Josue Carlito
CSC17B
4/23/2023

STORE FRONT

Introduction:

Title: Store Front

My storefront program emulates a simple store that displays its inventory and allows the user to purchase items. The store I chose to make was a shoe store called “Shoe Odyssey” with different shoes that I was able to think of. At the start of the program, the user first must sign in and add things to their cart and checkout. After they checkout they have the option to continue shopping or exit the application. The data is stored in binary files which keep record of individual user sales information, sign in information, and inventory.

Summary:

Lines of code: ~1000
Variables: ~50
Classes: 6 (Admin, SignIn, InventoryClass, CustomerClass, SalesRecord, ShoppingCart)

The program is implemented with an OOP design and uses binary files to store information. The normal user can simply purchase items and view their inventory. The admin is capable of adding and editing inventory, changing a user’s password and account status, and they are also capable of viewing a user’s total purchases.

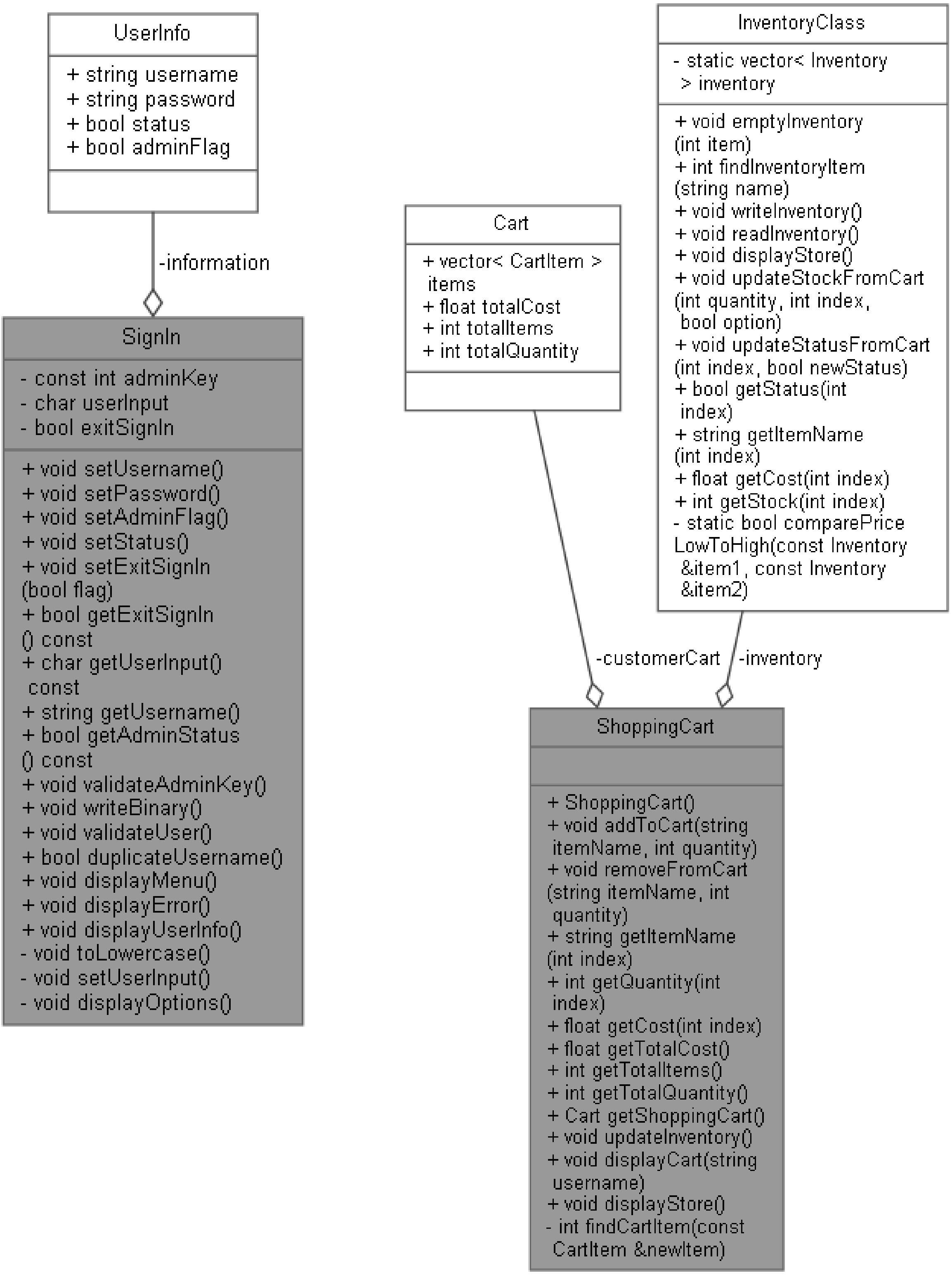
This program in total took me around ~16 hours to code and I decided to code a majority of it procedurally at first. I took this route because I’m not a huge fan of OOP design and I wanted to understand how it would look when simply using structures and functions. Once I got past that, I started converting things over to classes. The first class I chose to write was the sign-in class because it could be used for both my individual projects and my group project. The most difficult part when programming this project would have to be working with binary files and being able to update them and store information. It was challenging but also exciting when I was able to run my program and it would load previous information. Prior to this project, I’ve only coded static projects that reset every time they ran so it was exciting to learn how to create a more dynamic project with memory. The program is prefilled with some purchase history as well that can be viewed.

Description:

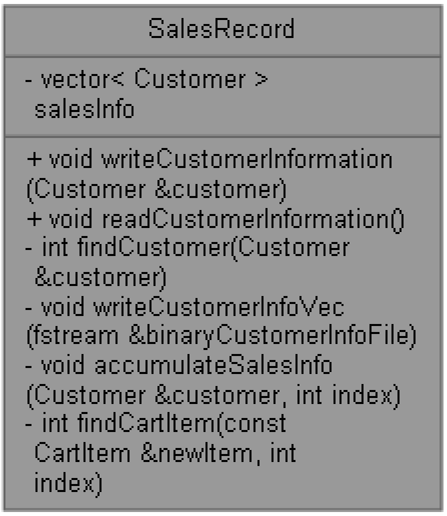
The program is prompt heavy and case sensitive. It requires the user to read each line carefully before they enter an input. The program begins with a sign in page where the user is capable of creating a normal user account, an admin account, or if they have already registered for an account then they can sign in. *To create an admin account the user must enter an admin key which is “1234”* I have already created an admin account with the username and password being “mlehr”. If a user signs in as a normal user they are shown the inventory and then they can add and remove items to their shopping cart and then check out. If a user signs in as an admin, they have the option to choose from inventory options, sales info, and customer information options (user sign in info). The inventory options allow the admin to change stock, add or remove stock, change an item’s name or price, add a new item, and then display the inventory and write the inventory changes. The sales information allows you to look up a user and see what they have purchased in total or view every user’s total purchases. Finally, the customer information options allow you to edit sign in information. You can view all the user’s sign in info, change the user’s password or account status, and then write the changes. Those are the complete admin functionalities.

Documentation:

Doxygen was used on version 12 (the final version). There is a doxygen folder in the repository. Click on the index.html file to be brought to the homepage and then navigate throughout the html pages to view the project's documentation.



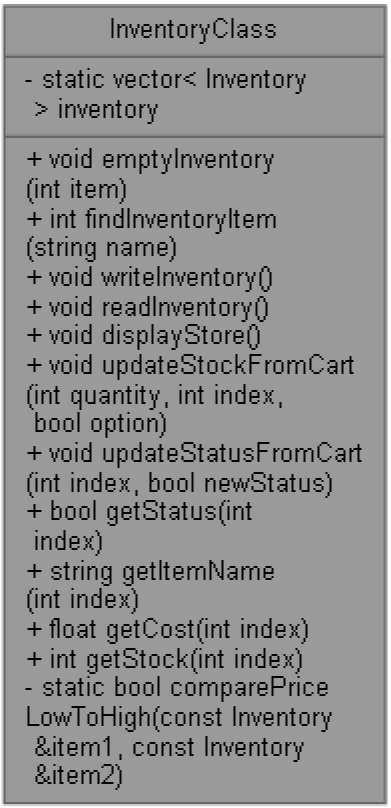
UserInfo
+ string username
+ string password
+ bool status
+ bool adminFlag
-information
SignIn
- const int adminKey
- char userInput
- bool exitSignIn
+ void setUsername()
+ void setPassword()
+ void setAdminFlag()
+ void setStatus()
+ void setExitSignIn (bool flag)
+ bool getExitSignIn () const
+ char getUserInput() const
+ string getUsername()
+ bool getAdminStatus () const
+ void validateAdminKey()
+ void writeBinary()
+ void validateUser()
+ bool duplicateUsername()
+ void displayMenu()
+ void displayError()
+ void displayUserInfo()
- void toLowercase()
- void setUserInput()
- void displayOptions()
Cart
+ vector< CartItem > items
+ float totalCost
+ int totalItems
+ int totalQuantity
InventoryClass
- static vector< Inventory > inventory
+ void emptyInventory (int item)
+ int findInventoryItem (string name)
+ void writeInventory()
+ void readInventory()
+ void displayStore()
+ void updateStockFromCart (int quantity, int index, bool option)
+ void updateStatusFromCart (int index, bool newStatus)
+ bool getStatus(int index)
+ string getItemName (int index)
+ float getCost(int index)
+ int getStock(int index)
- static bool comparePrice LowToHigh(const Inventory &item1, const Inventory &item2)
-customerCart
-inventory
ShoppingCart
+ ShoppingCart()
+ void addToCart(string itemName, int quantity)
+ void removeFromCart (string itemName, int quantity)
+ string getItemName (int index)
+ int getQuantity(int index)
+ float getCost(int index)
+ float getTotalCost()
+ int getTotalItems()
+ int getTotalQuantity()
+ Cart getShoppingCart()
+ void updateInventory()
+ void displayCart(string username)
+ void displayStore()
- int findCartItem(const CartItem &newItem)



SalesRecord

- vector< Customer > salesInfo

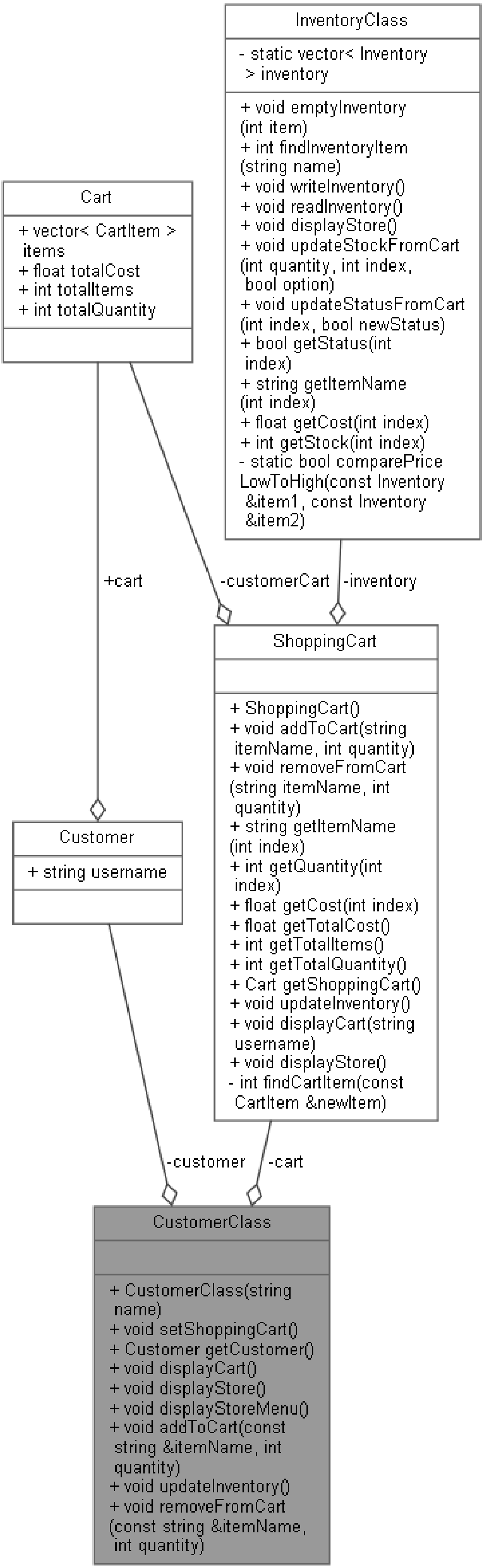
+ void writeCustomerInformation (Customer &customer)
+ void readCustomerInformation()
- int findCustomer(Customer &customer)
- void writeCustomerInfoVec (fstream &binaryCustomerInfoFile)
- void accumulateSalesInfo (Customer &customer, int index)
- int findCartItem(const CartItem &newItem, int index)



InventoryClass

- static vector< Inventory > inventory

+ void emptyInventory (int item)
+ int findInventoryItem (string name)
+ void writeInventory()
+ void readInventory()
+ void displayStore()
+ void updateStockFromCart (int quantity, int index, bool option)
+ void updateStatusFromCart (int index, bool newStatus)
+ bool getStatus(int index)
+ string getItemName (int index)
+ float getCost(int index)
+ int getStock(int index)
- static bool comparePrice LowToHigh(const Inventory &item1, const Inventory &item2)



Cart

+ vector< CartItem > items
+ float totalCost
+ int totalItems
+ int totalQuantity

InventoryClass

- static vector< Inventory > inventory

+ void emptyInventory (int item)
+ int findInventoryItem (string name)
+ void writeInventory()
+ void readInventory()
+ void displayStore()
+ void updateStockFromCart (int quantity, int index, bool option)
+ void updateStatusFromCart (int index, bool newStatus)
+ bool getStatus(int index)
+ string getItemName (int index)
+ float getCost(int index)
+ int getStock(int index)
- static bool comparePrice LowToHigh(const Inventory &item1, const Inventory &item2)

ShoppingCart

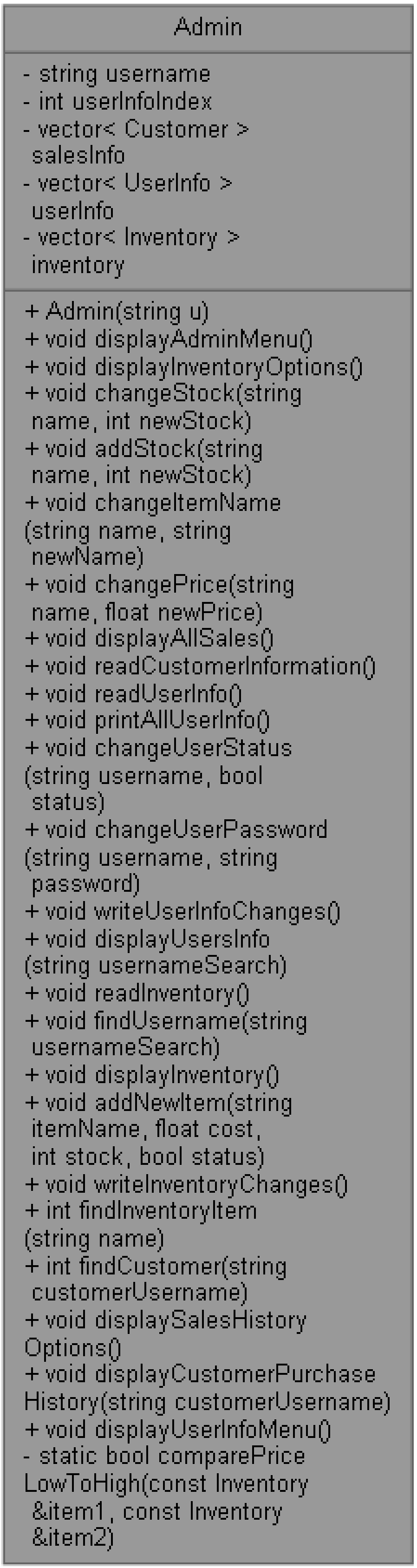
+ ShoppingCart()
+ void addToCart(string itemName, int quantity)
+ void removeFromCart (string itemName, int quantity)
+ string getItemName (int index)
+ int getQuantity(int index)
+ float getCost(int index)
+ float getTotalCost()
+ int getTotalItems()
+ int getTotalQuantity()
+ Cart getShoppingCart()
+ void updateInventory()
+ void displayCart(string username)
+ void displayStore()
- int findCartItem(const CartItem &newItem)

Customer

+ string username

CustomerClass

+ CustomerClass(string name)
+ void setShoppingCart()
+ Customer getCustomer()
+ void displayCart()
+ void displayStore()
+ void displayStoreMenu()
+ void addToCart(const string &itemName, int quantity)
+ void updateInventory()
+ void removeFromCart (const string &itemName, int quantity)


Admin
- string username
- int userInfoIndex
- vector< Customer > salesInfo
- vector< UserInfo > userInfo
- vector< Inventory > inventory
+ Admin(string u)
+ void displayAdminMenu()
+ void displayInventoryOptions()
+ void changeStock(string name, int newStock)
+ void addStock(string name, int newStock)
+ void changeItemName(string name, string newName)
+ void changePrice(string name, float newPrice)
+ void displayAllSales()
+ void readCustomerInformation()
+ void readUserInfo()
+ void printAllUserInfo()
+ void changeUserStatus(string username, bool status)
+ void changeUserPassword(string username, string password)
+ void writeUserInfoChanges()
+ void displayUsersInfo(string usernameSearch)
+ void readInventory()
+ void findUsername(string usernameSearch)
+ void displayInventory()
+ void addNewItem(string itemName, float cost, int stock, bool status)
+ void writeInventoryChanges()
+ int findInventoryItem(string name)
+ int findCustomer(string customerUsername)
+ void displaySalesHistoryOptions()
+ void displayCustomerPurchaseHistory(string customerUsername)
+ void displayUserInfoMenu()
- static bool comparePriceLowToHigh(const Inventory &item1, const Inventory &item2)