

Assignment 1

by Charlie Keirl

Submission date: 10-Jan-2024 11:12AM (UTC+0000)

Submission ID: 221325643

File name: 41750_Charlie_Keirl_Assignment_1_532657_1131055909.docx (641.09K)

Word count: 2319

Character count: 11329



L5 - ADV SOFT DEV & OOPS (ACCB5011) – ASSIGNMENT 1

Charlie Keirl (2103411)

Contents

Introduction 2

Task 1 2

User Guide 2

Implementation/ OOP Features..... 4

 The Classes 4

 Inheritance and Polymorphism..... 7

Testing..... 8

Task 2 11

References 12

Introduction

Within this assignment, I will be taking you through the program that I have decided to create out of the five options given to us via the brief as well as a section containing the certificate for the “C++Essentials Course” via Cisco Networking Academy. The program type I decided to create was an Inventory Management System based around a fruit and vegetable store. The program allows a user to view current products in stock, add and remove products as well as change the quantity of an already existing product within the inventory.



Task 1

User Guide

Below is a quick user guide to my program. There is also a video tutorial which can be found here: [Inventory Management System Run Down.mp4](#)

When we run the program, we are faced with the user type menu. This menu allows the user to be able to select what type of user they are. Depending on which user type is selected will determine what happens next in the program.

```
Welcome to Charlie's Fruit and Veg Store!  
  
Please select your user type:  
1. Staff  
2. Guest  
Enter your choice: _
```

If the user selects option 2 “Guest”, it will show the user what items are currently within the inventory and how many are in stock. If there are no items in stock, it will tell the user that.



```
Please select your user type:  
1. Staff  
2. Guest  
Enter your choice: 2  
ID: 1, Name: pineapple, Quantity: 3  
ID: 2, Name: apple, Quantity: 2  
ID: 3, Name: banana, Quantity: 4564
```

If the user selects option 1 “Staff”, it will ask the user to enter the password which will give the user access to the staff menu. The password to enter the new menu is “carrots” and once the password has been entered the staff version of the menu will appear.

```
Enter your choice: 1  
  
Welcome! Please enter the correct password to access the Inventory Management System: carrots  
  
Login successful! Welcome!  
  
Charlie's Fruit and Veg Staff Menu  
1. Add a product to the system  
2. Remove product from our inventory  
3. Update a product's quantity  
4. Display all products  
5. Return to user type selection menu  
  
Enter your choice:
```

Here, the user has five options to choose from with the first being to add a new product to the system. When the user selects option 1, it will ask the user to assign an ID number to the new item. Then it will ask for a name for the item as well as the quantity of said item. It will then let the user know if the item has been successfully added to the inventory and will then take the user back to the staff menu.

```
Enter your choice: 1
Enter a new product ID: 1
Enter a new product name: banana
Enter the product quantity: 3

Product added successfully.
```

If the user selects option 2 of the menu, they will be asked to enter the ID number assigned to the item within the inventory. It will then let the user know that the task has been completed and will then display the staff menu once again.

```
Enter your choice: 2
Enter the product ID to remove from the inventory: 1

Item removed successfully.
```

The third option allows the user to be able to update the quantity of an item currently in the inventory. When selected, the program will ask the user to enter in the ID of the product to be updated as well as the new quantity the item will be at. The program will then let the user know if the task has been successful and will then take them back to the staff menu.

```
Enter your choice: 3
Enter the product ID to update how many we have in stock: 1
Enter new quantity: 4

The product quantity you selected has updated successfully.
```

Just like with the guest option, option 4 of the staff menu will allow the user to see all the items within the inventory including their item ID, name, and quantity. It will then display the staff menu again.

```
Enter your choice: 4
ID: 1, Name: apple, Quantity: 3
ID: 2, Name: banana, Quantity: 45
ID: 3, Name: pear, Quantity: 6
```

Option 5 of the staff menu will simply take you back to the user selection menu and if the user wishes to gain access to the staff menu again, they are required to enter the password once again.

```
Enter your choice: 5
Welcome to Charlie's Fruit and Veg Store!

Please select your user type:
1. Staff
2. Guest
Enter your choice:
```

Implementation/ OOP Features

The Classes

When it comes to programs, classes are an essential aspect of your code. Classes are one of the essential aspects of C++ to be able to add object orientation to your code. They help to encapsulate data within your program which aids in readability and organisation. Classes also allow your program to make use of inheritance and polymorphism. [1]

Within this program, I have created four different classes. These classes are “item”, “product”, “inventory” and “UI”.  4

The “item” class is used as the base class for this program and is used to represent the items that are/will be stored within the inventory. This class contains four components: two attributes (“id” and “name” which are used to assign the ID and name of the item added to the system), a constructor (“item(int _id, const string& _name) : id(_id), name(_name) {}” which is used to initialise the ID and name attributes whenever a new item is created), two get methods (“getName()” and “getId()” which are used to access the assigned values of the ID and name of the items) and finally a pure virtual function (“virtual void display() const = 0;” which is used as a placeholder to be overridden in any derived classes. [2]

```
public:
    //Items info
    item(int id, const string& name) : id(id), name(name) {}

    int getId() const { return id; }
    string getName() const { return name; }
    virtual void display() const = 0;
```

The “product” class is similar to the “item” class however with some additional functions. The class utilises inheritance by inheriting items from within the “item” class. The “product” class also contains an attribute called “int quantity” which represents how many of a certain product there is within the inventory. We also have the constructor “Product(int _id, const string& _name, int _quantity) : item(_id, _name), quantity(_quantity) {}” which is used to initialise objects within this class with an ID, name and a total quantity. This is where it takes from the “item” class to set a name and ID to the product.

```
//Product info
Product(int id, const string& name, int quantity) : item(id, name), quantity(quantity) {}
```

We also have member functions within this class such as “getQuantity()” and “updateQuantity()” which are used to see what the quantity of the product is and to also modify the quantity of the product. There is also an override function used. The reason this is implemented is so that the “display” function in the “product” class can override a virtual function that can be found in the “item” class called “display()”.  5

```
//Updating product quantity
void updateQuantity(int newQuantity)
{
    quantity = newQuantity;
}
```

The “Inventory” class is used to store and manage all of the items within the inventory including quantity, adding, removing and even displaying. Within this class there is a private member. This member is called “vector<Item*> items” and it is used as a pointer to point to objects within the “item” class. There are also four member functions. “addItem(Item* item)” is used to point to an item within the “item” class and to then add it to the inventory.

```
//Adding something to the inventory
void addItem(item* item)
{
    items.push_back(item);
}
```

“removeItem(int itemId)” is used to locate an item stored within the inventory with the same item ID as the one entered and then removes it from the inventory.

“updateItemQuantity(int itemId, int newQuantity)” is used within the program to update the quantity of the selected product. It looks for an item ID matching the one the user has entered and then updates the quantity of the selected product with the number the user has entered. However, if no item with that ID was found, an error message is displayed to let the user know that there is no item with that ID.

```
//Updating the quantity of the item in stock
void updateItemQuantity(int itemId, int newQuantity)
{
    for (auto& item : items) {
        if (Product* product = dynamic_cast<Product*>(item))
        {
            if (product->getId() == itemId)
            {
                product->updateQuantity(newQuantity);
                cout << endl << "The product quantity you selected has updated successfully." << endl << endl;
                return;
            }
        }
    }
    cout << endl << "The product ID you have entered was not found. The quantity has not updated." << endl << endl;
}
```

The final member function within the “Inventory” class is “displayAllItems() const” which is used to display to the user all the items stored within the inventory. Just like with the quantity function, this one also displays a message to the user if there are no items currently in the inventory.

```
//Displaying all the items within the inventory
void displayAllItems() const
{
    if (items.empty())
    {
        cout << endl << "Sorry, but there are currently no items in stock!" << endl << endl;
    }
    else
    {
        for (const auto& item : items)
        {
            item->display();
        }
    }
}
```

The “UI” is also utilised within this program as it is used for the user interface and manages all that comes under this including user input and the menus. The “UI” class has two private members. The first is “Inventory inventory” which is used to store items and to manage

their operations. The second is "string password" which is used to store the password required to login to the staff menu (which in our case is "carrots").

```
private:
    Inventory inventory;
    string password = "carrots";
```

There are a lot of member functions within this class because of all the different actions it undertakes. The first function is "showUserTypeMenu()" which is used to display the very first menu that the user will see which will ask them to select what type of user they are.

```
//Function to display start user menu
void showUserTypeMenu()
{
    cout << "Welcome to Charlie's Fruit and Veg Store!" << endl << endl;
    cout << "Please select your user type:" << endl;
    cout << "1. Staff" << endl;
    cout << "2. Guest" << endl;
}
```

The next function is "userTypeOptions(int option)" which is used to then process the option that the user selects within the user selection menu. Depending on what option the user selects, the corresponding process will be executed. The next two functions coincide with each other. The first one is called "showStaffMenu()" and the second "stafMenuOptions(int option)". The first one displays the staff menu to the user with all the options the user can select and the second function allows the user to be able to execute their chosen option.

```
//Function to display the staff menu
void showStaffMenu()
{
    cout << endl;
    cout << "Charlie's Fruit and Veg Staff Menu" << endl;
    cout << "1. Add a product to the system" << endl;
    cout << "2. Remove product from our inventory" << endl;
    cout << "3. Update a product's quantity" << endl;
    cout << "4. Display all products" << endl;
    cout << "5. Return to user type selection menu" << endl << endl;
}
```

Then, the next three functions are to do with what the user wishes to do in the menu. These are "addProduct()" which asks the user for the details of the new product to add to the inventory, "removeProduct()" which allows the user to remove a product from the inventory by asking the user for the product ID and "updateProductQuantity()" which will allow the user to enter the updates quantity of a product by asking them to provide the ID of the product they wish to choose.

```

//Function to add a new product to the inv
void addProduct()
{
    int id, quantity;
    string name;

    cout << "Enter a new product ID: ";
    cin >> id;
    cout << "Enter a new product name: ";
    cin >> name;
    cout << "Enter the product quantity: ";
    cin >> quantity;

    Product* newProduct = new Product(id, name, quantity);
    inventory.addItem(newProduct);
    cout << endl << "Product added successfully." << endl << endl;
}

```

Then there is the “login()” function which asks the user to enter in the password and then it compares it to the stored password discussed above. If the password matches then the access is granted.

```

//Function for password checker
bool login()
{
    string inputPassword;
    cout << endl << "Welcome! Please enter the correct password to access the Inventory Management System: ";
    cin >> inputPassword;

    if (inputPassword == password)
    {
        cout << endl << "Login successful! Welcome!" << endl;
        return true;
    }
    else {
        cout << endl << "You have entered an incorrect password. Access denied." << endl;
        return false;
    }
}

```

Inheritance and Polymorphism

This code makes use of inheritance which is when one class (the child class) takes properties from another class (the base/parent class). It enables a program to tidy itself up without having to repeat any code and can also be used to determine whether or not a certain member of the class is it be public or private when inherited. [3]

Within my own program, I have indeed used inheritance. This inheritance is present in two classes: the “item” class (our base/parent class) and the “product” class (our child class). The “product” class inherits from the “item” class by extending the “item” class with the new attribute of “quantity” as well as implementing the pure virtual function found in said class.

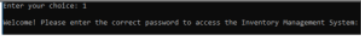
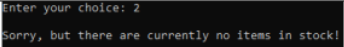
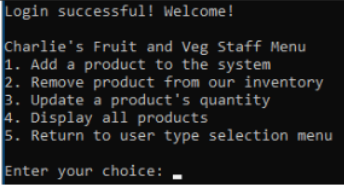
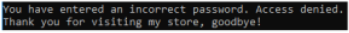
Polymorphism is also used in this program. Polymorphism is the ability to reuse code by making one single function and making it so that it has multiple uses within your program. It allows your code to interact with multiple different objects, even if they are not within the same class but they must share a base class. [4]

Within my own program, the use of polymorphism is present. Within the “item” class we have the pure virtual function called “display()”. Whenever the pointers are used in this class, the “display()” function is called. Depending on what type of object is used will depend on what the function does. Another example of polymorphism used within the

program is within the “inventory” class. A vector of pointers is used within this class called “vector<item*> items;” which allows the program to be able to store pointers for the “item” objects. With the use of polymorphism, these points can then store more than one piece of information such as the product name.

Testing

Below is a testing table for my program.

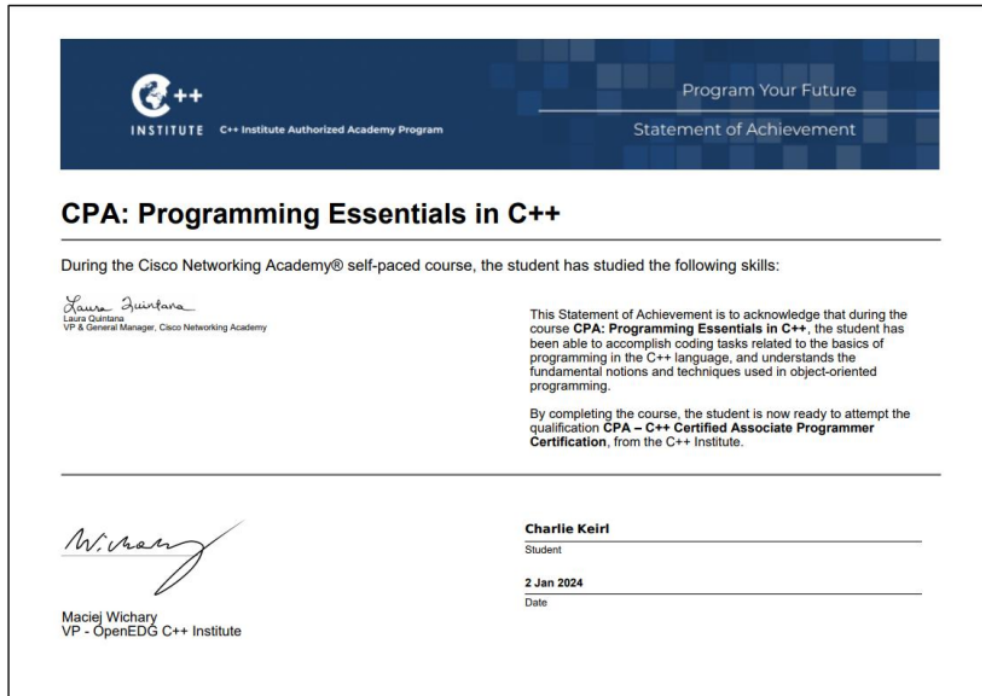
Test Case	Expected Outcome	Actual Outcome	Screenshot	Pass/Fail
User selection (staff)	When the user selects staff, the system should ask for a password to access the menu.	After staff was selected, the log in menu appeared where I was asked to enter the password.		Pass
User selection (guest)	When the user selects guest, the current inventory should appear.	After selecting guest, the current inventory appeared but as there was no stock it showed a message saying there was no stock.		Pass
Login – correct	After entering the correct password, the staff menu should appear.	After typing in the word “carrots”, the staff menu appeared.		Pass
Login – incorrect	After entering a password	After typing in an incorrect		Pass

	incorrectly, an error message should appear.	password, the system let the user know that the password was wrong.		
Add product	When selecting the add product option, it should ask the user to type in the ID, name and quantity of the new product.	After selecting option 1, the program asked for the ID, name and quantity of the new product.	<pre> Enter your choice: 1 Enter a new product ID: 1 Enter a new product name: apples Enter the product quantity: 3 Product added successfully. </pre>	Pass
Remove product	When selecting the remove product option, the program should ask for the ID of the product you wish to remove.	After selecting option 2, the program asked for the product ID to remove and then removed successfully.	<pre> Enter your choice: 2 Enter the product ID to remove from the inventory: 1 Item removed successfully. </pre>	Pass
Update quantity	The program should ask the user for the ID and the new quantity of the product.	After selecting option 3, it asked for the ID and the new quantity of the product I wanted to update.	<pre> Enter your choice: 3 Enter the product ID to update how many we have in stock: 1 Enter new quantity: 7 The product quantity you selected has updated successfully. </pre>	Pass
Display all products	When the user selects option 4, it	After selecting option 4, the IDs,	<pre> Enter your choice: 4 ID: 1, Name: apples, Quantity: 7 ID: 2, Name: beans, Quantity: 3 ID: 3, Name: grapes, Quantity: 5 </pre>	Pass

	will display all the products in store including their ID, name and the quantity.	names and quantity of all products were shown.		
Exit staff menu	Selecting option 5 should take the user back to the user selection menu.	After selecting option 5, I was taken back to the user selection menu.	<pre> Enter your choice: 5 Welcome to Charlie's Fruit and Veg Store! Please select your user type: 1. Staff 2. Guest Enter your choice: █ </pre>	Pass
Guest menu updates	Adding new products while in the staff menu should then be displayed if the user then goes to the guest menu.	After entering a few products to the inventory, I went back to the guest menu and it had been updated with the products I had entered.	<pre> Please select your user type: 1. Staff 2. Guest Enter your choice: 2 ID: 1, Name: apples, Quantity: 7 ID: 2, Name: beans, Quantity: 3 ID: 3, Name: grapes, Quantity: 5 </pre>	Pass
Invalid option	If a user enters an invalid option in the menu, does an error message appear?	After entering 6 into the staff menu, an error message appeared stating that that was not a valid option.	<pre> Enter your choice: 6 You have not selected a valid option. Please try again... Charlie's Fruit and Veg Staff Menu 1. Add a product to the system 2. Remove product from our inventory 3. Update a product's quantity 4. Display all products </pre>	Pass

Task 2

Below is my certificate of undertaking the 'C++Essentials' Course via Cisco Networking Academy with <https://www.netacad.com/portal/learning>



Here is a link to download a PDF of that certificate: [CharlieKeirl-C Summer2023-certificate.pdf](#)

References

- [1] Anonymous Anonymous C++ importance of classes and objects | C++ | cppsecrets.com. C++ importance of classes and objects | C++ | cppsecrets.com. Available: <https://cppsecrets.com/users/73841099712197110107103117112116575764103109971051084699111109/C00-Importance-of-Classes-and-Objects.php>.
- [2] Anonymous Anonymous Inheritance `virtual` functions, C++ FAQ. Inheritance `virtual` functions, C++ FAQ. Available: <https://isocpp.org/wiki/faq/virtual-functions>.
- [3] Anonymous Anonymous Inheritance in C++. Inheritance in C++. 2017. Available: <https://www.geeksforgeeks.org/inheritance-in-c/>.
- [4] Anonymous (-07-28T19:00:00+00:00). *Beginner's Guide: Understanding Polymorphism in C++*. Available: <https://www.udacity.com/blog/2021/07/understanding-polymorphism-in-cpp.html>.

Assignment 1

GRADEMARK REPORT

FINAL GRADE

80 / 100

GENERAL COMMENTS

Hi Charlie,

In-depth feedback is provided within the document, and a summary is presented here. Scores for individual sections are accessible through the rubric.

Areas of appreciation:

Comprehensive User Guide including the user type menu, staff login process, and various functionalities.

The testing section includes a well-organized table with various test cases, expected outcomes etc

areas of improvement:

In real-world applications, secure password practices should be emphasized.

references should be from creditable sources

I hope these comments are helpful.

PAGE 1

PAGE 2

PAGE 3



Comment 1

Good start.



Comment 2

clear differentiation of tasks based on the type of user

PAGE 4



Comment 3

the expected type of input could have been mentioned in the menu for clarity

PAGE 5



Comment 4

highlighting classes from the code is a good approach however a brief definition of the terms could have added credits



Comment 5

Implementing a maximum value for the inventory and incorporating checks during each update process to ensure items stay within the limit could enhance the system.

PAGE 6

PAGE 7

PAGE 8

PAGE 9



Comment 6

well done

PAGE 10

PAGE 11

PAGE 12

PAGE 13

GRADING FORM: DC-ASD

CHARLIE KEIRL

80

LITERATURE REVIEW

with IEEE referencing



3

KNOWLEDGE&UNDERSTANDING SKILLS

use of OOPs features



14

COGNITIVE&INTELLUCTUAL SKILLS

Testing &Error handling



16

PRACTICAL SKILLS

Working program Demo video



19

TRANSFERRABLE SKILLS

User Manual, document style



3

CISCO CERTIFICATE



25