Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

**Лабораторна робота №2**

**Тема: «Успадкування, агрегація та композиція»**

|  |
| --- |
| Виконав: ст. гр. КI-23 |
| Карпець О. Р. |
| Перевірив викладач:  Козірова Н. Л. |
|  |

Кропивницький 2024

**Тема:** Успадкування, агрегація та композиція

**Мета:**ознайомитись з основними поняттями успадкування, агрегація та композиція в ООП та навчитись їх програмно реалізовувати мовою С++.

**Варіант 8**

**Завдання 1**

1. Створіть клас "Магазин", який містить такі властивості:
   * Назва магазину
   * Адреса магазину
   * Список товарів, які є у магазині (використовуйте агрегацію)
2. Створіть клас "Товар", який містить такі властивості:
   * Назва товару
   * Ціна товару
3. Створіть клас "Кошик", який містить такі властивості:
   * Список товарів, які знаходяться у кошику (використовуйте композицію)
   * Загальна вартість товарів у кошику
4. Реалізуйте методи для додавання та видалення товарів з кошика та розрахунку загальної вартості.
5. Напишіть демонстраційну функцію main(), в якій створюються об'єкти класів "Магазин", "Товар" та "Кошик". Додайте товари до списку товарів магазину, додайте товари до кошика та виведіть інформацію про них та загальну вартість у кошику.
6. Продемонструйте успадкування, створивши похідний клас від "Товар" (наприклад, "Продукт") з додатковими властивостями та методами.
7. Розширте функціональність, додавши додаткові методи та властивості до класів за власним бажанням. Не забудьте додати коментарі та пояснення до коду, щоб роз'яснити його функціональність та зрозумілість.

**Завдання 2**

Створіть використовуючи композицію два класи, які матимуть свої властивості, перший реалізуйте за допомогою включення (composition), а другий за допомогою посилання (reference), контейнером для цих класів використайте клас з лабораторної роботи №1, створіть також третій – похідний клас, використовуючи наслідування, базовим класом може бути або клас з лр№1, або класи які ви створили за допомогою композиції. За бажанням можете зобразити схематично як ваші класи залежать один від одного.

**Лістинг cart.cpp**

#include "cart.h"

Cart::Cart() {}

void Cart::addProductFrom(Product \*product)

{

this->products.push\_back(product);

}

void Cart::deleteProductFrom(Product& delProduct) {

string delName = delProduct.getProductName();

for (auto it = products.begin(); it != products.end(); ++it) {

if ((\*it)->getProductName() == delName) {

delete \*it;

products.erase(it);

return;

}

}

}

float Cart::getAllCosts()

{

float allCost = 0;

for (auto& product : this->products) {

allCost += product->getProductCost();

}

return allCost;

}

vector<Product\*> Cart::getAllProductFrom()

{

return this->products;

}

**Лістинг cart.h**

#ifndef CART\_H

#define CART\_H

#include <string>

#include <vector>

#include "product.h"

using namespace std;

class Cart

{

public:

Cart();

void addProductFrom(Product \*product);

void deleteProductFrom(Product& delProduct);

float getAllCosts();

vector<Product\*> getAllProductFrom();

private:

vector<Product\*> products;

float productsCost;

};

#endif // CART\_H

**Лістинг FoodPorduct.cpp**

#include "FoodProduct.h"

FoodProduct::FoodProduct() {}

void FoodProduct::setExpiryDate(string expiryDate)

{

this->expiryDate = expiryDate;

}

string FoodProduct::getExpiryDate()

{

return this->expiryDate;

}

**Лістинг FoodProduct.h**

#ifndef FOODPRODUCT\_H

#define FOODPRODUCT\_H

#include <string>

#include "product.h"

using namespace std;

class FoodProduct : public Product

{

public:

FoodProduct();

void setExpiryDate(string);

string getExpiryDate();

private:

string expiryDate;

};

#endif // FOODPRODUCT\_H

**Лістинг product.cpp**

#include "product.h"

Product::Product() {}

void Product::setProductName(string productName)

{

this->productName = productName;

}

void Product::setProductCost(float productCost)

{

this->productCost = productCost;

}

string Product::getProductName()

{

return this->productName;

}

float Product::getProductCost()

{

return this->productCost;

}

**Лістинг product.h**

#ifndef PRODUCT\_H

#define PRODUCT\_H

#include <string>

using namespace std;

class Product

{

public:

Product();

void setProductName(string);

void setProductCost(float);

string getProductName();

float getProductCost();

protected:

string productName;

float productCost;

};

#endif // PRODUCT\_H

**Лістинг shop.cpp**

#include "shop.h"

#include "product.h"

Shop::Shop() {}

void Shop::setShopName(string shopName)

{

this->shopName = shopName;

}

void Shop::setShopAddress(string shopAddress)

{

this->shopAddress = shopAddress;

}

void Shop::setShopProducts(vector<Product\*> products)

{

this->products = products;

}

string Shop::getShopName()

{

return this->shopName;

}

string Shop::getShopAddress()

{

return this->shopAddress;

}

vector<Product\*> Shop::getShopProducts()

{

return this->products;

}

**Лістинг shop.h**

#ifndef SHOP\_H

#define SHOP\_H

#include <string>

#include <vector>

#include "product.h"

using namespace std;

class Shop

{

public:

Shop();

void setShopName(string);

void setShopAddress(string);

void setShopProducts(vector<Product\*> products);

string getShopName();

string getShopAddress();

vector<Product\*> getShopProducts();

private:

string shopName;

string shopAddress;

vector<Product\*> products;

};

#endif // SHOP\_H

**Лістинг Prj/main.cpp**

#include <QCoreApplication>

#include <iostream>

#include "FoodProduct.h"

#include "cart.h"

#include "shop.h"

using namespace std;

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Cart \*cart = new Cart();

Shop \*shop = new Shop();

for (int index = 0; index < 100; index++) {

FoodProduct \*product = new FoodProduct();

product->setProductCost(index);

string productName = ("Apple " + to\_string(index));

product->setProductName(productName);

string expiryDate = ("Expiry" + to\_string(index));

product->setExpiryDate(expiryDate);

cart->addProductFrom(product);

}

shop->setShopProducts(cart->getAllProductFrom());

cout << "All cart products info:" << endl;

for (const auto& product : shop->getShopProducts()) {

cout << "Product Name: " << product->getProductName() << endl;

cout << "Product Cost: " << product->getProductCost() << endl;

}

cout << "Product Cost: " << cart->getAllCosts() << endl;

return a.exec();

}

**Лістинг movie.cpp**

#include "movie.h"

Movie::Movie() {}

bool Movie::setTitle(string newTitle)

{

if (checkStrings(newTitle))

{

this->title = newTitle;

return true;

}

return false;

}

bool Movie::setDirector(string newDirector)

{

if (checkStrings(newDirector))

{

this->director = newDirector;

return true;

}

return false;

}

bool Movie::setYear(int newYear)

{

if (checkInts(newYear))

{

this->year = newYear;

return true;

}

return false;

}

bool Movie::setDuration(int newDuration)

{

if (checkInts(newDuration))

{

this->duration = newDuration;

return true;

}

return false;

}

bool Movie::checkStrings(string checkString)

{

if (checkString.length() > 0)

{

return true;

}

return false;

}

bool Movie::checkInts(int checkInt)

{

if (checkInt > 0)

{

return true;

}

return false;

}

**Лістинг movie.h**

#ifndef MOVIE\_H

#define MOVIE\_H

#include <string>

using namespace std;

class Movie

{

public:

Movie();

bool setTitle(string newTitle);

string getTitle() { return title; }

bool setDirector(string newDirector);

string getDirector() { return director; }

bool setYear(int newYear);

int getYear() { return year; }

bool setDuration(int newDuration);

int getDuration() { return duration; }

private:

string title;

string director;

int year;

int duration;

bool checkStrings(string checkString);

bool checkInts(int checkInt);

};

#endif // MOVIE\_H

**Лістинг movieadvanceddetails.cpp**

#include "movieadvanceddetails.h"

#include "moviedetails.h"

#include <iostream>

using namespace std;

MovieAdvancedDetails::MovieAdvancedDetails() : MovieDetails() {}

void MovieAdvancedDetails::printMovieInfo() {

cout << "Title: " << getMovieTitle() << std::endl;

cout << "Director: " << getMovieDirector() << std::endl;

cout << "Year: " << getMovieYear() << std::endl;

cout << "Duration: " << getMovieDuration() << " minutes" << std::endl;

}

**Лістинг movieadvanceddetails.h**

#ifndef MOVIEADVANCEDDETAILS\_H

#define MOVIEADVANCEDDETAILS\_H

#include "moviedetails.h"

class MovieAdvancedDetails : public MovieDetails

{

public:

MovieAdvancedDetails();

void printMovieInfo();

};

#endif // MOVIEADVANCEDDETAILS\_H

**Лістинг moviedetails.cpp**

#include "moviedetails.h"

MovieDetails::MovieDetails() {}

bool MovieDetails::setMovieTitle(string newTitle) {

return movie.setTitle(newTitle);

}

string MovieDetails::getMovieTitle() {

return movie.getTitle();

}

bool MovieDetails::setMovieDirector(string newDirector) {

return movie.setDirector(newDirector);

}

string MovieDetails::getMovieDirector() {

return movie.getDirector();

}

bool MovieDetails::setMovieYear(int newYear) {

return movie.setYear(newYear);

}

int MovieDetails::getMovieYear() {

return movie.getYear();

}

bool MovieDetails::setMovieDuration(int newDuration) {

return movie.setDuration(newDuration);

}

int MovieDetails::getMovieDuration() {

return movie.getDuration();

}

**Лістинг moviedetails.h**

#ifndef MOVIE\_DETAILS\_H

#define MOVIE\_DETAILS\_H

#include "movie.h"

class MovieDetails {

public:

MovieDetails();

bool setMovieTitle(string newTitle);

string getMovieTitle();

bool setMovieDirector(string newDirector);

string getMovieDirector();

bool setMovieYear(int newYear);

int getMovieYear();

bool setMovieDuration(int newDuration);

int getMovieDuration();

private:

Movie movie;

};

#endif // MOVIE\_DETAILS\_H

**Лістинг movieref.cpp**

#include "movieref.h"

MovieRef::MovieRef(Movie\* moviePtr) : movie(moviePtr) {}

bool MovieRef::setMovieTitle(string newTitle) {

return movie->setTitle(newTitle);

}

string MovieRef::getMovieTitle() {

return movie->getTitle();

}

bool MovieRef::setMovieDirector(string newDirector) {

return movie->setDirector(newDirector);

}

string MovieRef::getMovieDirector() {

return movie->getDirector();

}

bool MovieRef::setMovieYear(int newYear) {

return movie->setYear(newYear);

}

int MovieRef::getMovieYear() {

return movie->getYear();

}

bool MovieRef::setMovieDuration(int newDuration) {

return movie->setDuration(newDuration);

}

int MovieRef::getMovieDuration() {

return movie->getDuration();

}

**Лістинг movieref.h**

#ifndef MOVIEREF\_H

#define MOVIEREF\_H

#include "movie.h"

class MovieRef {

public:

MovieRef(Movie\* moviePtr);

bool setMovieTitle(string newTitle);

string getMovieTitle();

bool setMovieDirector(string newDirector);

string getMovieDirector();

bool setMovieYear(int newYear);

int getMovieYear();

bool setMovieDuration(int newDuration);

int getMovieDuration();

private:

Movie\* movie; // Посилання на Movie

};

#endif // MOVIEREF\_H

**Висновок:**

Під час виконання цієї лабораторної роботи ми ознайомилися з базовими поняттями успадкування, агрегації та композиції в об'єктно-орієнтованому програмуванні. Реалізовані класи продемонстрували, як за допомогою композиції можна створювати складні об'єкти, які містять інші об'єкти як свої властивості, що дозволяє краще структурувати дані та логіку програми. Водночас агрегація показала, як один клас може включати інший за допомогою посилання, забезпечуючи більшу гнучкість у взаємодії між об'єктами.

Через успадкування ми змогли розширити функціональність базового класу, додаючи нові властивості та методи до похідних класів, що дозволяє забезпечити повторне використання коду та підвищити ефективність розробки. Такий підхід підвищує читабельність та підтримуваність коду, що є однією з головних цілей ООП.