МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 9
На тему: "*Принцип поліморфізму*"
З дисципліни: "Об'єктно-орієнтоване програмування"

Лектор:
доцент кафедри ПЗ
Коротєєва Т.О.

Виконав:
студент групи ПЗ-16
Коваленко Д.М.

Прийняв:
доцент кафедри ПЗ
Яцишин С.І.

«_____» _____ 2022 р.
$\sum$ = _____.....

Львів — 2022

Тема. Принцип поліморфізму.

Мета. Навчитись створювати списки об'єктів базового типу, що включають об'єкти похідних типів. Освоїти способи вирішення проблеми неоднозначності при множинному наслідуванні. Вивчити плюси заміщення функцій при множинному наслідуванні. Навчитись використовувати чисті віртуальні функції, знати коли варто використовувати абстрактні класи.

Лабораторне завдання

Розробити ієрархію класів відповідно до варіанту. Використати множинне наслідування, продемонструвати вирішення проблеми з неоднозначністю доступу до членів базових класів за допомогою віртуального наслідування, за допомогою явного звертання до членів класу та за допомогою заміщення функцій в похідному класі (при потребі). Створити списки об'єктів базового типу, в них помістити об'єкти похідного типу. Продемонструвати виклик функцій з об'єктів – елементів списку. Використати оператор $dynamic_cast$ (при потребі). Створити абстрактний клас, використати чисто віртуальну функцію, що містить реалізацію в базовому класі. Для вивільнення динамічної пам'яті використовувати віртуальні деструктори. Сформувати звіт до лабораторної роботи. Відобразити в ньому діаграму наслідування класів.

Індивідуальне завдання

9. КлієнтГуртівні Базовий клас – WholeSaleClient. Далі – RegularWSClient, VIPWSClient. Базовий клас зберігає загальні дані про клієнтів. VIP до прикладу, дозволяє, швидше отримувати нотифікації, різні варіанти накопичувальних знижок тощо.

Теоретичні відомості

Поліморфізм – одна з трьох основних парадигм ООП. Якщо говорити коротко, поліморфізм – це здатність об'єкта використовувати методи похідного класу, який не існує на момент створення базового.

Уявімо ситуацію, що ми створюємо програму, в якій ми працюємо з класами тварин. Один з цих класів Bird(птахи), а другий Mammal(ссавці). Клас Bird містить функцію Fly(), а клас Mammal містить функцію Gallop() – біг галопом.

А тепер сталось так, що нам потрібно створити новий міфічний персонаж – крилатого Пегаса (Pegasus), який буде гібридом між птахом та ссавцем

Код програми

Назва файлу: *main.cpp*

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Назва файлу: *mainwindow.cpp*

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include "QMessageBox"

#include "list.h"
#include "wholesaleclient.h"
```

```
#include "wsregularclient.h"
#include "wsvipclient.h"

WholeSaleClient *clients[2];

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow) {
    ui->setupUi(this);
    ui->tab->setEnabled(false);
    ui->tab_2->setEnabled(false);
}

MainWindow::~MainWindow() {
    delete ui;
}

void MainWindow::on_startButton_clicked() {
    QString name = ui->startNameEdit->text();
    double money = ui->startMoneyEdit->text().toDouble();
    if (ui->typeComboBox->currentIndex() == 0) {
        clients[0] = new WSRegularClient(name, money);
        ui->currentMoneyEdit->setText(QString::number(money));
        ui->tab->setEnabled(true);
    } else {
        clients[1] = new WSVIPClient(name, money);
        ui->currentMoneyEdit_2->setText(QString::number(money));
        ui->tab_2->setEnabled(true);
    }
}

void MainWindow::on_currentBuyButton_clicked() {
    QString item = ui->currentItemEdit->text();
    double price = ui->currentPriceEdit->text().toDouble();
    clients[0]->buy(item, price);
    ui->currentMoneyEdit->setText(QString::number(dynamic_cast<WSRegularClient
    *>(clients[0])->getMoney()));
}

void MainWindow::on_currentBuyButton_2_clicked() {
    QString item = ui->currentItemEdit_2->text();
    double price = ui->currentPriceEdit_2->text().toDouble();
    clients[1]->buy(item, price);
    ui->currentMoneyEdit_2->setText(QString::number(dynamic_cast<WSVIPClient *>(
    clients[1])->getMoney()));
}

void MainWindow::on_checkButton_clicked() {
    QMessageBox msgBox;
    List * check = clients[0]->getCheck();
    QString checkStr = "Name: " + dynamic_cast<WSRegularClient *>(clients[0])->
    getName() + "\n";
    for (int i = 0; i <= check->getLastIndex(); i++) {
        checkStr += check->at(i) + "\n";
    }
    msgBox.setText("Your regular check:");
    msgBox.setDetailedText(checkStr);
    msgBox.exec();
}

void MainWindow::on_checkButton_2_clicked() {
```

```
    QMessageBox msgBox;
    List * check = clients[1]->getCheck();
    QString checkStr = "Name: " + dynamic_cast<WSVIPClient *>(clients[1])->
   getName() + "\n";
    for (int i = 0; i <= check->getLastIndex(); i++) {
        checkStr += check->at(i) + "\n";
    }
    msgBox.setText("Your regular check:");
    msgBox.setDetailedText(checkStr);
    msgBox.exec();
}
```

Назва файлу: *human.h*

```
    #ifndef HUMAN_H
#define HUMAN_H

#include "QString"

class Human {
protected:
    QString name;
public:
    virtual ~Human() = 0;
};

#endif // HUMAN_H
```

Назва файлу: *human.cpp*

```
#include "human.h"

Human::~Human() {}
```

Назва файлу: *client.h*

```
    #ifndef CLIENT_H
#define CLIENT_H

#include "human.h"

class Client : public virtual Human {
protected:
    double money;
public:
    virtual ~Client() = 0;
    void withdrawMoney(double amount) { this->money -= amount; };
    double getMoney() { return this->money; };
};

#endif // CLIENT_H
```

Назва файлу: *client.cpp*

```
    #include "client.h"

Client::~Client() {}
```

Назва файлу: *wholesaleclient.h*

```
#ifndef WHOLESALECLIENT_H
#define WHOLESALECLIENT_H
#define SIZE 1000
```

```
#include "list.h"
#include "human.h"

class WholeSaleClient : public virtual Human {
    protected:
    List * cart;
    List * check;
    public:
    WholeSaleClient():
    cart(new List()),
    check(new List())
    {};
    virtual ~WholeSaleClient();
    virtual void buy(QString item, double price) = 0;
    void addToCart(QString item);
    void addToCheck(QString item, double price);
    List * getCheck() { return this->check; };
};

#endif // WHOLESALECLIENT_H
```

Назва файлу: *wholesaleclient.cpp*

```
#include "wholesaleclient.h"

WholeSaleClient::~WholeSaleClient() {}

void WholeSaleClient::addToCart(QString item) {
    this->cart->append(item);
}

void WholeSaleClient::addToCheck(QString item, double price) {
    this->check->append(item + " " + QString::number(price));
}
```

Назва файлу: *wsregularclient.h*

```
    #ifndef WSREGULARCLIENT_H
#define WSREGULARCLIENT_H

#include "client.h"
#include "wholesaleclient.h"

class WSRegularClient : public Client, public WholeSaleClient {
public:
    WSRegularClient(QString name, double money) {
        this->name = name;
        this->money = money;
    };
    virtual ~WSRegularClient();
    void buy(QString item, double price);
};

#endif // WSREGULARCLIENT_H
```

Назва файлу: *wsregularclient.cpp*

```
#include "wholesaleclient.h"

WholeSaleClient::WholeSaleClient() {}
WholeSaleClient::~WholeSaleClient() {}
```

Назва файлу: *wsvipclient.h*

```
#ifndef WSVIPCLIENT_H
#define WSVIPCLIENT_H

#include "client.h"
#include "wholesaleclient.h"

class WSVIPClient : public Client, public WholeSaleClient {
public:
    WSVIPClient(QString name, double money) {
        this->name = name;
        this->money = money;
    };
    virtual ~WSVIPClient();
    void buy(QString item, double price);
};

#endif // WSVIPCLIENT_H
```

Назва файлу: *wsvipclient.cpp*

```
#include "wsvipclient.h"

WSVIPClient::~WSVIPClient() {}

void WSVIPClient::buy(QString item, double price) {
    if (this->money < price) return;
    price *= 0.95;
    this->addToCart(item);
    this->addToCheck(item, price);
    this->withdrawMoney(price);
}
```

Назва файлу: *list.h*

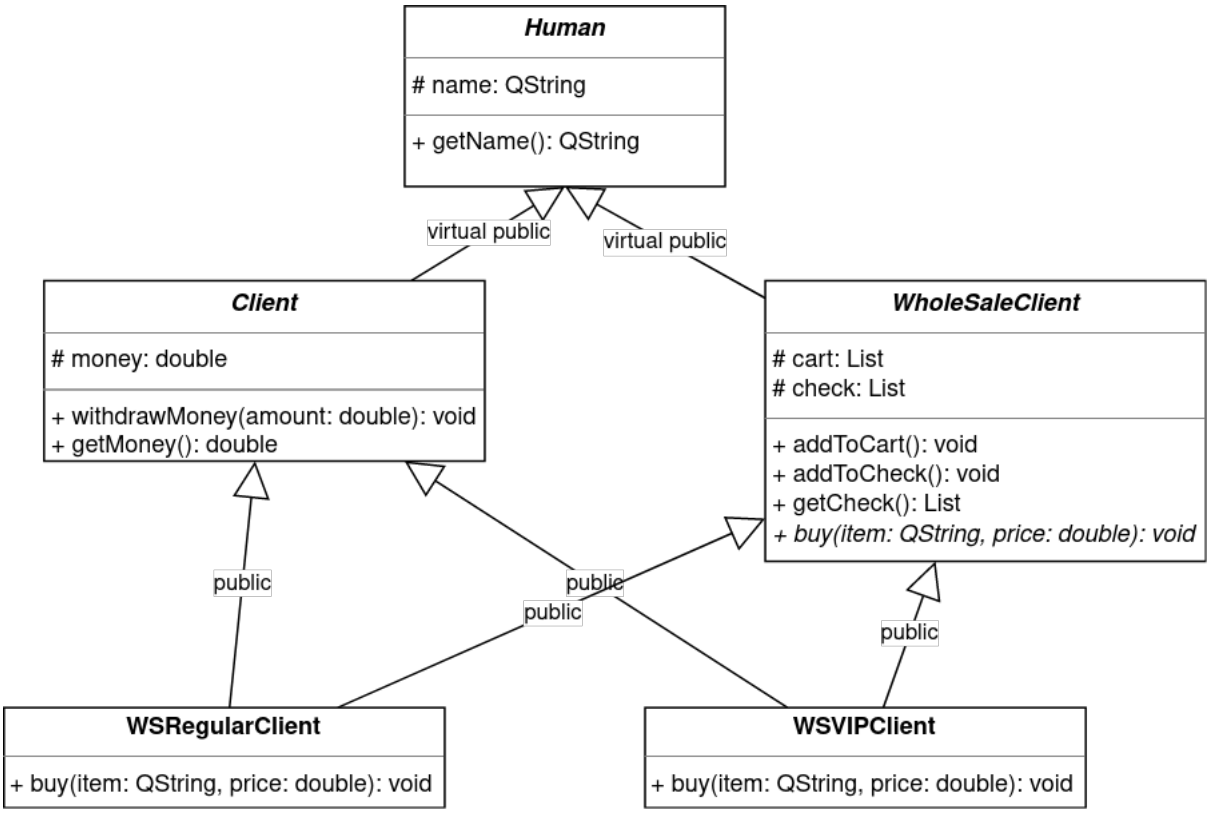
```
#ifndef LIST_H
#define LIST_H

#include "QString"
#define SIZE 1000

class List {
    private:
    QString * data;
    int lastIndex;
    public:
    List() {
        data = new QString[SIZE];
        lastIndex = -1;
    };
    void append(QString data);
    QString * getList() { return this->data; };
    int getLastIndex() { return this->lastIndex; };
    QString at(int i) { return data[i]; };
};

#endif // LIST_H
```

Назва файлу: *list.cpp*

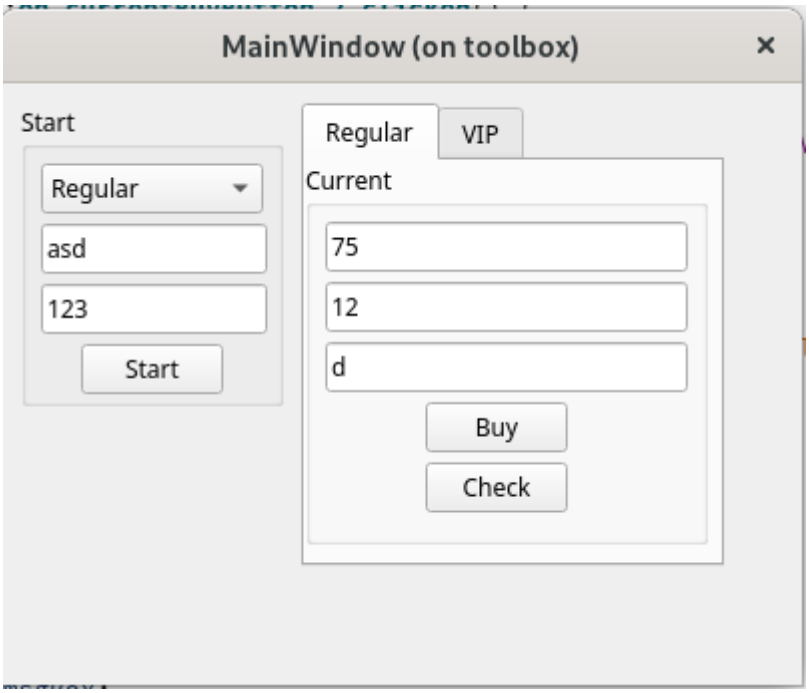
```
#include "list.h"

void List::append(QString data) {
```

```
    this->lastIndex++;
    this->data[this->lastIndex] = data;
}
```

UML-діаграма



Робота програми



Висновок

Під час виконання лабораторної роботи я навчився створювати списки об'єктів базового типу, що включають об'єкти похідних типів. Освоїв способи вирішення проблеми неоднозначності при множинному наслідуванні. Вивчив плюси заміщення функцій при множинному наслідуванні. Навчився використовувати чисті віртуальні функції, знаю коли варто використовувати абстрактні класи.