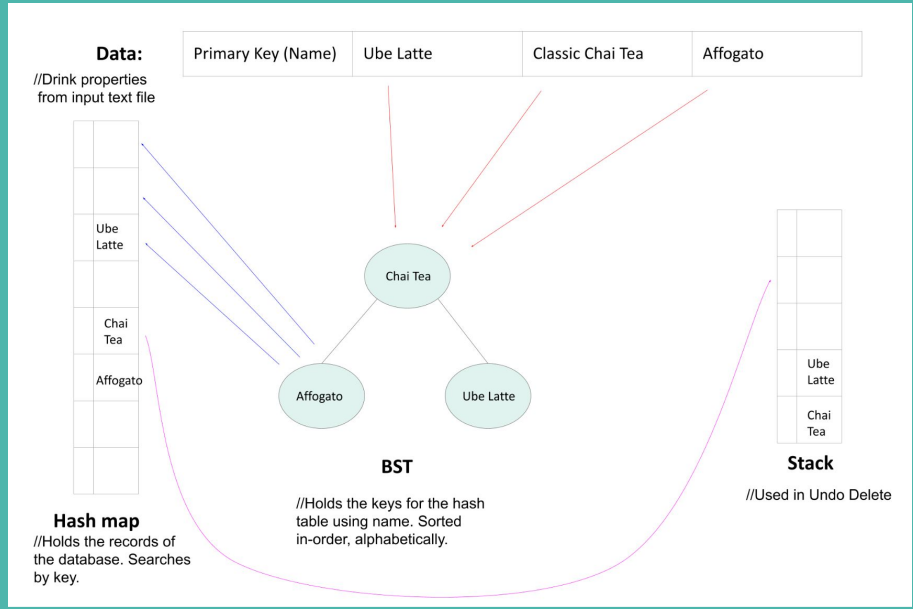
← Option **9** (End program)

Stats / Help / End

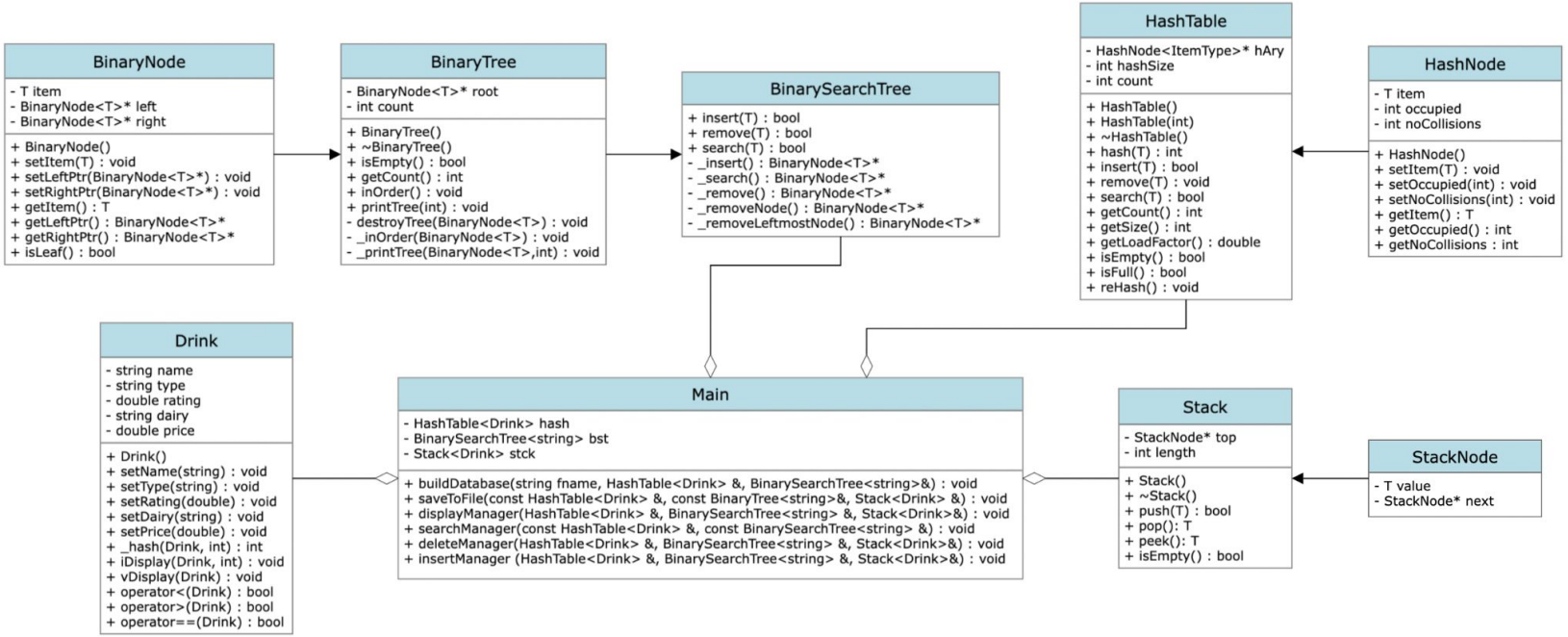
Data Structure Design

Integrates:

Binary Search Tree
Hash Table
Stack



UML Design


BinaryNode
- T item
- BinaryNode<T>* left
- BinaryNode<T>* right
+ BinaryNode()
+ setItem(T) : void
+ setLeftPtr(BinaryNode<T>*) : void
+ setRightPtr(BinaryNode<T>*) : void
+ getItem() : T
+ getLeftPtr() : BinaryNode<T>*
+ getRightPtr() : BinaryNode<T>*
+ isLeaf() : bool
BinaryTree
- BinaryNode<T>* root
- int count
+ BinaryTree()
+ ~BinaryTree()
+ isEmpty() : bool
+ getCount() : int
+ inOrder() : void
+ printTree(int) : void
- destroyTree(BinaryNode<T>) : void
- _inOrder(BinaryNode<T>) : void
- _printTree(BinaryNode<T>,int) : void
BinarySearchTree
+ insert(T) : bool
+ remove(T) : bool
+ search(T) : bool
- _insert() : BinaryNode<T>*
- _search() : BinaryNode<T>*
- _remove() : BinaryNode<T>*
- _removeNode() : BinaryNode<T>*
- _removeLeftmostNode() : BinaryNode<T>*
HashTable
- HashNode<ItemType>* hAry
- int hashSize
- int count
+ HashTable()
+ HashTable(int)
+ ~HashTable()
+ hash(T) : int
+ insert(T) : bool
+ remove(T) : void
+ search(T) : bool
+ getCount() : int
+ getSize() : int
+ getLoadFactor() : double
+ isEmpty() : bool
+ isFull() : bool
+ reHash() : void
HashNode
- T item
- int occupied
- int noCollisions
+ HashNode()
+ setItem(T) : void
+ setOccupied(int) : void
+ setNoCollisions(int) : void
+ getItem() : T
+ getOccupied() : int
+ getNoCollisions : int
Drink
- string name
- string type
- double rating
- string dairy
- double price
+ Drink()
+ setName(string) : void
+ setType(string) : void
+ setRating(double) : void
+ setDairy(string) : void
+ setPrice(double) : void
+ _hash(Drink, int) : int
+ iDisplay(Drink, int) : void
+ vDisplay(Drink) : void
+ operator<(Drink) : bool
+ operator>(Drink) : bool
+ operator==(Drink) : bool
Main
- HashTable<Drink> hash
- BinarySearchTree<string> bst
- Stack<Drink> stck
+ buildDatabase(string fname, HashTable<Drink> &, BinarySearchTree<string>&) : void
+ saveToFile(const HashTable<Drink> &, const BinaryTree<string>&, Stack<Drink> &) : void
+ displayManager(HashTable<Drink> &, BinarySearchTree<string> &, Stack<Drink>&) : void
+ searchManager(const HashTable<Drink> &, const BinarySearchTree<string> &) : void
+ deleteManager(HashTable<Drink> &, BinarySearchTree<string> &, Stack<Drink>&) : void
+ insertManager (HashTable<Drink> &, BinarySearchTree<string> &, Stack<Drink>&) : void
Stack
- StackNode* top
- int length
+ Stack()
+ ~Stack()
+ push(T) : bool
+ pop(): T
+ peek(): T
+ isEmpty() : bool
StackNode
- T value
- StackNode* next

Hash Function

Hash Method:

- Modulo Division
- Fold Shift

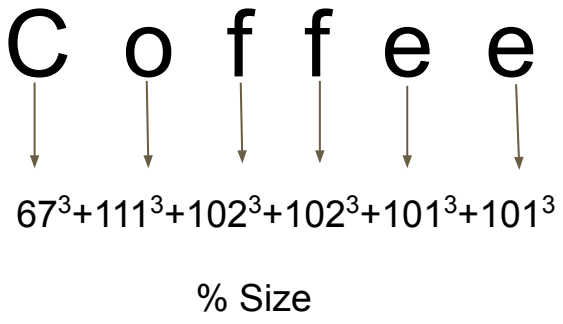
Hash Function:

Extracts the alphanumeric keys from name of drink.

Sums the cubes of the ASCII value of characters followed by modulo division.

Benefit:

- Minimizes the sum being small which minimizes collisions
- Minimizes clustering




```
/*
Hash function: takes the key and returns the index in the hash table
*/
int _hash(const Drink &key, int size)
{
    string k = key.getName();
    int sum = 0;
    for (int i = 0; k[i]; i++)
        sum += k[i] * k[i] * k[i];
    return sum % size;
}
```

Collision Resolution

Linear Probing:

handles a collision by starting at the key's mapped bucket, and then linearly searches subsequent buckets until an empty bucket is found.

```
Next Address =
     (Current Address + 1) MOD listSize
```