Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №4

з дисципліни «Об’єктно-орієнтоване програмування»

**Тема:** Поліморфізм. Обробка винятків

Виконав: студент гр. КН-23

**Чубенко.І.К**

Перевірив:Асистент  
**Козірова Н.Л.**

Кропивницький 2024

Тема: Поліморфізм. Обробка винятків  
Мета: Ознайомитись з поняттям поліморфізму у мові C++ та навчитись використовувати віртуальні функції для досягнення поліморфізму. Також вивчити принципи обробки винятків у мові C++.

Завдання 1

Примітка

Для виконання цього завдання вам потрібно середовище розробки Qt

Creator, якщо Ви раніше не працювали з ним то встановіть, та ознайомтесь.

Наступним кроком створіть новий проект Qt: Використовуйте Qt Creator

для створення нового проекту типу "Qt Widgets Application". При створені

проекту ви отримаєте всі файли, які потрібні і навіть більше, отже вітаю,

з цим «більше» вам і прийдеться працювати, а якщо точніше то до цього

завдання вам треба зробити інтерфейс. Тему для завдання можна обрати на

вибір

1. Напишіть гру «Піймай муху», на формі знаходиться муха, яка має

тікати від курсора, також на формі має знаходитися пастка, коли муха

попадеться в пастку гра закінчується. В реалізації програми має бути

похідний клас, що унаслідується від базового класу «QMainWindow»,

в похідному класі, перевизначить функцію «event»

Лістинг mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QEvent>

#include <QLabel>

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

protected:

bool event(QEvent \*event) override;

private:

QLabel \*fly;

QLabel \*trap;

void moveFly();

bool checkCollision();

};

#endif

Лістинг main.cpp

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Лістинг mainwindow.cpp

#include "mainwindow.h"

#include <QMessageBox>

#include <QMouseEvent>

#include <QRandomGenerator>

#include <QtMath>

#include <QPixmap>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), fly(new QLabel(this)), trap(new QLabel(this))

{

QPixmap flyImage("E:/Second\_course/ООП/testqt/muha/fly.png");

fly->setPixmap(flyImage.scaled(30, 30, Qt::KeepAspectRatio));

fly->setGeometry(50, 50, 30, 30);

QPixmap trapImage("E:/Second\_course/ООП/testqt/muha/sik.png");

trap->setPixmap(trapImage.scaled(30, 30, Qt::KeepAspectRatio));

trap->setGeometry(150, 150, 100, 100);

}

MainWindow::~MainWindow() {}

bool MainWindow::checkCollision()

{

return fly->geometry().intersects(trap->geometry());

}

bool MainWindow::event(QEvent \*event)

{

if (event->type() == QEvent::MouseMove) {

QMouseEvent \*mouseEvent = static\_cast<QMouseEvent\*>(event);

int cursorX = static\_cast<int>(mouseEvent->position().x());

int cursorY = static\_cast<int>(mouseEvent->position().y());

int flyX = fly->x();

int flyY = fly->y();

int dx = cursorX - flyX;

int dy = cursorY - flyY;

double distance = qSqrt(dx \* dx + dy \* dy);

if (distance < 50) {

int escapeDistance = 20;

int newX = flyX - static\_cast<int>(escapeDistance \* dx / distance);

int newY = flyY - static\_cast<int>(escapeDistance \* dy / distance);

if (newX < 0) newX = escapeDistance;

if (newY < 0) newY = escapeDistance;

if (newX + fly->width() > width()) newX = width() - fly->width() - escapeDistance;

if (newY + fly->height() > height()) newY = height() - fly->height() - escapeDistance;

fly->move(newX, newY);

}

if (checkCollision()) {

QMessageBox::information(this, "Перемогаа!", "Ви піймали муху!");

}

}

return QMainWindow::event(event);

}

Завдання 2

Варіант 6

1. Розробіть систему керування магазином, яка включатиме обробку різних

типів товарів. Кожен тип товару має свої характеристики, які потрібно

реалізувати за допомогою наслідування, сетерів та гетерів.

Кожен товар має наступні характеристики:

• Назва

• Ціна

• Кількість на складі

Кожен тип товару має власні додаткові характеристики:

Меблі:

• Матеріал

• Колір

Електроніка:

• Бренд

• Гарантійний термін

2. Створіть базовий абстрактний клас Product з віртуальними функціями та

використати поліморфізм для реалізації додаткових методів та функцій.

Також, додайте виняткові ситуації для обробки некоректних даних.

3. Створіть похідні класи Furniture та Electronics, які успадковуються від

класу Product. Реалізуйте в них відповідні віртуальні функції та додайте

додаткові характеристики, які були зазначені вище.

4. У вашій програмі мають бути використані виняткові ситуації для

обробки некоректних даних, наприклад, якщо некоректна ціна товару або

від'ємна кількість на складі.

Ви можете розширити його, додати додаткові методи та функціональні

можливості, які вам здаються відповідними.

#include <iostream>

#include <string>

#include <Windows.h>

#include <stdexcept>

class Product {

protected:

std::string name;

double price;

int stock;

public:

Product(const std::string& name, double price, int stock)

: name(name) {

setPrice(price);

setStock(stock);

}

std::string getName() const { return name; }

void setName(const std::string& name) { this->name = name; }

double getPrice() const { return price; }

virtual void setPrice(double price) {

if (price < 0)

throw std::invalid\_argument("Ціна не може бути від'ємною.");

this->price = price;

}

int getStock() const { return stock; }

virtual void setStock(int stock) {

if (stock < 0)

throw std::invalid\_argument("Кількість на складі не може бути від'ємною.");

this->stock = stock;

}

virtual void display() const = 0;

};

class Furniture : public Product {

private:

std::string material;

std::string color;

public:

Furniture(const std::string& name, double price, int stock, const std::string& material, const std::string& color)

: Product(name, price, stock), material(material), color(color) {}

std::string getMaterial() const { return material; }

void setMaterial(const std::string& material) { this->material = material; }

std::string getColor() const { return color; }

void setColor(const std::string& color) { this->color = color; }

void display() const override {

std::cout << "Меблі: " << name << "\nЦіна: " << price << " грн\nКількість: " << stock

<< "\nМатеріал: " << material << "\nКолір: " << color << "\n";

}

};

class Electronics : public Product {

private:

std::string brand;

int warrantyPeriod;

public:

Electronics(const std::string& name, double price, int stock, const std::string& brand, int warrantyPeriod)

: Product(name, price, stock), brand(brand) {

setWarrantyPeriod(warrantyPeriod);

}

std::string getBrand() const { return brand; }

void setBrand(const std::string& brand) { this->brand = brand; }

int getWarrantyPeriod() const { return warrantyPeriod; }

void setWarrantyPeriod(int warrantyPeriod) {

if (warrantyPeriod < 0)

throw std::invalid\_argument("Гарантійний термін не може бути від'ємним.");

this->warrantyPeriod = warrantyPeriod;

}

void display() const override {

std::cout << "Електроніка: " << name << "\nЦіна: " << price << " грн\nКількість: " << stock

<< "\nБренд: " << brand << "\nГарантійний термін: " << warrantyPeriod << " місяців\n";

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try {

Furniture chair("Стілець", 1200.0, 10, "Дерево", "Коричневий");

Electronics phone("Смартфон", 15000.0, 5, "Samsung", 24);

chair.display();

std::cout << "-----------------------\n";

phone.display();

}

catch (const std::invalid\_argument& e) {

std::cerr << "Помилка: " << e.what() << "\n";

}

return 0;

}

Висновок