Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення

**Звіт**

Про виконання лабораторної роботи №8

**На тему:**

«Наслідування. Створення та використання ієрархії класів»

з дисципліни

«Об’єктно-орієнтоване програмування»

**Лектор:**

Доцент каф. ПЗ

Коротєєва Т. О.

**Виконав:**

ст. гр. ПЗ-11

Солтисюк Д. А.

**Прийняла:**

Доцент каф. ПЗ

Коротєєва Т. О.

« \_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_\_ \_\_\_\_\_\_\_\_\_ .

Львів – 2022

**Тема**: Наслідування. Створення та використання ієрархії класів

**Мета**: Навчитися створювати базові та похідні класи, використовувати

наслідування різного типу доступу, опанувати принципи використання

множинного наслідування. Навчитися перевизначат и методи в похідному

класі, освоїти принципи такого перевизначення.

**Теоретичні відомості**

Наслідуванням називається процес визначення класу на основі іншого

класу. На новий (дочірній) клас за замовчуванням поширюються всі

визначення змінних екземпляра і методів зі старого (батьківського) класу,

але можуть бути також визначені нові компоненти або «перевизначені»

визначення батьківських функцій і дано нові визначення. Прийнято

вважати, що клас А успадковує свої визначення від класу В, якщо клас А

Визначений на основі класу В зазначеним способом.

Класи можуть бути пов'язані один з одним різними відношеннями.

При наслідуванні всі атрибути і методи батьківського класу успадковуються

Класом нащадком. Наслідування може бути багаторівневим, і тоді

класи, що знаходяться на нижніх рівнях ієрархії, успадкують всі властивості

(атрибути і методи) всіх класів, прямими або непрямими

нащадками яких вони є.

Крім одиничного, існує і множинне наслідування, коли клас наслідує

відразу кілька класів. При цьому він успадкує властивості всіх класів,

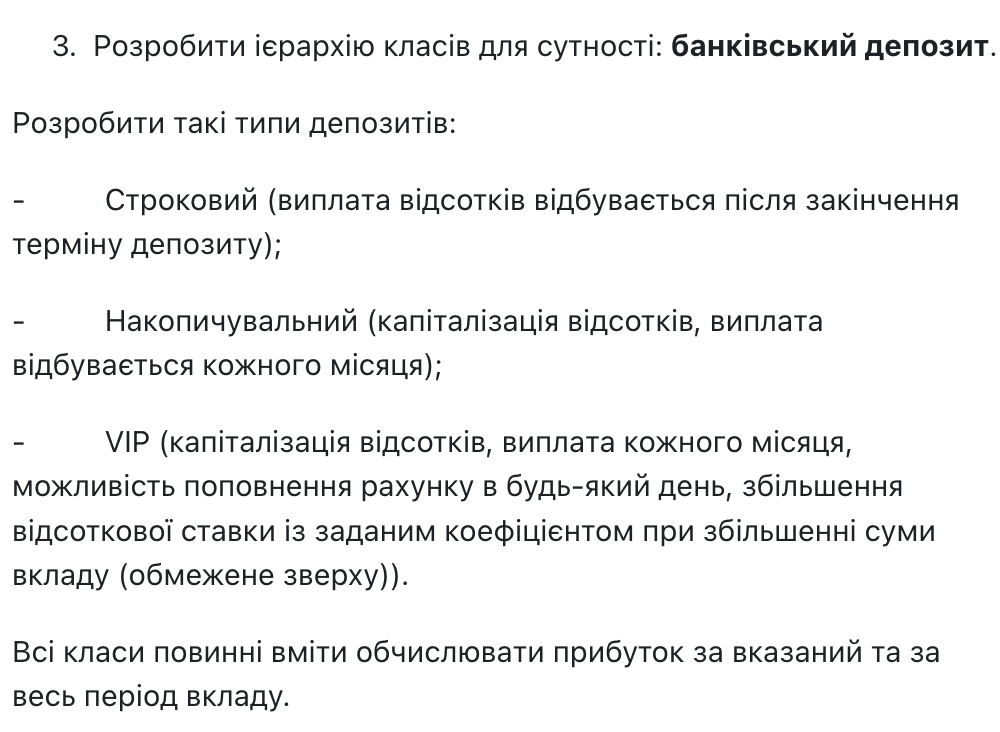
нащадком яких він є. Така зміна семантики методу називається

поліморфізмом. Поліморфізм – це виконання методом з

одним і тим же ім'ям різних дій залежно від контексту, зокрема, від

приналежності до того чи іншого класу.

У різних мовах програмування поліморфізм реалізується різними способами.

**Завдання. Варіант №3**

**Хід роботи**

**Код програми:**

bankdeposit.cpp:

#include "bankdeposit.h"

BankDeposit::BankDeposit(long \_open\_date, long \_terminate\_date, double \_rate,

double \_amount) {

open\_date = \_open\_date;

termination\_date = \_terminate\_date;

rate = \_rate;

amount = \_amount;

}

std::string BankDeposit::print\_class\_name() { **return** "BankDeposit"; }

bankdepostit.h:

#pragma once

#include <string>

class BankDeposit {

protected:

long open\_date;

long termination\_date;

double rate;

double amount;

public:

BankDeposit(long \_open\_date, long \_terminate\_date, double \_rate,

double \_amount);

virtual double calculate\_income(long to) = 0;

virtual double calculate\_income() = 0;

virtual std::string print\_class\_name() = 0;

};

complexbankdeposit.cpp:

#include "complexbankdeposit.h"

ComplexBankDeposit::ComplexBankDeposit(long \_open\_date, long \_terminate\_date,

double \_rate, double \_amount)

: BankDeposit(\_open\_date, \_terminate\_date, \_rate, \_amount) {}

double ComplexBankDeposit::calculate\_income(long to) {

auto month\_count = (to - **this**->open\_date) / (60. \* 60 \* 24 \* 30);

auto progressive\_amount = **this**->amount;

**for** (auto i = 0; i < month\_count; i++) {

progressive\_amount += progressive\_amount \* rate;

}

**return** progressive\_amount - **this**->amount;

}

double ComplexBankDeposit::calculate\_income() {

**return this**->calculate\_income(**this**->termination\_date);

}

std::string ComplexBankDeposit::print\_class\_name() {

**return** "ComplexBankDeposit";

}

complexbankdeposit.h:

#pragma once

#include "bankdeposit.h"

class ComplexBankDeposit : public BankDeposit {

public:

ComplexBankDeposit(long \_open\_date, long \_terminate\_date, double \_rate,

double \_amount);

std::string print\_class\_name() override;

double calculate\_income(long to) override;

double calculate\_income() override;

};

main.cpp:

#include <QApplication>

#include "widget.h"

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

QApplication a(argc, argv);

Widget w;

w.show();

**return** a.exec();

}

simplebankdeposit.cpp:

#include "simplebankdeposit.h"

SimpleBankDeposit::SimpleBankDeposit(long \_open\_date, long \_terminate\_date,

double \_rate, double \_amount)

: BankDeposit(\_open\_date, \_terminate\_date, \_rate, \_amount) {}

double SimpleBankDeposit::calculate\_income(long to) {

auto month\_count = (to - **this**->open\_date) / (60. \* 60 \* 24 \* 30);

**return this**->amount \* rate \* month\_count;

}

double SimpleBankDeposit::calculate\_income() {

**return this**->calculate\_income(**this**->termination\_date);

}

std::string SimpleBankDeposit::print\_class\_name() {

**return** "SimpleBankDeposit";

}

simplebankdeposit.h:

#pragma once

#include "bankdeposit.h"

class SimpleBankDeposit : public BankDeposit {

public:

SimpleBankDeposit(long \_open\_date, long \_terminate\_date, double \_rate,

double \_amount);

std::string print\_class\_name() override;

double calculate\_income(long to) override;

double calculate\_income() override;

};

vipbankdeposit.cpp:

#include "vipbankdeposit.h"

VIPBankDeposit::VIPBankDeposit(long \_open\_date, long \_terminate\_date,

double \_rate, double \_amount)

: ComplexBankDeposit(\_open\_date, \_terminate\_date, \_rate, \_amount) {}

void VIPBankDeposit::add\_money(double money) {

**this**->amount += money;

**this**->rate = std::min(.02, **this**->rate + money / 1000000);

}

std::string VIPBankDeposit::print\_class\_name() { **return** "VIPBankDeposit"; }

vipbankdeposit.h:

#pragma once

#include "complexbankdeposit.h"

class VIPBankDeposit : public ComplexBankDeposit {

public:

VIPBankDeposit(long \_open\_date, long \_terminate\_date, double \_rate,

double \_amount);

void add\_money(double);

std::string print\_class\_name() override;

};

widget.cpp:

#include "widget.h"

#include <QFile>

#include <QGridLayout>

#include <QTextStream>

#include <vector>

#include "bankdeposit.h"

#include "complexbankdeposit.h"

#include "simplebankdeposit.h"

#include "vipbankdeposit.h"

void Widget::on\_output() {

auto terminate\_date = 60 \* 60 \* 24 \* 30 \* 3;

auto monthly\_rate = .01;

auto initial\_amount = 10000;

SimpleBankDeposit d1(0, terminate\_date, monthly\_rate, initial\_amount);

ComplexBankDeposit d2(0, terminate\_date, monthly\_rate, initial\_amount);

VIPBankDeposit d3(0, terminate\_date, monthly\_rate, initial\_amount);

d3.add\_money(1000000);

**this**->class\_names\_output->setMarkdown(

QString("### BankDeposit classes:\n\n"

"\* %1\n"

"\* %2\n"

"\* %3")

.arg(QString::fromStdString(d1.print\_class\_name()))

.arg(QString::fromStdString(d2.print\_class\_name()))

.arg(QString::fromStdString(d3.print\_class\_name())));

auto output\_string =

QString("### Parameters\n"

"`monthly rate = %4, initial amount = %5, months = %6`:\n"

"\n"

"### Results\n"

"\* Simple Deposit income: %1\n"

"\* Complex Deposit income: %2\n"

"\* VIP Deposit income: %3")

.arg(QString::number(d1.calculate\_income()))

.arg(QString::number(d2.calculate\_income()))

.arg(QString::number(d3.calculate\_income()))

.arg(QString::number(monthly\_rate))

.arg(QString::number(initial\_amount))

.arg(QString::number(terminate\_date / (60 \* 60 \* 24 \* 30)));

QFile file("results.md");

**if** (file.open(QIODevice::Append)) {

QTextStream stream(&file);

stream << output\_string << Qt::endl << Qt::endl;

file.close();

}

**this**->results\_output->setMarkdown(output\_string);

}

Widget::Widget(QWidget \*parent) : QWidget(parent) {

auto \*main\_layout = **new** QGridLayout;

**this**->output\_btn = **new** QPushButton("Print output and save to file");

**this**->class\_names\_output = **new** QTextEdit;

**this**->class\_names\_output->setReadOnly(**true**);

**this**->results\_output = **new** QTextEdit;

**this**->results\_output->setReadOnly(**true**);

main\_layout->addWidget(**this**->class\_names\_output, 0, 0);

main\_layout->addWidget(**this**->results\_output, 0, 1);

main\_layout->addWidget(**this**->output\_btn, 1, 0, 1, 2);

connect(**this**->output\_btn, &QPushButton::released, **this**, &Widget::on\_output);

setLayout(main\_layout);

}

widget.h:

#pragma once

#include <QPushButton>

#include <QTextEdit>

#include <QWidget>

class Widget : public QWidget {

Q\_OBJECT

public:

Widget(QWidget \*parent = nullptr);

private slots:

void on\_output();

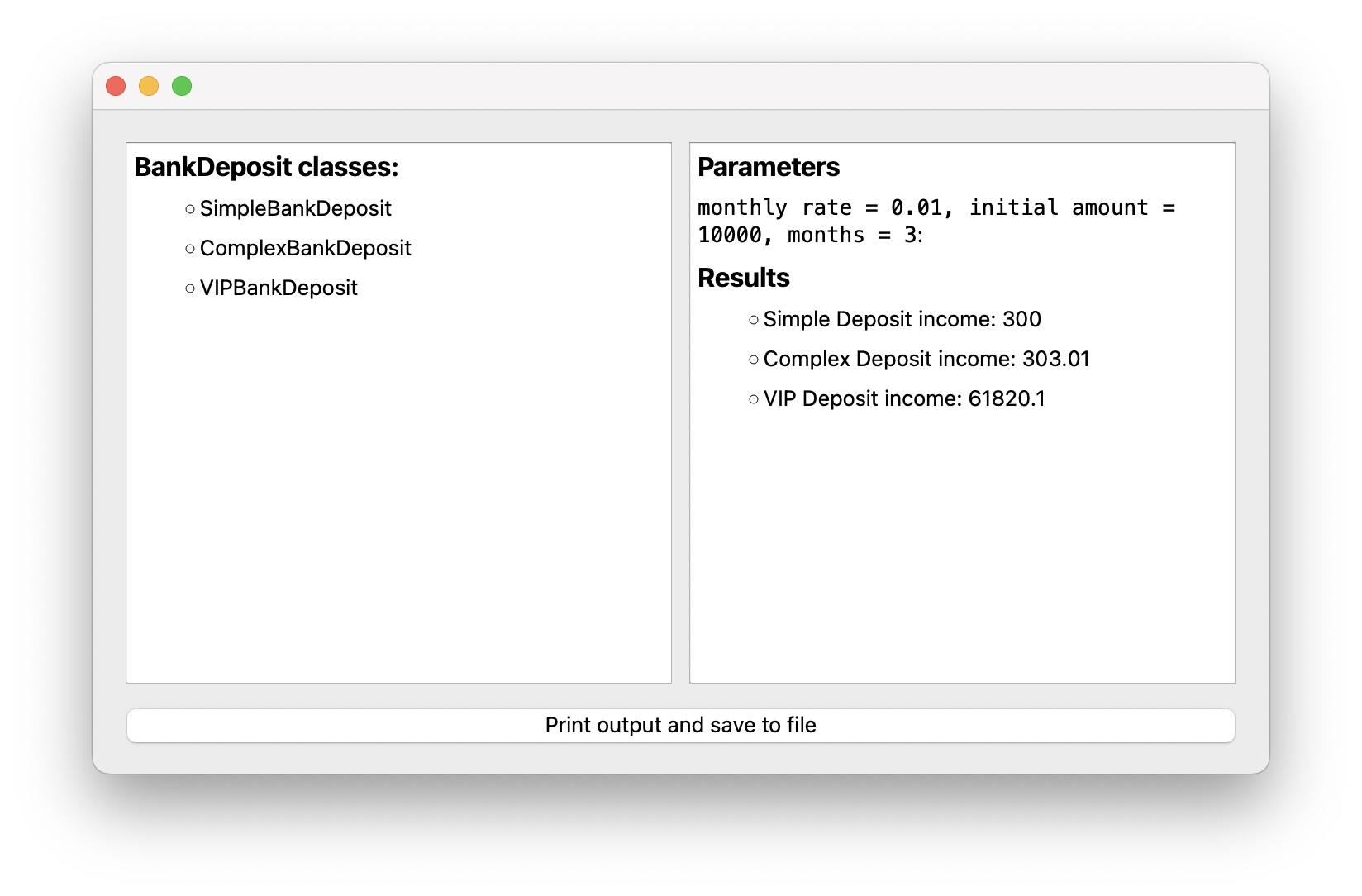
private:

QPushButton \*output\_btn;

QTextEdit \*class\_names\_output;

QTextEdit \*results\_output;

};



**Результати виконання програми**

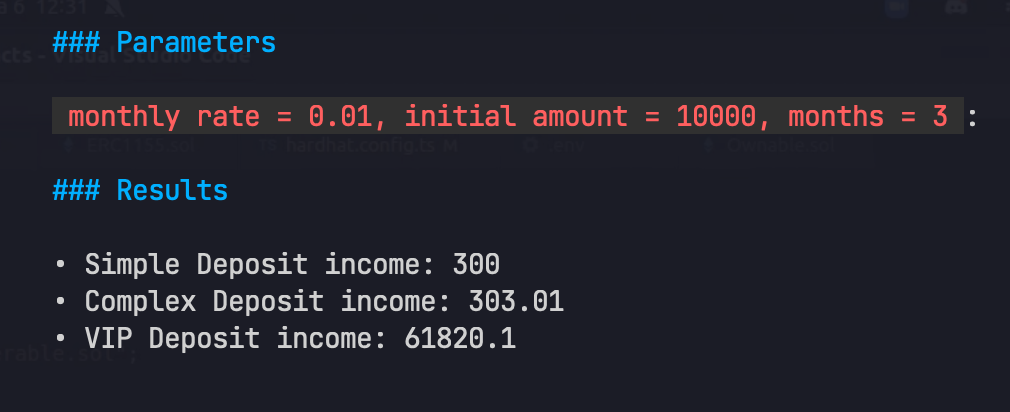
Рис. 1. Результати обчислень програми

Рис. 2. Вивід результату у форматі markdown у файл

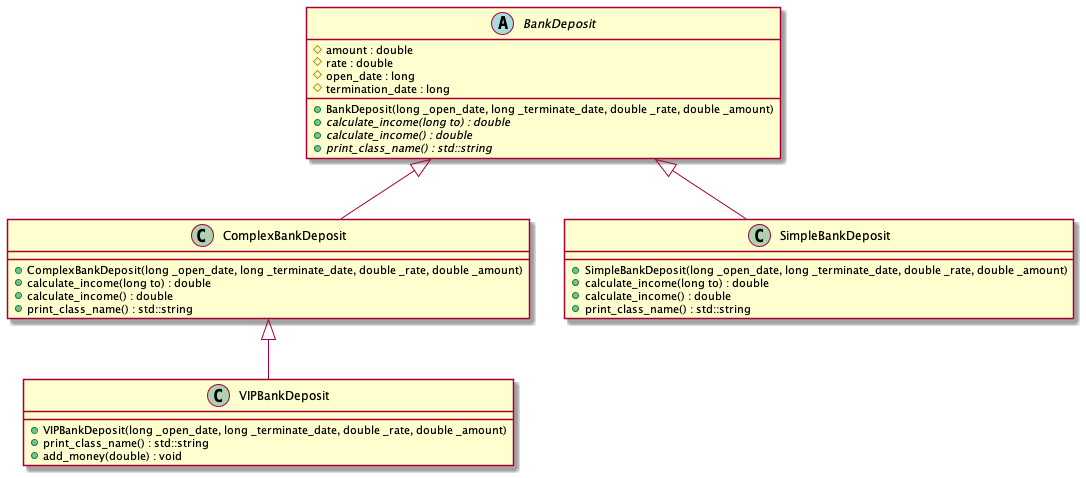


Рис. 3. Схема класів

**Висновок**

Виконуючи цю лабораторну роботу, я навчився створювати базові та похідні класи,

Використовувати наслідування різного типу доступу, опанував принципи

використання множинного наслідування. Навчився

перевизначати методи в похідному класі, освоїв принципи такого перевизначення.