Oliver Dodwell - 18002691

Unit 15 Assignment 2

Table of Contents

Order Handler Design (P3)

Class order_handler
Private: -item_names : string array -item_prices : float array -item_quantity : integer array
Public: +add_order(name, price, quantity) +add name to item_names +add price to item_prices +add amount to item_quantity

Main
Order_handler handler; handler.add_order (“Coffee”, “4.5”, “1”)

Order Handler Implementation (P4)

Entering data for two different orders

```
-== Options ==-

1. Add an order
2. Retrieve an order
3. Change an order
4. Get an order's total
5. Get the total value of all orders
6. Exit

Input: 1

Enter Item Name (Replace spaces with underscores): wool_sweater
Enter Item Price: 99.99
Enter Quantity: 3

        Order Number: 1
        Total: 299.97 pounds

--------------------------------------------------

-== Options ==-

1. Add an order
2. Retrieve an order
3. Change an order
4. Get an order's total
5. Get the total value of all orders
6. Exit

Input: 1

Enter Item Name (Replace spaces with underscores): jeans
Enter Item Price: 65
Enter Quantity: 2

        Order Number: 2
        Total: 130 pounds

--------------------------------------------------
```

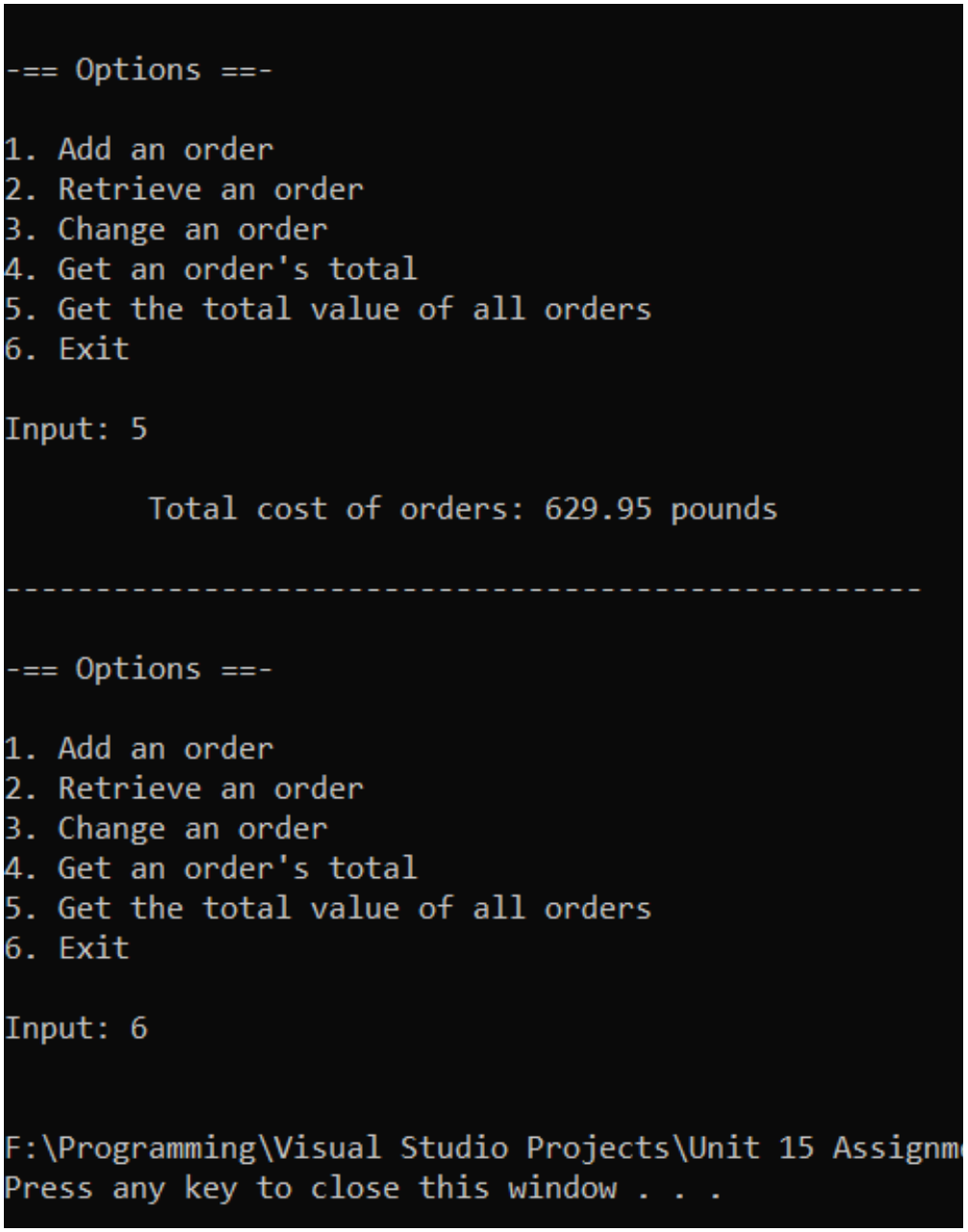
Retrieving data for two different orders

```
-== Options ==-

1. Add an order
2. Retrieve an order
3. Change an order
4. Get an order's total
5. Get the total value of all orders
6. Exit

Input: 2

Enter Order Number: 1

        Order Number: 1
        Item: wool_sweater
        Price: 99.99
        Quantity: 3

--------------------------------------------------

-== Options ==-

1. Add an order
2. Retrieve an order
3. Change an order
4. Get an order's total
5. Get the total value of all orders
6. Exit

Input: 2

Enter Order Number: 1

        Order Number: 1
        Item: wool_sweater
        Price: 99.99
        Quantity: 3

--------------------------------------------------
```

Changing data for a single order and getting the total price

```
-== Options ==-

1. Add an order
2. Retrieve an order
3. Change an order
4. Get an order's total
5. Get the total value of all orders
6. Exit

Input: 3

Enter Order Number: 1

1. Change Name
2. Change Price
3. Change quantity
Input: 3

Enter new quantity: 5

-----------------------------------------------------

-== Options ==-

1. Add an order
2. Retrieve an order
3. Change an order
4. Get an order's total
5. Get the total value of all orders
6. Exit

Input: 4

Enter Order Number: 1

        Total cost of order 1 is 499.95 pounds

-----------------------------------------------------
```

Retrieving the total cost of all orders and exiting



Source Code

```
#include <iostream>
using namespace std;

//Creating the class
class order_handler {
private:
	//Declaring array names and sizes
	string item_names[100] = {};
	float item_prices[100] = {};
	int item_quantity[100] = {};
	//Declaring counter used for order numbers
	int counter = 1;
public:
	//Declaring add_order method
	void add_order(string name, float price, int quantity)
	{
		item_names[counter] = name;
		item_prices[counter] = price;
		item_quantity[counter] = quantity;
		cout << endl << "	Order Number: " << counter;
		cout << endl << "	Total: " << price * quantity << " pounds" << endl;
		counter += 1;
	}
	//Declaring get_order method
	void get_order(int reference)
	{
		cout << "	Order Number: " << reference << endl;
		cout << "	Item: " << item_names[reference] << endl;
		cout << "	Price: " << item_prices[reference] << endl;
		cout << "	Quantity: " << item_quantity[reference] << endl;
	}
	//Declaring set_order method
	void set_order(int reference, int object) {
		if (object == 1) {
			cout << endl << "Enter new name: ";
			string temp1;
			cin >> temp1;
			item_names[reference] = temp1;
		}
		else if (object == 2) {
			cout << endl << "Enter new price: ";
			float temp1;
			cin >> temp1;
			item_prices[reference] = temp1;
		}
		else if (object == 3) {
			cout << endl << "Enter new quantity: ";
			int temp1;
			cin >> temp1;
			item_quantity[reference] = temp1;
		}
	}
	//Declaring get_total method
	void get_total(int reference) {
		double result;
		double price = item_prices[reference];
```

```
                double quantity = item_quantity[reference];
                result = price * quantity;
                cout << endl << "          Total cost of order " << reference << " is " << result << " pounds"
<< endl;
        }
        //Declaring get_total_order method
        void get_total_order() {
                double result = 0;
                for (int i = 0; i < counter; i++) {
                        double price = item_prices[i];
                        double quantity = item_quantity[i];
                        result += price * quantity;
                }
                cout << "          Total cost of orders: " << result << " pounds" << endl;
        }
};

//Main
int main()
{       //Declaring an instance of the order_handler class
        order_handler handler;
        string user_option;
        while (true) {
                //Menu
                cout << "-== Options ==-\n\n1. Add an order\n2. Retrieve an order\n3. Change an
order\n4. Get an order's total\n5. Get the total value of all orders\n6. Exit\n\nInput: ";
                //Get user input
                cin >> user_option;
                cout << endl;

                if (user_option == "1")
                {
                        string temp1;
                        float temp2;
                        int temp3;

                        cout << "Enter Item Name (Replace spaces with underscores): ";
                        cin >> temp1;

                        cout << "Enter Item Price: ";
                        cin >> temp2;

                        cout << "Enter Quantity: ";
                        cin >> temp3;

                        handler.add_order(temp1, temp2, temp3);
                }
                else if (user_option == "2")
                {
                        cout << "Enter Order Number: ";
                        int temp1;
                        cin >> temp1;
                        cout << endl;

                        handler.get_order(temp1);
                }
                else if (user_option == "3")
                {
```

```
                        cout << "Enter Order Number: ";
                        int reference;
                        cin >> reference;

                        cout << endl << "1. Change Name\n2. Change Price\n3. Change quantity\nInput:
";
                        int temp1;
                        cin >> temp1;

                        handler.set_order(reference, temp1);
                }
                else if (user_option == "4")
                {
                        cout << "Enter Order Number: ";
                        int reference;
                        cin >> reference;

                        handler.get_total(reference);
                }
                else if (user_option == "5")
                {
                        handler.get_total_order();
                }
                else
                {
                        exit(0);
                }
                cout << endl << "-----------------------------------------------------" << endl << endl;
        }
};
```

Gym Handler (P5)

Adding data for two new members and removing a member

```
-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Quit Program

Input: 1
Enter Name (No Spaces): OliverDodwell

   Name: OliverDodwell
   ID: 0

------------------------------

-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Quit Program

Input: 1
Enter Name (No Spaces): OlabodeAke

   Name: OlabodeAke
   ID: 1

------------------------------

-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Quit Program

Input: 2
Enter ID: 0

Member successfully removed

Updated IDs:
OlabodeAke : 0
------------------------------
```

Extending a member's subscription, cancelling it and then quitting the program

```
-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Quit Program

Input: 3
Enter ID: 0
How many months extension: 5
Old subscription length: 0 months
New subscription length: 5 months
------------------------------

-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Quit Program

Input: 4
Enter ID: 0
OlabodeAke's subscription length is now 0 months
------------------------------

-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Quit Program

Input: 5

F:\Programming\Visual Studio Projects\Unit 15 Assignment 2 Gym Handler\x64\Debug
Press any key to close this window . . .
```

Results of anomalus input (Test Plan)

Task	Input (Separated by commas)	Result	Issues(s)	Solution(s)
Add five members and remove two	1, MemberOne, 1, MemberTwo, 1, MemberThree, 1, MemberFour, 1, MemberFive, 2, 2, 2, 3	Updated IDs: MemberOne = 0 MemberTwo = 1 MemberFour = 2	N/A	N/A
Extend a membership by a negative value	1, MemberOne, 3, 0, -5	New Subscription Length: -5 months	Negative values are unsuitable for this data	Prevent the integer value going below 0
Cancel an unallocated ID	4, 0	s subscription length is now 0 months	Misleading but doesn't cause program to crash	Use IF() statements to check if there is a name allocated to the given ID
Delete an unallocated ID	2, 0	Updated IDs:	Misleading but doesn't cause program to crash	Use IF() statements to check if there is a name allocated to the given ID
Enter a name using spaces	1, Member One	Name: Member	C++ ignores user input after spaces	Implement a new way to get user input

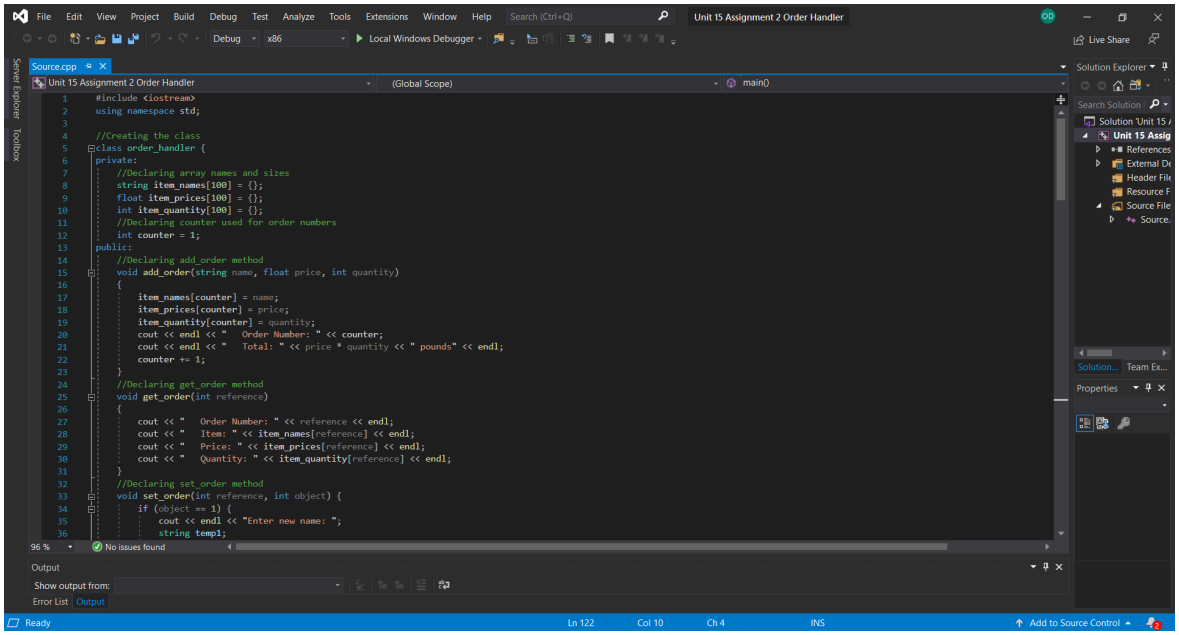
On Screen help (P6)

```
-----------------------------

-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Print all IDs
6. Help
7. Quit Program

Input: 1
Enter Name (No Spaces): Jack

CANNOT PERFORM THIS OPERATION: MEMBER LIMIT REACHED; FURTHER ATTEMPTS WILL OVERWRITE EXISTING MEMBERS

-----------------------------
```

```
-= Options =-

1. Create Member
2. Delete Member
3. Extend Membership
4. Cancel Membership
5. Print all IDs
6. Help
7. Quit Program

Input: 6

-= HELP =-
-If the program asks for a name, ensure there are no spaces as c++ cannot read them properly
-Ensure you are entering appropriate numbers when asked for the price or quantity of an order.

For other issues, contact administration

-----------------------------
```

Justification (M2)

IDE (Visual Studio):



Lists

Lists are a quick, easy to use way to store volatile data and reference it later in the program. In the program for Task 2 I use these to store data such as the order reference id, product names, price per unit and the quantity ordered. Lists were easier for me to learn on demand in C++ due to the similarity of how they work in other languages, whereas I have never touched on using a dynamic data storage like vectors before and had issues attempting to implement vectors.

Counter Variable

This variable's main purpose was to start at 0 and provide it's value as an order reference to the user, adding 1 to its total with each new order to ensure a unique value for each order, enabling the rest of the program to run smoothly.

Selection

My main use of selection within the program was to determine what function to call based on the user input, such as deciding to run the "get_order()" function when the user inputs the value "2". Given the nature of the program, I didn't use any selection outside of understanding what to do based on the user input as the commands were fairly straightforward and simply a case of telling the program to do a specific task as opposed to making decisions based on given data.

For Loop

I used a single for loop in the program for the purpose of adding up the value of each individual order, so that it can output the total value of all the orders combined to the

user. It was a quick and simple way to iterate through lists and control when it stopped iterating.

While Loop

The while loop is used to keep the entire program looping until the user specifically asks to exit the program in the user input stage of the program. This ensures the user doesn't have to restart after every command which would result in lost data, rendering the program useless. A simple "while (true) {}" statement was all that was needed in addition to an exit option in the menu that runs "exit(0)", ending the program with a code.

Analysis of test results (M3)

The test plan proved to be very useful in identifying the weaknesses of the program whilst showing how it could be improved to provide a much better user experience. The first test served the purpose of testing if a particular section of code worked and if not, identify exactly where the issue was. Fortunately, it worked just fine so I moved onto the next test.

The next test highlighted an issue with the function that extends a membership. You could provide input that would cause the subscription length to a negative value which could be a confusing response to the user's request. To solve this, I added an if statement to check if the resulting integer is below 0 and if so, set the value to 0 which worked.

For the third test, I tried cancelling the membership of an unassigned user. Due to how I structured the ID system, it only resulted in a logical error giving the user a misleading response. To fix it I simply added another if statement to check if the name referenced by the given ID has content and if it didn't it skipped it and provided a response accordingly.

My fourth test was effectively the same as the third, just with a different function. I attempted deleting a user that wasn't assigned yet and it did the same as the previous function, just gave a misleading response. I applied the same solution and it worked as expected.

My fifth and final test was for a bug present from the beginning to see what could be done about it. When assigning a name to a new member during runtime, it wouldn't save anything beyond a space in the user input. For example, if you typed "Oliver Dodwell" it would only save the "Oliver". My temporary solution to this was to include in the message asking for user input to use underscores instead of spaces.

Documentation (M4)

GymHandler functions

x_menu()	Runs the user interface and calls functions as requested. Add additional functions as options within this menu.
x_correction()	Used to correct the counter in the event that the array already contains registered users.
x_create(string Name)	Create a member, automatically assigns an ID.
x_delete(int ID)	Delete a member by ID.
x_extend(int ID, int n)	Extends a member's subscription length by ID.
x_cancel(int ID)	Cancels a member's subscription, setting it to 0.
x_print()	Prints out all the assigned IDs and corresponding details.
x_help()	Prints out help information.
x_quit()	Quits the program.
String Names[10]	Records the name values in a string array.
Int Months[10]	Records the subscription length in terms of months.
Int Counter[10]	Handles ID assignment, automatically updates to reflect the currently stored IDs.

Evaluate an object-oriented application (D2)

Minecraft

Minecraft was released in 2009 and continues to be one of the most played video games on a regular basis. It was developed in Java, an extremely popular object oriented programming language. I believe Minecraft is an exceptional example of object oriented programming for many reasons. The main concept of the game is an infinite world made of blocks where you can create or destroy, typically aimed at a younger audience but appeals to many regardless.

Blocks

Object oriented programming is exceptionally well at creating a template for a class and then modifying it for many purposes, blocks in minecraft are a perfect example of this in action. A block occupies a specific set of coordinates in a 3 dimensional space and each different block can have unique properties that use function names derived from the template but with unique code and such.

A visual example of this is textures. One block representing grass will obviously have a texture for grass whereas one for leaves will have the appropriate texture, despite being derived from the same class. Another example of differing properties is how on some blocks, the player's movement speed is altered to reflect the block (e.g you slip around lose grip on ice but are slowed when moving through cobwebs)

Entities

With object oriented programming, you can handle NPCs (non-player characters) relatively easy. You can create a template for traits shared among entities such as movement, looking, jumping, combat and so on from which each subclass can modify the function to work for the entity appropriately. This allows the developers to create any amount of different entities for the game using maintainable and reusable code, allowing more time to develop other areas of the game.

Modding

One of the main reasons for Minecraft's continued popularity is through unofficial modifications to the game, referred to as 'mods'. These are created to add content, mechanics and quality of life improvements to the game. The only reason there are so many available mods and tools to develop them is because of how easy it is to expand upon existing code in object oriented programming to seamlessly integrate with the main game.