Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

SEMESTER PROJECT

OBJECT ORIENTED PROGRAMMING

- **SUBMITTED TO : DR SOBIA KHALID**
- **SUBMITTED BY : SYEDA FARWA BATOOL**
- **ROLL NO : 2022 - BSE - 071**

Skincare store management system

DESCRIPTION:

The Skincare Store Management System is a C++ application designed to streamline the management and processing of skincare products, facilitating a seamless experience for both customers and store owners. It encompasses a range of features aimed at efficient product management, customer interaction, and order processing.

FEATURES:

- **Product Management:**

The system initializes with a predefined list of skincare products, each with details including name, price, and quantity.
Although not directly accessible through user interaction, the system's architecture allows for the addition, updating, and removal of products by the store owner through code modifications.

- **Product Catalog:**

Customers can easily browse the available skincare products through a user-friendly menu interface.
The menu displays comprehensive information about each product, including its name, price, and current availability.

- **Cart Management:**

Customers can add products to their shopping cart, with the system ensuring real-time updates on product availability.
The cart management system allows for increasing or decreasing the quantity of items within the cart.

- **Order Processing:**

Upon completion of product selection, customers are prompted to provide their personal details, including name, address, and contact information.
The system calculates the total bill based on the selected products and displays it to the customer for confirmation.

- **File Export:**

After order confirmation, the system generates a detailed order summary, including customer information and the total bill.
This order summary is then exported to a file named "order.txt" for record-keeping and inventory management purposes.

IMPLEMENTATION:

The Skincare Store Management System leverages object-oriented programming principles to ensure modularity, scalability, and maintainability. Key components of the system include:

- **SkincareProduct Class:** Represents individual skincare products, encapsulating attributes such as name, price, and quantity.
- **CartItem Class:** Represents items within the customer's shopping cart, associating a skincare product with its quantity.
- **SkincareStore Class:** Manages the inventory of skincare products, providing essential functions for displaying the product menu, updating cart contents, and calculating the total bill.
- **Customer and Order Classes:** Facilitate order processing, with the Order class extending the Customer class to include order-specific functionality.

Inheritance and polymorphism are utilized to enhance code reusability and flexibility. The system's user interface is designed to provide clear prompts and feedback, ensuring a smooth and intuitive experience for both customers and store owners.

- **Inheritance:**

The Order class inherits from the Customer class. This inheritance relationship allows the Order class to access and utilize the properties and methods of the Customer class.

- **Polymorphic Behavior:**

Polymorphism is demonstrated through method overriding. Specifically, the getDetails() method is overridden in the Order class to provide order-specific details.
The getDetails() method in the Order class returns a string containing both the customer details and the total bill. This overridden method provides a polymorphic behavior, allowing

the Order class to present its specific details while retaining the functionality of the parent class method.

SUMMARY:

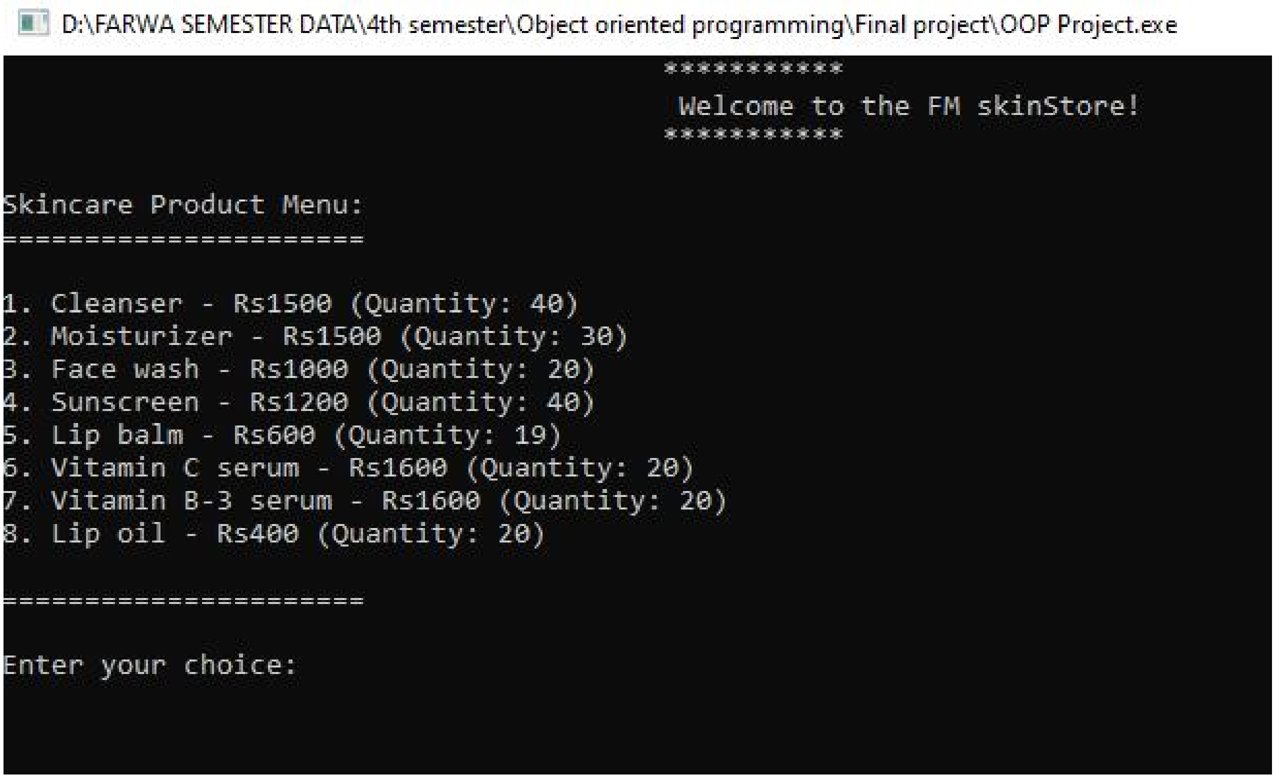
The Skincare Store Management System offers a robust solution for managing skincare products, catering to the needs of both customers and store owners. By combining efficient product management, seamless customer interactions, and streamlined order processing, the system facilitates enhanced productivity and customer satisfaction.

CODE

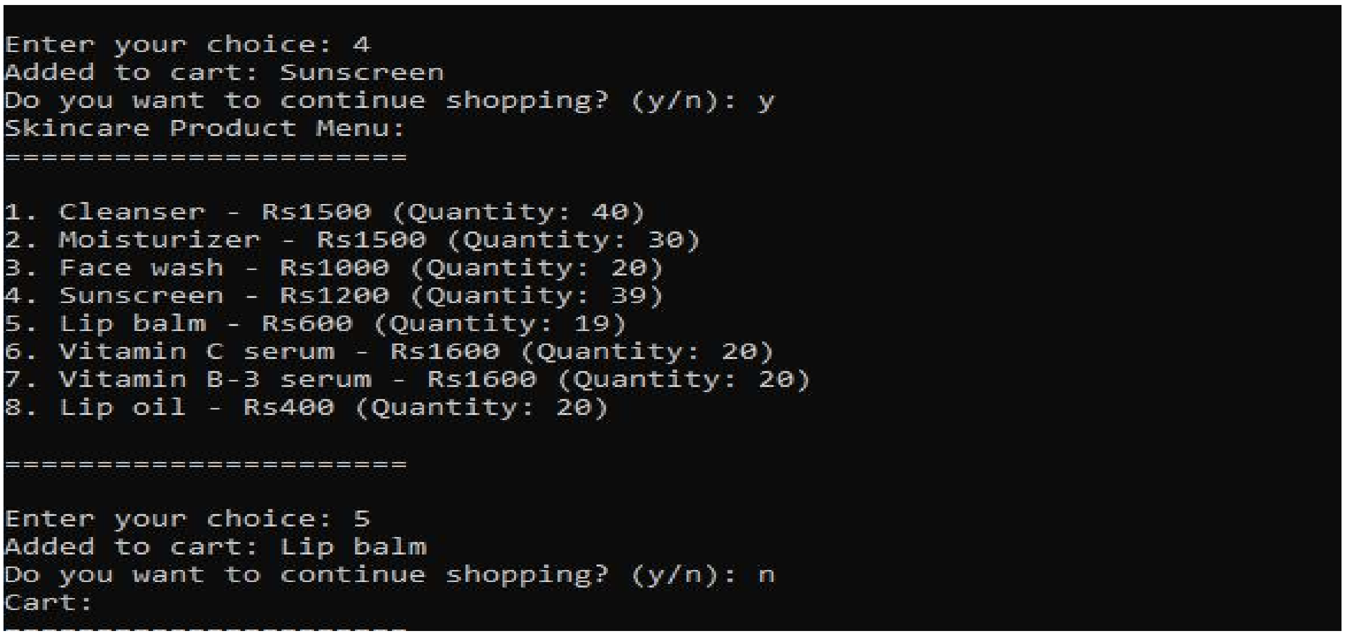
```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
using namespace std;

// Custom to_string function
template <typename T>
string to_string_custom(T value) {
    ostringstream oss;
    oss << value;
    return oss.str();
}

class SkincareProduct {
private:
    string name;
    double price;
    int quantity;

public:
    SkincareProduct() : name(""), price(0.0), quantity(0) {}

    SkincareProduct(const string& name, double price, int quantity)
        : name(name), price(price), quantity(quantity) {}

    string getName() const { return name; }
    double getPrice() const { return price; }
    int getQuantity() const { return quantity; }

    void decreaseQuantity() {
        if (quantity > 0) {
            quantity--;
        }
    }
};

class CartItem {
private:
    SkincareProduct product;
    int quantity;

public:
```

```cpp
    CartItem() : quantity(0) {}
    CartItem(const SkincareProduct& product, int quantity = 1)
        : product(product), quantity(quantity) {}

    SkincareProduct getProduct() const { return product; }
    int getQuantity() const { return quantity; }

    void increaseQuantity() { quantity++; }
};

class SkincareStore {
private:
    SkincareProduct products[10];
    CartItem cart[22];
    int cartSize;

public:
    SkincareStore() : cartSize(0) {
        // Add some sample products
        products[0] = SkincareProduct("Cleanser", 1500, 40);
        products[1] = SkincareProduct("Moisturizer", 1500, 30);
        products[2] = SkincareProduct("Face wash", 1000, 20);
        products[3] = SkincareProduct("Sunscreen", 1200, 40);
        products[4] = SkincareProduct("Lip balm", 600, 19);
        products[5] = SkincareProduct("Vitamin C serum", 1600, 20);
        products[6] = SkincareProduct("Vitamin B-3 serum", 1600, 20);
        products[7] = SkincareProduct("Lip oil", 400, 20);
    }

    void displayMenu() const {
        cout << "Skincare Product Menu:\n";
        cout << "======================\n";
        cout << endl;
        for (int i = 0; i < 8; ++i) {
            cout << i + 1 << ". " << products[i].getName() << " - Rs" << products[i].getPrice()
<< " (Quantity: " << products[i].getQuantity() << ")" << endl;
        }
        cout << endl;
        cout << "======================\n";
    }

    void addToCart(int choice) {
        if (choice > 0 && choice <= 8) {
            SkincareProduct selectedProduct = products[choice - 1];
            if (selectedProduct.getQuantity() > 0) {
                bool found = false;
                for (int i = 0; i < cartSize; ++i) {
                    if (cart[i].getProduct().getName() == selectedProduct.getName()) {
                        cart[i].increaseQuantity();
                        found = true;
                        break;
                    }
                }
                if (!found) {
                    cart[cartSize++] = CartItem(selectedProduct);
                }
                products[choice - 1].decreaseQuantity(); // update the quantity in the
products array
                cout << "Added to cart: " << selectedProduct.getName() << endl;
            }
            else {
                cout << "Sorry, the selected product is out of stock." << endl;
            }
        }
        else {
            cout << "Invalid choice. Please try again." << endl;
            displayMenu();
            int newChoice;
            cout << "Please enter again: ";
            cin >> newChoice;
            addToCart(newChoice);
        }
    }

    void displayCart() const {
        cout << "Cart:\n";
        cout << "======================\n";
        for (int i = 0; i < cartSize; ++i) {
            CartItem cartItem = cart[i];
            SkincareProduct product = cartItem.getProduct();
            cout << product.getName() << " - Rs" << product.getPrice() << " (Quantity: " <<
cartItem.getQuantity() << ")" << endl;
        }
        cout << "======================\n";
    }

    double calculateTotalBill() const {
        double total = 0.0;
        for (int i = 0; i < cartSize; ++i) {
            CartItem cartItem = cart[i];
            SkincareProduct product = cartItem.getProduct();
            total += product.getPrice() * cartItem.getQuantity();
        }
        return total;
    }
};

class Customer {
private:
    string name;
    string address;
    string contact;

public:
    Customer(const string& name, const string& address, const string& contact)
        : name(name), address(address), contact(contact) {}

    string getName() const { return name; }
    string getAddress() const { return address; }
    string getContact() const { return contact; }

    // Virtual function for polymorphism
    virtual string getDetails() const {
        return "Name: " + name + "\nAddress: " + address + "\nContact: " + contact;
    }
};

class Order : public Customer {
private:
    double totalBill;

public:
    Order(const Customer& customer, double totalBill)
        : Customer(customer), totalBill(totalBill) {}

    double getTotalBill() const { return totalBill; }

    // Override the virtual function
    string getDetails() const override {
        return Customer::getDetails() + "\nTotal Bill: Rs" + to_string_custom(totalBill);
    }

    void saveToFile() const {
        ofstream file;
        file.open("order.txt", ios::app);
        if (file.is_open()) {
            file << getDetails() << endl;
            file.close();
        }
        else {
            cout << "Unable to open file." << endl;
        }
    }

    void showAllOrders() const {
        ifstream file("order.txt");
        if (file.is_open()) {
            string line;
            while (getline(file, line)) {
                cout << line << endl;
            }
            file.close();
        }
        else {
            cout << "Unable to open file." << endl;
        }
    }
};

int main() {
    SkincareStore store;
    int choice;
    char continueShopping;
    cout << "\t" << "\t" << "\t" << "\t" << "\t" << "***********" << endl;
    cout << "\t" << "\t" << "\t" << "\t" << "\t" << " Welcome to the FM skinStore!\n";
    cout << "\t" << "\t" << "\t" << "\t" << "\t" << "***********" << endl;
    cout << endl;
    do {
        // system("cls"); // Use this line if you are on Windows to clear the screen
        store.displayMenu();
        cout << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        store.addToCart(choice);

        cout << "Do you want to continue shopping? (y/n): ";
        cin >> continueShopping;
    } while (continueShopping == 'y' || continueShopping == 'Y');

    store.displayCart();

    // Get customer information
    string name, address, contact;
    cout << "Enter your name: ";
    cin.ignore();
    getline(cin, name);
    cout << "Enter your address: ";
    getline(cin, address);
    cout << "Enter your contact number: ";
    getline(cin, contact);
    Customer customer(name, address, contact);
    double totalBill = store.calculateTotalBill();

    Order order(customer, totalBill);

    cout << "----------------------\n";
    cout << order.getDetails() << endl;  // Using polymorphism here
    order.saveToFile();

    // Show all orders
    cout << "\nAll Orders:\n";
    order.showAllOrders();

    return 0;
}
```

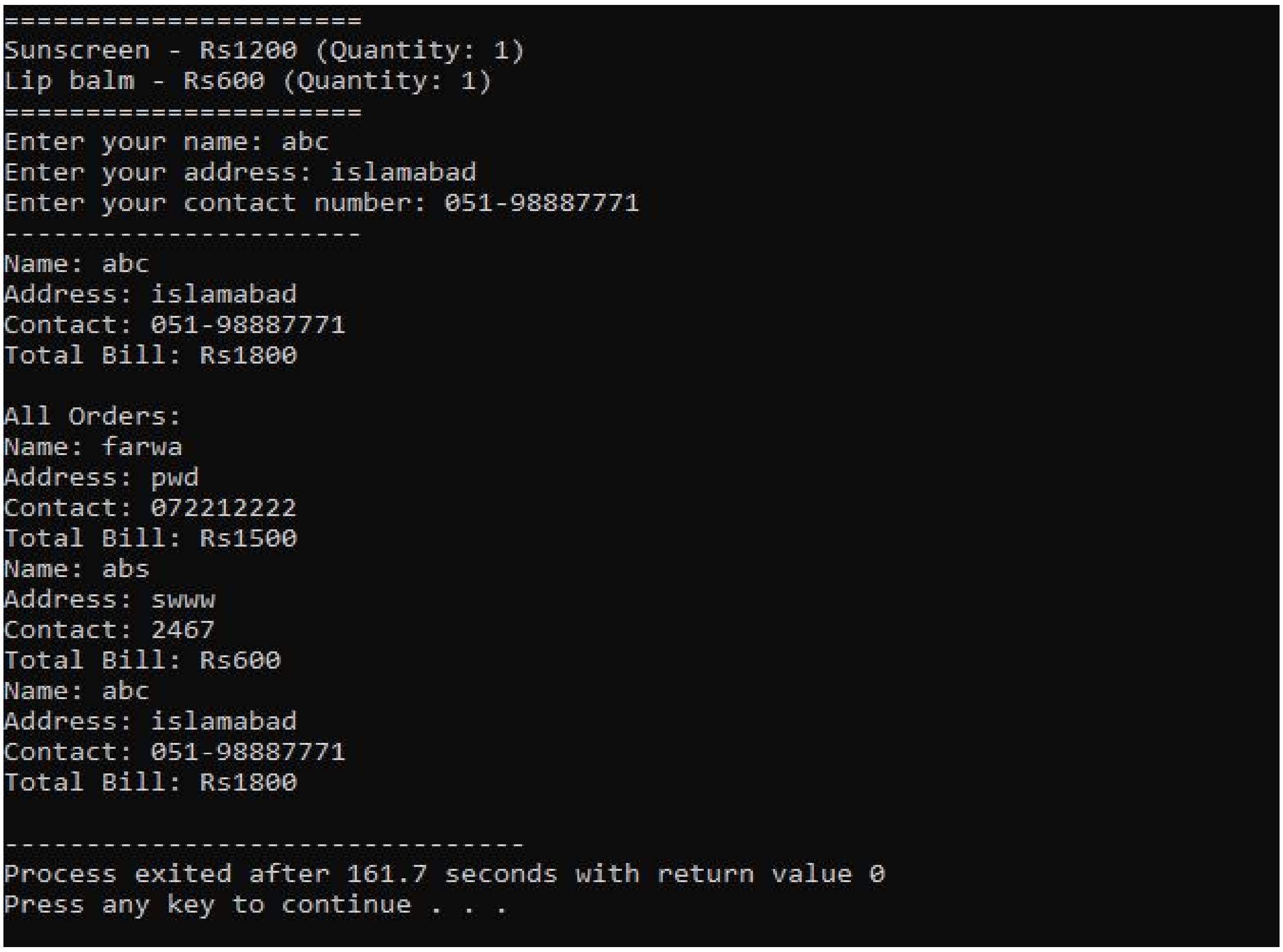
OUTPUT



```
D:\FARWA SEMESTER DATA\4th semester\Object oriented programming\Final project\OOP Project.exe
                                                  ************
                                                   Welcome to the FM skinStore!
                                                  ************


Skincare Product Menu:
======================

1. Cleanser - Rs1500 (Quantity: 40)
2. Moisturizer - Rs1500 (Quantity: 30)
3. Face wash - Rs1000 (Quantity: 20)
4. Sunscreen - Rs1200 (Quantity: 40)
5. Lip balm - Rs600 (Quantity: 19)
6. Vitamin C serum - Rs1600 (Quantity: 20)
7. Vitamin B-3 serum - Rs1600 (Quantity: 20)
8. Lip oil - Rs400 (Quantity: 20)

======================

Enter your choice:
```

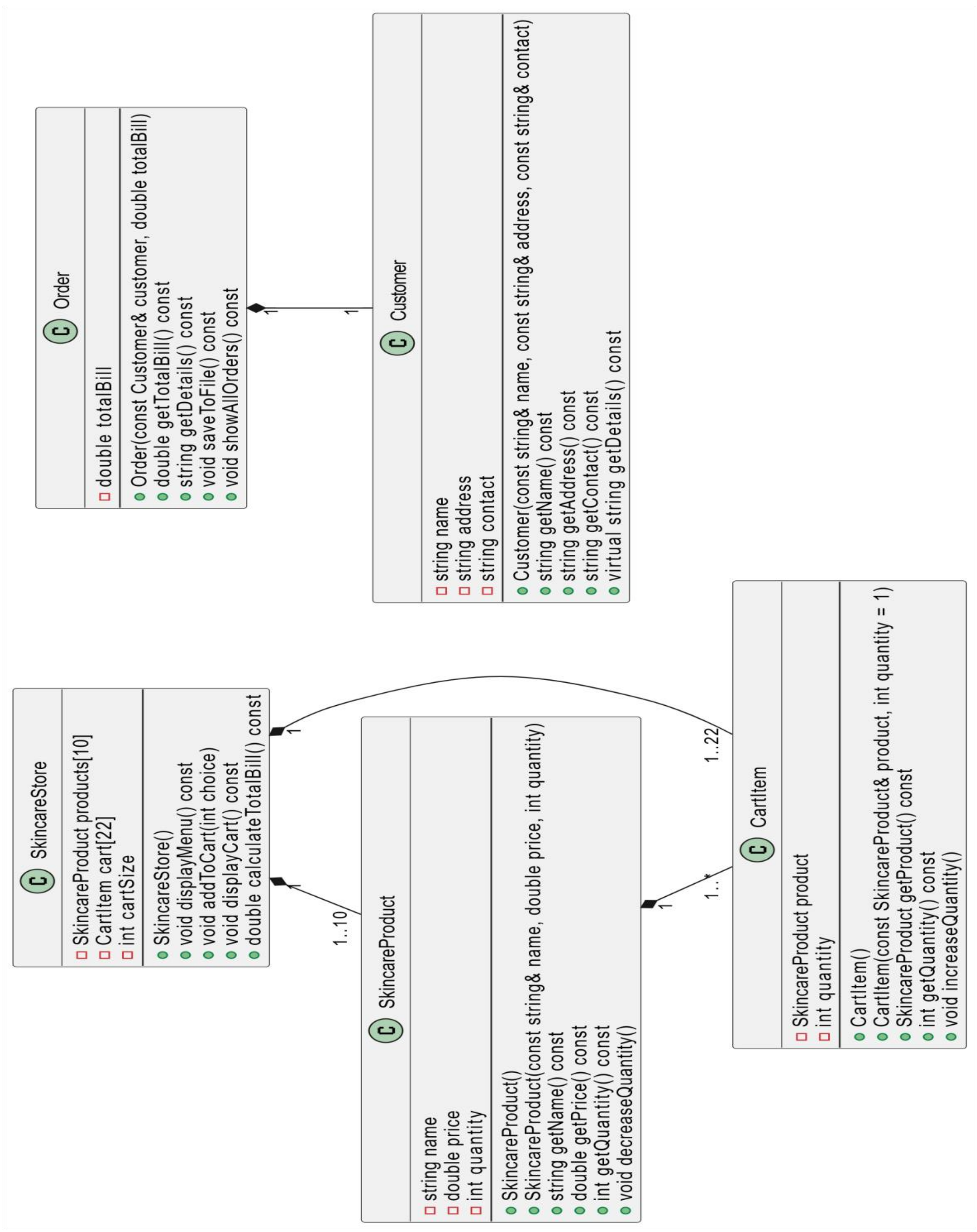


```
Enter your choice: 4
Added to cart: Sunscreen
Do you want to continue shopping? (y/n): y
Skincare Product Menu:
======================

1. Cleanser - Rs1500 (Quantity: 40)
2. Moisturizer - Rs1500 (Quantity: 30)
3. Face wash - Rs1000 (Quantity: 20)
4. Sunscreen - Rs1200 (Quantity: 39)
5. Lip balm - Rs600 (Quantity: 19)
6. Vitamin C serum - Rs1600 (Quantity: 20)
7. Vitamin B-3 serum - Rs1600 (Quantity: 20)
8. Lip oil - Rs400 (Quantity: 20)

======================

Enter your choice: 5
Added to cart: Lip balm
Do you want to continue shopping? (y/n): n
Cart:
```



```
======================
Sunscreen - Rs1200 (Quantity: 1)
Lip balm - Rs600 (Quantity: 1)
======================
Enter your name: abc
Enter your address: islamabad
Enter your contact number: 051-98887771
----------------------
Name: abc
Address: islamabad
Contact: 051-98887771
Total Bill: Rs1800

All Orders:
Name: farwa
Address: pwd
Contact: 072212222
Total Bill: Rs1500
Name: abs
Address: swww
Contact: 2467
Total Bill: Rs600
Name: abc
Address: islamabad
Contact: 051-98887771
Total Bill: Rs1800

--------------------------------
Process exited after 161.7 seconds with return value 0
Press any key to continue . . .
```

UML DIAGRAM



C Order
double totalBill
Order(const Customer& customer, double totalBill)
double getTotalBill() const
string getDetails() const
void saveToFile() const
void showAllOrders() const
1
1
C Customer
string name
string address
string contact
Customer(const string& name, const string& address, const string& contact)
string getName() const
string getAddress() const
string getContact() const
virtual string getDetails() const
C SkincareStore
SkincareProduct products[10]
CartItem cart[22]
int cartSize
SkincareStore()
void displayMenu() const
void addToCart(int choice)
void displayCart() const
double calculateTotalBill() const
1
1
1..10
1..22
C SkincareProduct
string name
double price
int quantity
SkincareProduct()
SkincareProduct(const string& name, double price, int quantity)
string getName() const
double getPrice() const
int getQuantity() const
void decreaseQuantity()
1
1..*
C CartItem
SkincareProduct product
int quantity
CartItem()
CartItem(const SkincareProduct& product, int quantity = 1)
SkincareProduct getProduct() const
int getQuantity() const
void increaseQuantity()