

VNUHCM-UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

CSC10003 – OBJECT-ORIENTED PROGRAMMING

---

## OOP-Lab: Assignment 06

---

**Lecturer**

Mr. Nguyễn Lê Hoàng Dũng

Mr. Hồ Tuấn Thanh

**Class**

**23CLC08**

**Student**

23127226 – Nguyễn Đăng Minh



October 29th, 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>How the program works</b>	<b>3</b>
2.1	Product . . . . .	3
2.2	Node . . . . .	3
2.3	Linked List . . . . .	5
2.4	User . . . . .	6
2.5	Demonstration of the program . . . . .	7
2.5.1	Menu . . . . .	7
2.5.2	Add new product . . . . .	8
2.5.3	Save all orders to a file . . . . .	9
2.5.4	Select a product . . . . .	10
2.5.5	Increase quantity . . . . .	11
2.5.6	Apply promotion . . . . .	13
2.5.7	Delete a product . . . . .	14
2.5.8	Load products from file . . . . .	15

# 1 Introduction

This report explains how the console application program works similar to the shopping cart in Shopee. The image below is the shopee cart.

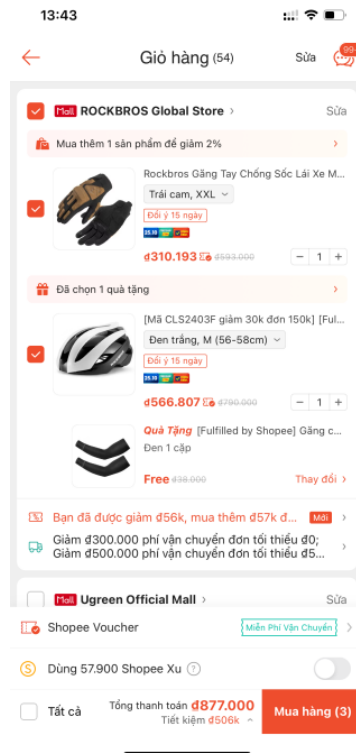


Figure 1: Shopee cart

My console application has some functions for user to interact the same way as the Shopee cart.

```
----- [USER OPTION] -----
1. *Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: █
```

Figure 2: Menu of the program

## 2 How the program works

The program consists of four classes: Product, Node, LinkedList, and User.

### 2.1 Product

The Product class will contain all information about the product such as its name, shop, type, etc. Below is the UML of the Product class.

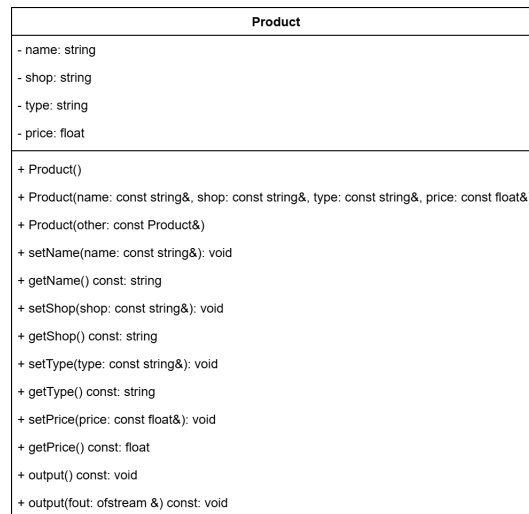


Figure 3: Product class diagram

The **output()** function is used to print the product's information.

### 2.2 Node

The Node class will contain a Product and a pointer to the next Node. Here I decide to use Linked-List structure to organize the product list.

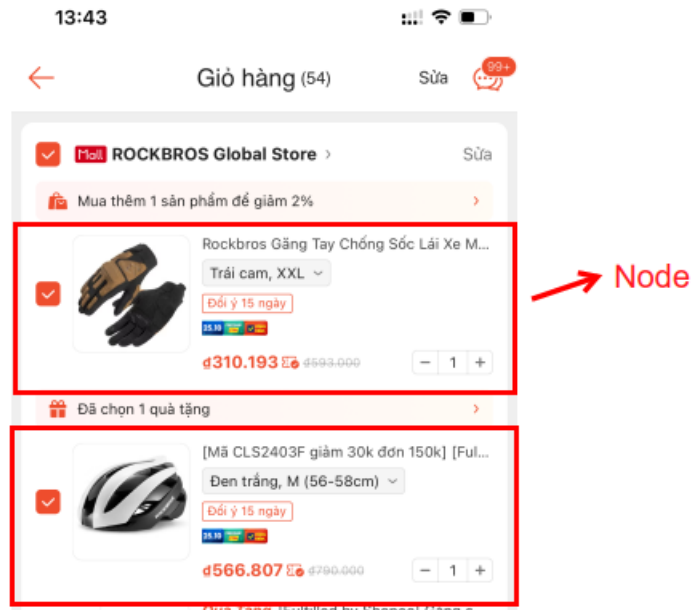


Figure 4: Linked List structure

Each block that contains a product, quantity, and selection state will be considered a Node in the Linked List. The Linked List will contain all the Node. The reason why I use Linked List is because it is easier and faster to delete an item. Since this is a small program so other operations such as traversal will not take so much time.

Node class will have the quantity, selection state, and the total price. Total price of a Node is the multiplication of product price and the quantity. Below is the UML class diagram of Node class.

Node
- product: Product - pNext: Node* - quantity: int - totalPrice: float - select: bool
+ Node() + Node(product: const Product&) + Node(product: const Product&, quantity: const int&) + Node() + setProduct(product: const Product&): void + getProduct() const: Product + setNext(pNext: Node*): void + getNext() const: Node* + setQuantity(quantity: const int&): void + getQuantity() const: int + adjustQuantity(change: const int&): void + getTotalPrice() const: float + setSelect(select: bool): void + getSelect() const: bool + output(id: const int&) const: void + output(id: const int &, fout: ostream &) const: void + applyPromotion(): void

Figure 5: Node class diagram

When user wants to update the quantity or selection state, the attributes in the Node class will change.

### 2.3 Linked List

The linked list will have a head and a tail of type Node\* (pointer to Node). Its attributes also have num (amount of current element), totalPrice (total price of all nodes). In order to get the sum prices of only-selected nodes, there is a method **getSelectedPrice()** which only takes account into selected products.

Each Node in Linked List will have a temporary id (their current order). When delete a Node, the id of all Nodes after the deleted one will decrease one. The id purpose is for user to easily choose the product to adjust the quantity or delete it.

The Linked List class also has a **adjustQuantity(id: int, change: int)** to adjust the quantity of a Node at a specific id.

Below is the UML class diagram of Linked List.

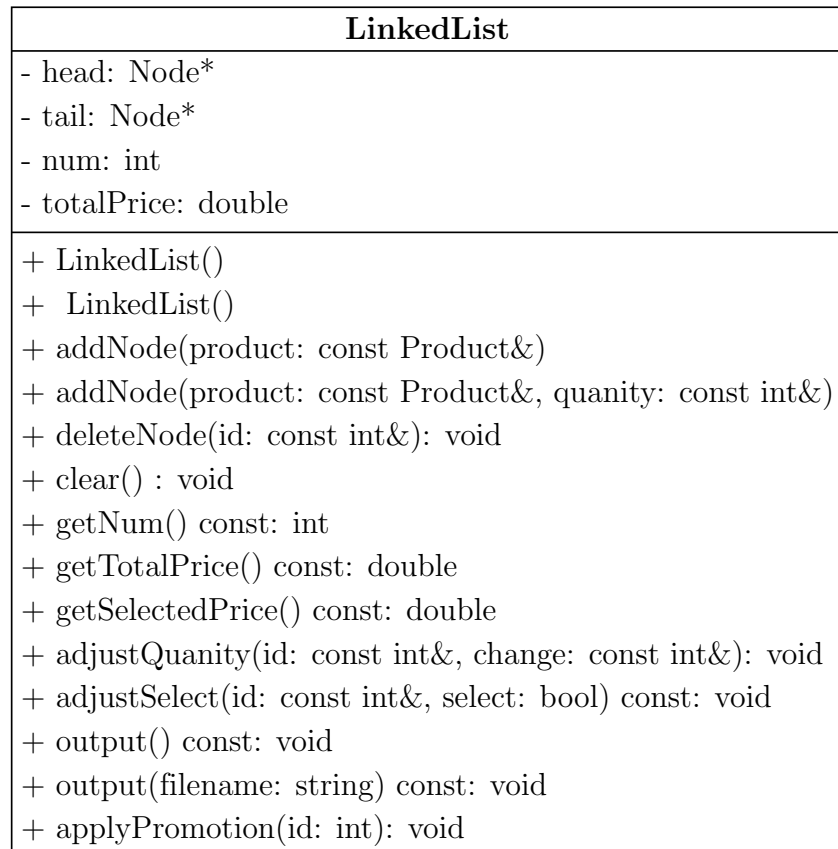


Figure 6: LinkedList class diagram

Both LinkedList and Node has destructor to delete dynamic allocated memory.

## 2.4 User

The User class symbolizes the user interface, it contains the linked list and its operations based on the linked list.

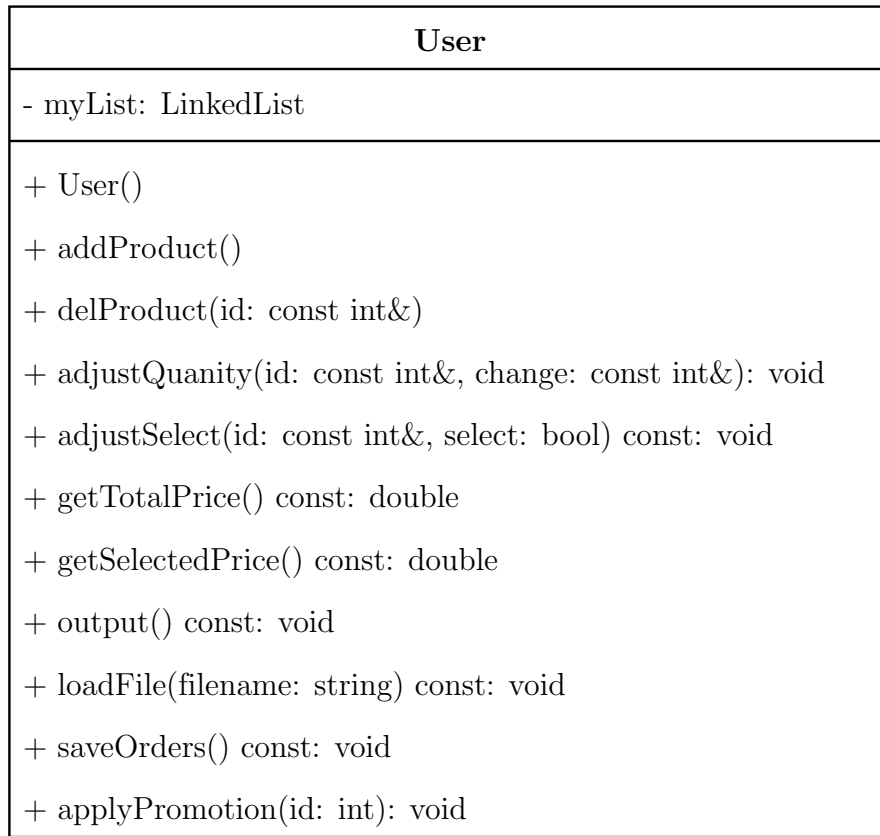


Figure 7: User class diagram

The User class has many functions that are as same as Linked List. It is for the user interaction, when user chooses to interact some function, the linked list will handle those. It is similar to separate abstraction and implementation, the user will not know the implementation below. They just need to know the interface (methods).

## 2.5 Demonstration of the program

The decrease-quantity, unselect function is as same as the reversed so I will not demonstrate them.

### 2.5.1 Menu

When open the program, you will receive a menu with thirteen options.



```
----- [USER OPTION] -----
1. *Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: █
```

Figure 8: Menu

### 2.5.2 Add new product

Here is an example of how you can add a product and then display it to the screen.

```
----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 2
Product name: a
Shop: a
Type: a
Price: 1
```

Figure 9: Add a product

Suppose I add two more products and then display it. The selection state of all products is initialize to be false (that is why price to pay equals zero). The quantity by default is also one.

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 0
2. Product{name: b; shop: b; type: b; price: 2} | quantity: 1 | total price: 2 | select: 0
3. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 0

```

Figure 10: Display all products and their state

### 2.5.3 Save all orders to a file

Type option 11 and all of your orders will be saved into a file named "order.txt".

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 11
Orders will be saved into orders.txt file!

```

Figure 11: save orders

File orders.txt:

```

Assignment04 > ≡ orders.txt
1  1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2  2. Product{name: b; shop: b; type: b; price: 2} | quantity: 2 | total price: 4 | select: 1
3  3. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
4

```

Figure 12: orders.txt

#### 2.5.4 Select a product

To select a product, you need to type option 6 and the id of the product.

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 6
Enter id of product to select: 1

```

Figure 13: Select product with id 1 (name = a)

Now if I redisplay, the selection state of the first product (id = 1) is changed. Besides, the price to pay is also equal to the first product's price because it is selected.

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2. Product{name: b; shop: b; type: b; price: 2} | quantity: 1 | total price: 2 | select: 0
3. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 1

```

Figure 14: Selection state changed

### 2.5.5 Increase quantity

Type option 4 and product's id to increase the quantity of that product. Of course, if you type an invalid id (does not match any product), nothing will change.

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 4
Enter id of product to increase quantity: 2

```

Figure 15: Increase quantity of product 2 (name = b)

Display to check:

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2. Product{name: b; shop: b; type: b; price: 2} | quantity: 2 | total price: 4 | select: 0
3. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 1

```

Figure 16: Quantity of product 2 is now two

Now if I select product 2, the price will be changed to 5.

```

----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2. Product{name: b; shop: b; type: b; price: 2} | quantity: 2 | total price: 4 | select: 1
3. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 5

```

Figure 17: Total price now includes product 2

Price to pay has been update to 5.

### 2.5.6 Apply promotion

Type option 12 and the product's id to apply promotion (50% discount) on that product. The price of that product will be only half after applying promotion.

```
----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 12
Enter id of a product to apply promotion: 2
```

Figure 18: apply promotion on product 2

Display all after applying promotion:

```
----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2. Product{name: b; shop: b; type: b; price: 2} | quantity: 2 | total price: 2 | select: 1
3. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 3
```

Figure 19: Display all products

### 2.5.7 Delete a product

Type option 3 and the product's id to delete that product. If you type an invalid id, nothing will change.

```
----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 3
Enter id of product to delete: 2
```

Figure 20: Delete second product

Display to check:

```
----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 1
```

Figure 21: No more product 2 in list

### 2.5.8 Load products from file

Type option 10 and the file's name to load all the product inside that file to the cart. Below is the example of file products.txt that contains information of new products.

```
Assignment04 > ≡ products.txt
1      d
2      d
3      d
4      4
5      2
6
7      e
8      e
9      e
10     5
11     3
```

Figure 22: products.txt

Display all after being loaded:

```
----- [USER OPTION] -----
1. Display all products
2. Add new product
3. Delete a product
4. Increase quantity of a product
5. Decrease quantity of a product
6. Select a product
7. Unselect a product
8. Display total prices of selected product
9. Display user options
10. Load products from file
11. Save orders in a file
12. Add 50% discount to a product
13. Exit
Enter option: 1
1. Product{name: a; shop: a; type: a; price: 1} | quantity: 1 | total price: 1 | select: 1
2. Product{name: c; shop: c; type: c; price: 3} | quantity: 1 | total price: 3 | select: 0
Price to pay: 1
```

Figure 23: Display all products