Team2 | Catalog Project

CSCI 3920: *Advanced Programming Python & Java*

Professor Pastorino

Due: *December 12th, 2021 by 5:00pm*

Project Final Report

Project Report for our Catalog Project

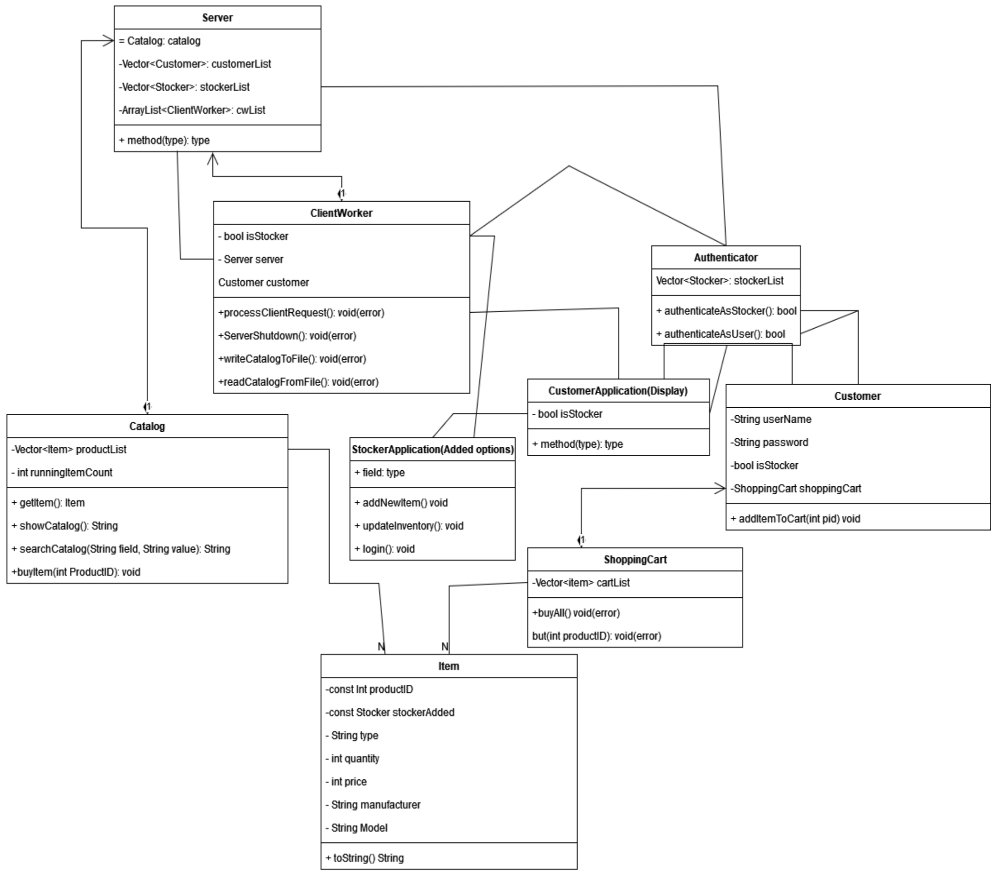
- Team Members :
    - Bryan Heckman
    - Carlos Valdez
    - Joshua Venable
- Project Hosted at: https://github.com/JVenable987/ADVJavaPythonFinalProject

- Project Goals
    - The goal of this project was to create a working Catalog and Shopping Cart interface so that one, a company owner can add items to their respective catalog, and in-turn a Customer can append items to their Shopping Cart in order to receive a total for that specified purchase based on the items that exist in that owner's Catalog. We wanted to approach this project as though an up and coming entrepreneur needed to have a place to have such information and give access to customer's to buy items from said company's catalog. The neat thing about our Catalog is that it allows declarations of type/manufacturer/model of a particular item to be added in the Catalog in case an entrepreneur wishes to expand into different fields of the market while at the same time giving access to customer's to purchase those items once added.
- Use Cases
    - Although not all use-cases were implemented to the interface, they were considered prior to submission of the project as majority was completed, in this section we include use-cases that were initially proposed as well for full transparency.
    - <u>Use Cases</u> - some use-cases were altered based on the direction of the project
    - Use Case #1 - User(Owner) will add/remove items to a catalog
        - Implemented
    - Use Case #2 - User(Owner) will be able see a complete list of the items added to the Catalog

    - Implemented
- Use Case #3 - Customer will be able to be added to the system given a username, a password, and status of stocker/non-stocket
    - Partially implemented (stocker GUI feature not handled/authentication — extra)
- Use Case #4 - Customer will be able to add an item to his/her Shopping Cart using their name, using the item's ID, and a quantity amount of that specific item
    - Implemented
- Use Case #5 - User will be able to search a Customer that exists in the system, and have that customer's Shopping Cart appear with the item they wish to buy, with a total calculated based on the quantity desired
    - Implemented
- Use Case #6- Search for a product in the catalog using a search, and then display the list of products based on search input
    - Not Implemented
- Use Case #7: If the user adds an item that is no longer in the catalog, be prompted by GUI that the item is no longer in the catalog
    - Not Implemented
- Use Case #8: The user can select that they are finished with their order to "check out/finalize"
    - Partially implemented(server implemented but client not)
- Use Case #9: The user can cancel their order entirely, should they decide they do not want anything via "cancelization"
    - Partially implemented(server implemented but client not)
- Use Case #10: The user can see how much their purchase costs, both as products are added/removed from the order, and finalizing the order
    - Partially Implemented
- Use Case #11: Users will be able to type in a search box or a drop down menu and select the item they are searching for
    - Not Implemented
- Use Case #12: User(Owner) will be able to save to file, and load from file
    - Implemented, but could be improved with a database feature (extra)

- Communication Protocol
    - Our project communicates between the client and the server by sending strings of text encoded in utf-8. These text strings contain arguments separated by the pipe operator ‘|’.
    - Client sends: “0|customerName|password|isStocker” Server will create customer record.
    - Client sends: “1||customerName” will checkout the customer on the server
    - “2|customerName|itemId|quantity” will add that item of that quantity to customer’s cart on the server
    - “3|itemName|ItemId|.....” will create new item on the server
    - “4|” will send the whole product catalog to the client
    - “5|customerName” empties the customer’s cart
    - “6|customerName” sends the total of the customer’s cart
    - “7|customerName|itemId|quantity” removes item of quantity from customer’s cart
    - “8|customerName” server will respond with the list of items in a customer’s cart.

- System Design
    - This UML is posted on our Github for reference


Server
= Catalog: catalog
-Vector<Customer>: customerList
-Vector<Stocker>: stockerList
-ArrayList<ClientWorker>: cwList
+ method(type): type
ClientWorker
- bool isStocker
- Server server
Customer customer
+processClientRequest(): void(error)
+ServerShutdown(): void(error)
+writeCatalogToFile(): void(error)
+readCatalogFromFile(): void(error)
Authenticator
Vector<Stocker>: stockerList
+ authenticateAsStocker(): bool
+ authenticateAsUser(): bool
CustomerApplication(Display)
- bool isStocker
+ method(type): type
Customer
-String userName
-String password
-bool isStocker
-ShoppingCart shoppingCart
+ addItemToCart(int pid) void
Catalog
-Vector<Item> productList
- int runningItemCount
+ getItem(): Item
+ showCatalog(): String
+ searchCatalog(String field, String value): String
+buyItem(int ProductID): void
StockerApplication(Added options)
+ field: type
+ addNewItem() void
+ updateInventory(): void
+ login(): void
ShoppingCart
-Vector<item> cartList
+buyAll() void(error)
but(int productID): void(error)
Item
-const Int productID
-const Stocker stockerAdded
- String type
- int quantity
- int price
- String manufacturer
- String Model
+ toString() String
1
1
1
N
N

    - The system has classes in both Java and Python. This is to limit the need to transfer information between the Server and Client, as the client can store the same information as what the server has, and so doesn't have to send a large amount of messages. We were able to correctly implement a Python server with a Java Client. Our System Design had changed throughout the project, as along our programming development cycle needed to add functions, variables, etc

- Current State of Project
    - (Check requirements document, see if anything is missing)
    - The current state of the project is complete although not all features have been implemented, those things that were unable to be implemented include 2-3 use-cases alongside extra graphical user-interfaces that we ambitiously thought were doable when starting the project
    - Classes are not named/implemented quite as they are in requirements document

    - Tested with single Client successfully
    - Supports multi-Client using multithreading and safety using Locks
    - Client GUI using JavaFX
    - Server implemented in Python
    - Communication through Pipe '|' delimiter(From PR#2), for several use-cases
        - Were unable to complete ambitious task of creating administrator classes
        - Were unable to complete ambitious task of creating authentication checking for customer/owner's/stockers

- Challenges Experienced
    - I think we had large dreams of adding extra details not taught in the course and some that were taught to make our project stand out a little more than the other groups. This would include the idea of having multiple GUI's like a login page, to authenticate users based on login, where one route would lead to a GUI specifically designed for the owner of the system, and another route that would lead to a customer's experience when visiting said webpage
    - We discussed this a bit with Javier, but the idea of adding a database to the project, it was clear then that we should focus more on getting it to work versus thinking of all the extra implementation that was possible for the project. Glad we did that versus the latter as it could become too many components that would interact but not really accomplish a specific task
    - Another challenge that we experienced was getting enough time to work together on the project, and for people to explain what they intended to have happen with the sections of code in the project. While we all had our own branches, we didn't collaborate together enough, and things got very messy in the project as a result of that.