개요

제출

편집

코딩 결과

과제 052 객체를 이용한 Cart (과제 013 참고)

제출 마감일: 2023-04-07 23:59
업로드 가능한 파일 수: 3
제출 방식: 개인

목적

이 실습은 클래스의 의존성(dependency) 을 다루는 한 가지 방법을 연습을 합니다.

설명

소프트웨어를 설계하는 과정은 의존성과의 싸움이라도 해도 과언이 아닙니다.

과제를 진행하면서 문제를 풀기 위해 우리만의 함수를 구현해 봤습니다.

예를 들어, 사각형 도형을 생성하기 위해 create_rectangle 함수를 구현했습니다.

우리가 만든 프로그램은 이 함수에 의존적이죠.

C++ STL 은 훌륭한 알고리즘 함수들과 잘 디자인된 클래스를 제공하고 있습니다

std::vector, std::sort 등을 사용하는 프로그램은 역시 이들 함수나 클래스에 의존적이죠.

우리가 정의한 함수보다 STL 알고리즘 함수나 클래스가 덜 변할 것이라 기대되므로

의존성이 약간 더 낮다고 말 할 수 있습니다.

의존성이 생기면 의존하는 대상의 변경이 우리 프로그램의 변경에 영향을 끼칩니다.

예를 들어, create_rectangle 함수가 Shape 객체를 반환하거나,

std::vector 클래스가 push_back 을 지원하지 않는 등의 변경입니다.

개발자로서 우리는 이런 변화에 적절히 대비해 프로그램을 개발하도록 요구 받습니다.

객체에서는 보통 다른 객체에게 메시지를 보낼 때, 즉 다른 객체의 멤버 함수를 호출할 때 의존성이 생깁니다.

쇼핑 카트의 예를 들어 보겠습니다.

Cart 클래스는 구매하려는 제품을 담고 있습니다.

Checkout 클래스는 카트에 담긴 제품들의 구매 가격을 계산하고, 할인, 쿠폰, 포인트, 결제 방법들을 고려하여 최종 금액을 결제합니다.

Cart --> Checkout 이렇게 의존성이 생겼습니다.

문제

먼저, 과제 013 프로그램을 Cart, Item, Checkout 클래스를 이용하여 구현하세요.

다음으로, 이들 클래스를 사용하여 제공되는 프로그램이 동작하도록 하시오.

```
// main.cpp
int main() {
```

```
Checkout checkout;
Cart cart{&checkout};

while (true) {
    std::cout << "1. Add item" << std::endl;
    std::cout << "2. Delete item" << std::endl;
    std::cout << "3. View item details" << std::endl;
    std::cout << "4. View total cost" << std::endl;
    std::cout << "5. Quit" << std::endl;
    std::cout << "Enter your choice: ";

    int choice;
    std::cin >> choice;

    if (choice == 5) {
        break;
    }

    switch (choice) {
        case 1: {
            std::cout << "Enter item name: ";
            std::string name;
            std::cin >> name;

            std::cout << "Enter item quantity: ";
            int quantity;
            std::cin >> quantity;

            std::cout << "Enter item price: ";
            int price;
            std::cin >> price;

            Item item(name, quantity, price);
            cart.addItem(item);

            std::cout << "\tItem " << item.getId() << " added successfully." << std::endl;
            break;
        }
        case 2: {
            std::cout << "Enter item ID: ";
            int id;
            std::cin >> id;
            cart.deleteItem(id);
            std::cout << "\tItem deleted successfully." << std::endl;
            break;
        }
        case 3: {
            std::cout << "Enter the item number: ";
            int id;
            std::cin >> id;
            cart.viewItemDetails(id, std::cout);
            break;
        }
        case 4: {
            int totalPrice = cart.checkout();
```

```
            std::cout << "Total cost: " << totalPrice << std::endl;
            break;
        }
        default: {
            break;
        }
      }
   }
   return 0;
}
```

// Cart.h

```
using const_iterator = std::vector<Item>::const_iterator;

class Cart {
public:
   Cart(Checkout* checkout) : checkout_(checkout) {}

   void addItem(const Item& item) ;
   void deleteItem(int id) ;
   int checkout() ;
   void viewItemDetails(int id, std::ostream& out) const ;

   const_iterator cbegin() const ;
   const_iterator cend() const ;

private:
   Checkout* checkout_;
   std::vector<Item> items;
};
```

// Checkout.h

```
enum DiscountCode {
   NO_DISCOUNT,
   DISCOUNT_20,
   DISCOUNT_50
};

class Checkout {
public:
   Checkout(DiscountCode discountCode=NO_DISCOUNT) : discountCode(discountCode) {}

   int calculateTotalPrice(std::vector<Item>::const_iterator begin,
                           std::vector<Item>::const_iterator end) const;

   void setShippingAddress(const std::string &shippingAddress);
   void setDiscountCode(DiscountCode discountCode);
```

```
private:
    std::string shippingAddress;
    DiscountCode discountCode;
};
```

// Item.h

```
class Item {
public:
    Item(std::string name, int quantity, int price)
            : name(name), quantity(quantity), price(price) {
        static int IdCounter=0;
        id = IdCounter++;
    }

    int getId() const { return id; }
    std::string getName() const { return name; }
    int getQuantity() const { return quantity; }
    int getPrice() const { return price; }

private:
    int id;
    std::string name;
    int quantity;
    int price;
};
```

입력

```
1
Banana 2 1000
1
Apple 2 1500
3 0
3 1
4
2 0
3 0
3 1
4
5
```

출력

```
Enter item name: Enter item quantity: Enter item price:  Item 0 added successfully.
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
  Enter your choice: Enter item name: Enter item quantity: Enter item price:  Item 1 added successfully.
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
  Enter your choice: Enter the item number: Item 0:
Name: Banana
  Quantity: 2
Price: 1000
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
  Enter your choice: Enter the item number: Item 1:
Name: Apple
  Quantity: 2
Price: 1500
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
  Enter your choice:  Total cost: 5000
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
  Enter your choice: Enter item ID:  Item deleted successfully.
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
  Enter your choice: Enter the item number:  Item not found.
1. Add item
2. Delete item
3. View item details
4. View total cost
```

5. Quit
Enter your choice: Enter the item number: Item 1:
Name: Apple
Quantity: 2
Price: 1500
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
Enter your choice: Total cost: 3000
1. Add item
2. Delete item
3. View item details
4. View total cost
5. Quit
Enter your choice:

제출파일

Cart.cpp

Checkout.cpp

p52.csv

VPL