Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

**Лабораторна робота №4**

**Тема: «Поліморфізм. Обробка винятків»**

|  |
| --- |
| Виконав: ст. гр. КI-23 |
| Карпець О. Р. |
| Перевірив викладач:  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++.

**Варіант 8**

**Завдання 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»

**Завдання 2**

Розробіть систему керування одягом, яка включатиме різні типи одягу. Кожен тип одягу має свої характеристики, які потрібно реалізувати за допомогою наслідування, сетерів та гетерів.

Кожен одяг має наступні характеристики:

• Назва

• Розмір

• Колір

Кожен тип одягу має власні додаткові характеристики:

Футболка:

• Матеріал

• Тип коміра

Штани:

• Матеріал

• Тип застібки

Створіть базовий абстрактний клас Clothing з віртуальними функціями та використати поліморфізм для реалізації додаткових методів та функцій. Також, додайте виняткові ситуації для обробки некоректних даних.

Створіть похідні класи TShirt та Pants, які успадковуються від класу Clothing. Реалізуйте в них відповідні віртуальні функції та додайте додаткові характеристики, які були зазначені вище.

У вашій програмі мають бути використані виняткові ситуації для обробки некоректних даних, наприклад, якщо некоректний розмір одягу або порожнє ім'я. Ви можете розширити його, додати додаткові методи та функціональні можливості, які вам здаються відповідними.

**Завдання 1**

**Лістинг arrestcockroach.cpp**

#include "arrestcockroach.h"

ArrestCockroach::ArrestCockroach(QWidget \*parent)

: QMainWindow(parent), drawing(false) {

setFixedSize(800, 600);

cockroachPosition = QPoint(QRandomGenerator::global()->bounded(width() - 30), QRandomGenerator::global()->bounded(height() - 30));

timer = new QTimer(this);

connect(timer, &QTimer::timeout, this, &ArrestCockroach::moveCockroach);

timer->start(1000);

}

ArrestCockroach::~ArrestCockroach() {}

void ArrestCockroach::paintEvent(QPaintEvent \*event) {

QPainter painter(this);

painter.setBrush(Qt::black);

painter.drawRect(cockroachPosition.x(), cockroachPosition.y(), 30, 30);

painter.setPen(Qt::red);

if (drawnPoints.size() > 1) {

painter.drawPolyline(drawnPoints.data(), drawnPoints.size());

}

if (drawnPoints.size() > 2) {

painter.drawLine(drawnPoints.last(), drawnPoints.first());

}

}

void ArrestCockroach::mousePressEvent(QMouseEvent \*event) {

if (event->button() == Qt::LeftButton) {

drawnPoints.clear();

drawnPoints.append(event->pos());

drawing = true;

}

}

void ArrestCockroach::mouseMoveEvent(QMouseEvent \*event) {

if (drawing) {

drawnPoints.append(event->pos());

update();

}

}

void ArrestCockroach::mouseReleaseEvent(QMouseEvent \*event) {

if (event->button() == Qt::LeftButton) {

drawing = false;

checkIfCaught();

update();

}

}

void ArrestCockroach::moveCockroach() {

cockroachPosition = QPoint(QRandomGenerator::global()->bounded(width() - 30), QRandomGenerator::global()->bounded(height() - 30));

update();

}

void ArrestCockroach::checkIfCaught() {

QPoint cockroachCenter(cockroachPosition.x() + 15, cockroachPosition.y() + 15);

if (drawnPoints.size() > 2) {

QPolygon polygon;

for (const QPoint &point : drawnPoints) {

polygon << point;

}

if (polygon.containsPoint(cockroachCenter, Qt::OddEvenFill)) {

QMessageBox::information(this, "WIN", "You cought cockroach!");

drawnPoints.clear();

}

}

}

**Лістинг arrestcockroach.h**

#ifndef ARRESTCOCKROACH\_H

#define ARRESTCOCKROACH\_H

#include <QMainWindow>

#include <QPainter>

#include <QMouseEvent>

#include <QTimer>

#include <QMessageBox>

#include <QRandomGenerator>

#include <QVector>

class ArrestCockroach : public QMainWindow {

Q\_OBJECT

public:

ArrestCockroach(QWidget \*parent = nullptr);

~ArrestCockroach();

protected:

void paintEvent(QPaintEvent \*event) override;

void mousePressEvent(QMouseEvent \*event) override;

void mouseMoveEvent(QMouseEvent \*event) override;

void mouseReleaseEvent(QMouseEvent \*event) override;

private:

void moveCockroach();

void checkIfCaught();

QPoint cockroachPosition;

QVector<QPoint> drawnPoints;

bool drawing;

QTimer \*timer;

};

#endif // ARRESTCOCKROACH\_H

**Лістинг mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

**Лістинг mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**Лістинг main.cpp**

#include "arrestcockroach.h"

#include <QApplication>

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

QApplication app(argc, argv);

ArrestCockroach w;

w.show();

return app.exec();

}

**Завдання 2**

**Лістинг clothing.cpp**

#include "clothing.h"

Clothing::Clothing(QString name, QString size, QString color) {

setName(name);

setSize(size);

setColor(color);

}

void Clothing::setName(QString name) {

if (name.isEmpty()) {

throw invalid\_argument("Name cant be empty");

}

this->name = name;

}

QString Clothing::getName() {

return name;

}

void Clothing::setSize(QString size) {

if (size.isEmpty()) {

throw invalid\_argument("Size cant be empty");

}

this->size = size;

}

QString Clothing::getSize() {

return size;

}

void Clothing::setColor(QString color) {

if (color.isEmpty()) {

throw invalid\_argument("Color cant be empty");

}

this->color = color;

}

QString Clothing::getColor() {

return color;

}

void Clothing::printInfo()

{

}

**Лістинг clothing.h**

#ifndef CLOTHING\_H

#define CLOTHING\_H

#include <QString>

#include <iostream>

#include <stdexcept>

using namespace std;

class Clothing {

protected:

QString name;

QString size;

QString color;

public:

Clothing(QString name, QString size, QString color);

virtual ~Clothing() = default;

virtual void setName(QString name);

virtual QString getName();

virtual void setSize(QString size);

virtual QString getSize();

virtual void setColor(QString color);

virtual QString getColor();

virtual void printInfo();

};

#endif // CLOTHING\_H

**Лістинг pants.cpp**

#include "pants.h"

Pants::Pants(QString name, QString size, QString color, QString material, QString fasteningType)

: Clothing(name, size, color), material(material), fasteningType(fasteningType) {}

void Pants::setMaterial(QString material) {

if (material.isEmpty()) {

throw invalid\_argument("Material cant be empty");

}

this->material = material;

}

QString Pants::getMaterial() {

return material;

}

void Pants::setFasteningType(QString fasteningType) {

if (fasteningType.isEmpty()) {

throw invalid\_argument("Fastening type cant be empty");

}

this->fasteningType = fasteningType;

}

QString Pants::getFasteningType() {

return fasteningType;

}

void Pants::printInfo() {

cout << "Pants: " << name.toStdString()

<< ", Size: " << size.toStdString()

<< ", Color: " << color.toStdString()

<< ", Material: " << material.toStdString()

<< ", Fastening Type: " << fasteningType.toStdString() << endl;

}

**Лістинг pants.h**

#ifndef PANTS\_H

#define PANTS\_H

#include "clothing.h"

class Pants : public Clothing {

private:

QString material;

QString fasteningType;

public:

Pants(QString name, QString size, QString color, QString material, QString fasteningType);

void setMaterial(QString material);

QString getMaterial();

void setFasteningType(QString fasteningType);

QString getFasteningType();

void printInfo() override;

};

#endif // PANTS\_H

**Лістинг tshirt.cpp**

#include "tshirt.h"

TShirt::TShirt(QString name, QString size, QString color, QString material, QString collarType)

: Clothing(name, size, color), material(material), collarType(collarType) {}

void TShirt::setMaterial(QString material) {

if (material.isEmpty()) {

throw invalid\_argument("Material cant be empty");

}

this->material = material;

}

QString TShirt::getMaterial() {

return material;

}

void TShirt::setCollarType(QString collarType) {

if (collarType.isEmpty()) {

throw invalid\_argument("Collar type cant be empty");

}

this->collarType = collarType;

}

QString TShirt::getCollarType() {

return collarType;

}

void TShirt::printInfo() {

cout << "T-Shirt: " << name.toStdString()

<< ", Size: " << size.toStdString()

<< ", Color: " << color.toStdString()

<< ", Material: " << material.toStdString()

<< ", Collar Type: " << collarType.toStdString() << endl;

}

**Лістинг tshirt.h**

#ifndef TSHIRT\_H

#define TSHIRT\_H

#include "clothing.h"

class TShirt : public Clothing {

private:

QString material;

QString collarType;

public:

TShirt(QString name, QString size, QString color, QString material, QString collarType);

void setMaterial(QString material);

QString getMaterial();

void setCollarType(QString collarType);

QString getCollarType();

void printInfo() override;

};

#endif // TSHIRT\_H

**Лістинг main.cpp**

#include <QCoreApplication>

#include "tshirt.h"

#include "pants.h"

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

QCoreApplication a(argc, argv);

TShirt tshirt("Nike T-Shirt", "M", "Red", "Cotton", "Round Neck");

Pants pants("Levi's Jeans", "L", "Blue", "Denim", "Zipper");

tshirt.printInfo();

pants.printInfo();

return a.exec();

}

**Висновок**

У процесі виконання роботи вивчив основи поліморфізму та обробки винятків у C++. Мною було реалізовано дві програми:

1. Гра "Піймай муху", в якій використав класи Qt. Ця програма продемонструвала, як перевизначати функції та обробляти події у віконному інтерфейсі.
2. Система керування одягом, де створив абстрактний клас і використав наслідування для керування різними типами одягу. Для перевірки коректності вхідних даних застосував механізми обробки винятків.

Ця робота допомогла мені закріпити знання про ключові аспекти об'єктно-орієнтованого програмування та їх практичне застосування в C++.