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
Ясногородський Н. В.
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

3. Розробити ієрархію класів для сутності: **банківський депозит**.

Розробити такі типи депозитів:

- Строковий (виплата відсотків відбувається після закінчення терміну депозиту);

- Накопичувальний (капіталізація відсотків, виплата відбувається кожного місяця);

- VIP (капіталізація відсотків, виплата кожного місяця, можливість поповнення рахунку в будь-який день, збільшення відсоткової ставки із заданим коефіцієнтом при збільшенні суми вкладу (обмежене зверху)).

Всі класи повинні вміти обчислювати прибуток за вказаний та за весь період вкладу.

Хід роботи

Код програми:

main.cpp:

```
#include <QApplication>

#include "widget.h"

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

deposit.cpp:

```
#include "deposit.h"

#include <algorithm>

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"; }

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";
}

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;
}
```

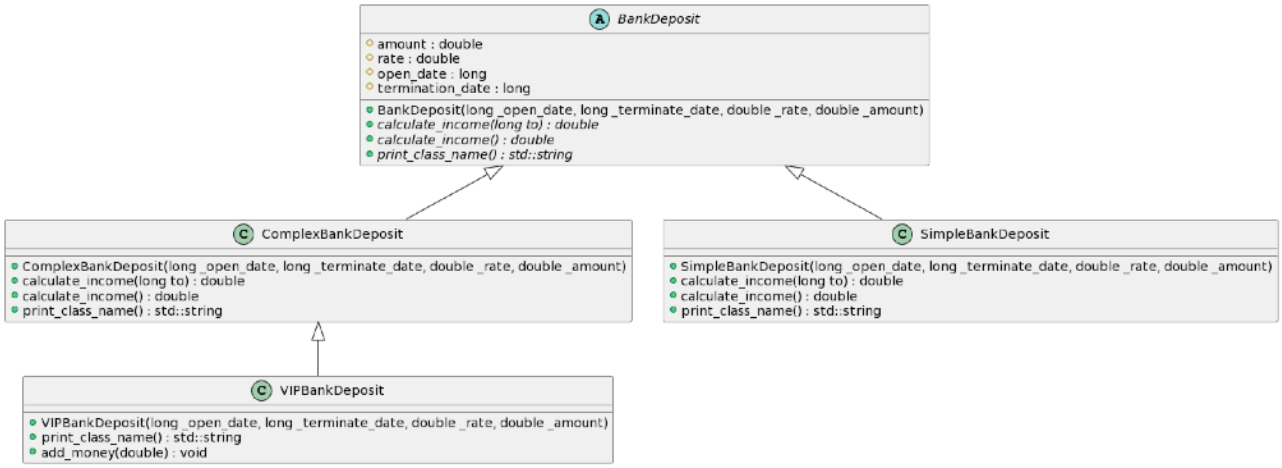
```
double ComplexBankDeposit::calculate_income() {
  return this->calculate_income(this->termination_date);
}

std::string ComplexBankDeposit::print_class_name() {
  return "ComplexBankDeposit";
}

VIPBankDeposit::VIPBankDeposit(long _open_date, long _terminate_date,
                               double _rate, double _amount)
    : ComplexBankDeposit(_open_date, _terminate_date, _rate, _amount) {}

std::string VIPBankDeposit::print_class_name() { return "VIPBankDeposit"; }

void VIPBankDeposit::add_money(double money) {
  this->amount += money;
  this->rate = std::min(.02, this->rate + money / 1000000);
}
```

widget.cpp:

```
#include "widget.h"

#include <QFile>
#include <QGridLayout>
#include <QTextStream>

#include "deposit.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(
          "### Results\n"
          "`monthly rate = %4, initial amount = %5, months = %6`:\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");

  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->output_btn, 0, 1);
  main_layout->addWidget(this->results_output, 0, 2);

  connect(this->output_btn, &QPushButton::released, this, &Widget::on_output);

  setLayout(main_layout);
}
```

Діаграмма:



Результати виконання програми

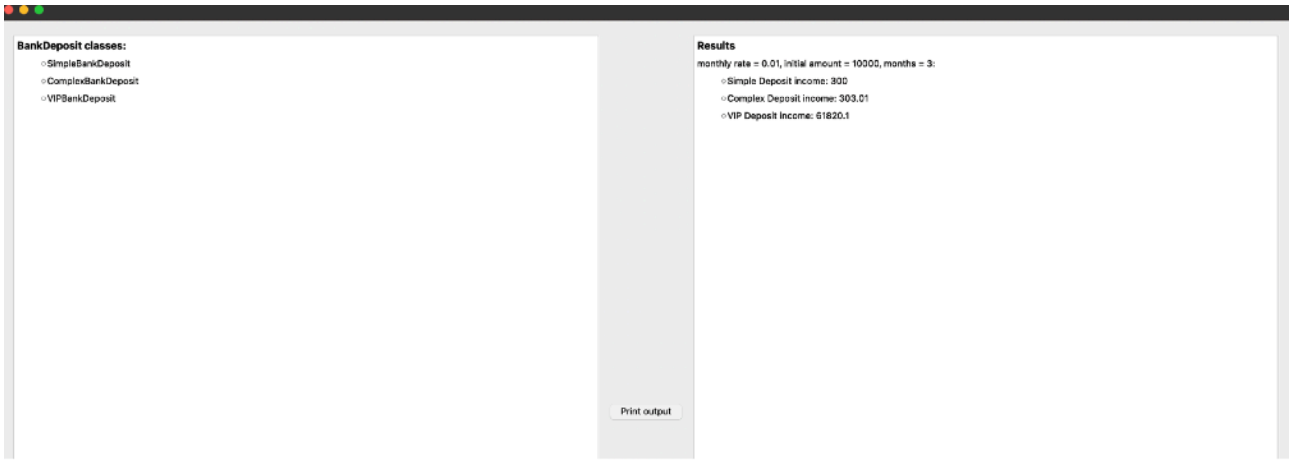


Рис. 1. Результати обчислень програми

```
### Results

 monthly rate = 0.01, initial amount = 10000, months = 3 :

• Simple Deposit income: 300
• Complex Deposit income: 303.01
• VIP Deposit income: 61820.1

### Results

 monthly rate = 0.01, initial amount = 10000, months = 3 :

• Simple Deposit income: 300
• Complex Deposit income: 303.01
• VIP Deposit income: 61820.1

### Results

 monthly rate = 0.01, initial amount = 10000, months = 3 :

• Simple Deposit income: 300
• Complex Deposit income: 303.01
• VIP Deposit income: 61820.1
```

Рис. 2. Вивід markdowm у файл

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.