UWE Bristol | University of the West of England

The British College

Submitted by : Albin Pokharel

CS AND DF

RE-ASSESMENT

MODULE: Programming with C++

Module Tutor: Sukant Kumar Sahu Sir

As part of our final project, we were tasked with creating a stock analyzer that would store and analyze the client's data and produce useful results. I first created a step-by-step guide on how I was going to do this job in order to complete the work. The design underwent numerous significant steps, each of which is described below.

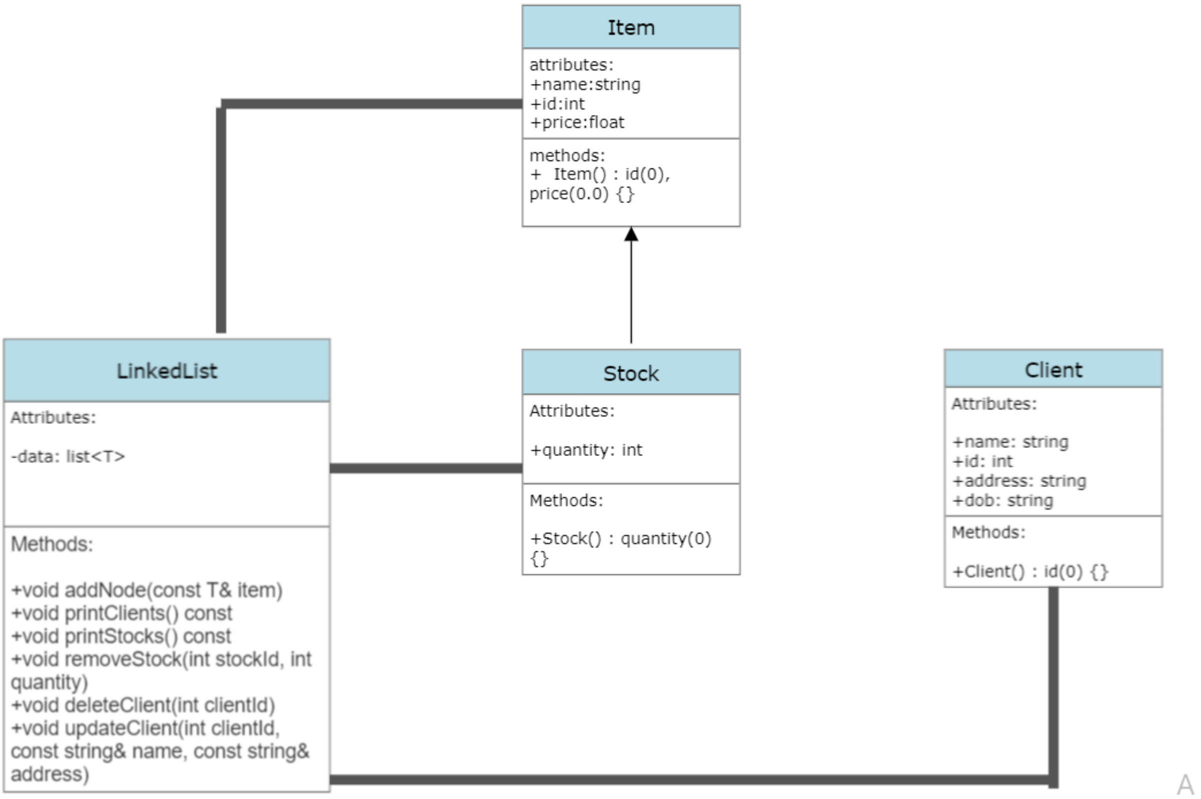
DATE: 2023/8/1

I started by creating an algorithm on paper to get started. I identified the key classes and functions required for the program after carefully reading and understanding the question multiple times. Then, taking into account the need for inheritance and linked lists, those components were ordered in serial order. In order to get insight into how I might improve the effectiveness of my program, I also tried to include the necessary reasoning and literary devices in the article.

The design's UML needed to be created at this point. I created a crude representation of the classes, their members, and the relationships between them in order to construct UML. I reduced repeated functions and member variables in my design after making numerous failed attempts. Then they were linked together so that the design process may lead to a fully functional computer code.

DATE: 2023/1/24

UML DIAGRAM OF CLASSES


Item
attributes:
+name:string
+id:int
+price:float
methods:
+ Item() : id(0), price(0.0) {}
LinkedList
Attributes:
-data: list<T>
Methods:
+void addNode(const T& item)
+void printClients() const
+void printStocks() const
+void removeStock(int stockId, int quantity)
+void deleteClient(int clientId)
+void updateClient(int clientId, const string& name, const string& address)
Stock
Attributes:
+quantity: int
Methods:
+Stock() : quantity(0) {}
Client
Attributes:
+name: string
+id: int
+address: string
+dob: string
Methods:
+Client() : id(0) {}
A

I was prepared to begin the coding portion of my project after finishing the UML design. I selected VS Code as the project's IDE in accordance with the specifications. I developed a C++ software that builds a linked list of nodes, each of which has details about a customer, including their stock list, buy history, and stock sold. A new node can be added to a linked list, its contents can be printed, a node can be updated with a given client ID or deleted with a specified client ID, among other techniques. In a different "deleted" linked list, the removed nodes are added. Each table's data (Client, Stock), as well as the linked list's nodes, are represented by classes in the program (LinkedList). Other standard C++ libraries used by the program include string and iostream.

DATE: 2023/8/5

The LinkedList class, which has a head pointer that points to the first node in the list and a deletedHead pointer that points to the first node in a separate "deleted" linked list, is used to implement the linked list.The "deleted" linked list is intended to keep track of nodes that have been removed from the main list.linked list. For manipulating the linked list, the LinkedList class provides several methods, including:

addNode: accepts Client, StockList, PurchaseTable, and StockSold objects and creates a new node in the linked list with the given information. printList: prints the data held in each node after iterating over the linked list's nodes.

updateNode: This function accepts a client ID and new data, searches for the node associated with the client ID, and changes the node's information if it is located.

deleteNode:With the help of the deleteNode function, a client ID can be entered to find a specific node, remove it from the linked list, and add it to the "deleted" linked list.

A linked list data structure is implemented using a custom LinkedList class in the int main() part of the given code. The class contains several methods, including addNode, printList, updateNode, and deleteNode.

The addNode method is used to create a new Node object, assign values to its properties, and link it to the head of the list. The printList method iterates over the nodes in the list and prints their information. The updateNode method is used to update the information of a node with a given client ID by searching for the node and updating its information. The deleteNode method is used to delete a node from the linked list by searching for the node with the given client ID, removing it from the list, and adding it to a "deleted" linked list. The program executes through the main function's conclusion before terminating abruptly without a return statement.

All in all, the offered application is a handy and effective tool for building and managing linked list data structures, and it provides a wealth of functions and features that are both practical and beneficial.

I searched the internet for resources to help me get through these challenges. I searched through a variety of sites, including forums and online tutorials, to find answers to the issues I was having. Although these references undoubtedly helped me in some ways, they ultimately fell short of providing me with the thorough insight and information that I required to make my program appear and operate as a genuine one.

After completing most of the work,I had a little help from my friends and seniors . They greatly aided in my comprehension of the assignment and gave me the advice and information I need to complete it. I was able to get through the challenges that had been stopping me from finishing my program with their assistance.

CONCLUSION

The provided C++ program demonstrates a stock management system that efficiently handles client and stock information using object-oriented principles. Through inheritance and templated data structures, it captures and manages client names, addresses, stock details, and quantities. The user-friendly menu interface enables adding, viewing, modifying, and removing entries. Exception handling ensures resilience. The program showcases the power of encapsulation and polymorphism, offering a versatile solution for hospital tasks. It serves as a blueprint for future enhancements and showcases the practical application of object-oriented programming in real-world scenarios.