**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**



### ЗВІТ

До 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-16

Шеремета А.І.

**Прийняв:**

асист. каф. ПЗ

Дивак І.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

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

**Мета** **роботи:** Навчитися створювати базові та похідні класи, використовувати наслідування різного типу доступу, опанувати принципи використання множинного наслідування. Навчитися перевизначати методи в похідному класі, освоїти принципи такого перевизначення.

**Теоретичні відомості**

Наслідування - це один з принципів об'єктно-орієнтовного програмування, який дає класу можливість використовувати програмний код іншого (базового) класу, доповнюючи його своїми власними деталями реалізації. Іншими словами, під час успадкування відбувається отримання нового (похідного) класу, який містить програмний код базового класу з зазначенням власних особливостей використання.

Існує 3 режими успадкування.

Public: якщо ми виведемо підклас із загальнодоступного базового класу. Тоді відкритий член базового класу стане відкритим у похідному класі, а захищені члени базового класу стануть захищеними в похідному класі.

Protected: якщо ми виведемо підклас із захищеного базового класу. Тоді як відкриті члени, так і захищені члени базового класу стануть захищеними в похідному класі.

Private: якщо ми виведемо підклас із приватного базового класу. Тоді як відкриті члени, так і захищені члени базового класу стануть приватними в похідному класі.

До закритих членів базового класу неможливо отримати прямий доступ у похідному класі, тоді як до захищених членів можна отримати прямий доступ.

class A {

public:

int x;

protected:

int y;

private:

int z;

};

class B : public A {

// x is public

// y is protected

// z is not accessible from B

};

class C : protected A {

// x is protected

// y is protected

// z is not accessible from C

};

class D : private A // 'private' is default for classes

{

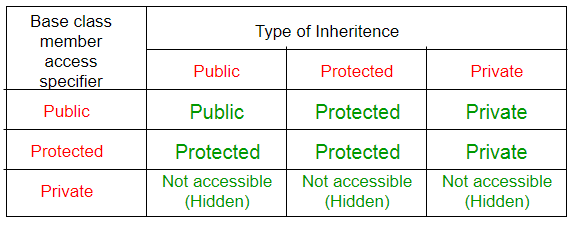
// x is private

// y is private

// z is not accessible from D

};

Таблиця знизу підсумовує три вищезазначені типи та показує специфікатор доступу членів базового класу в підкласі, якщо вони отримані в публічному, захищеному та приватному типах:



**Лабораторне завдання**

1. Розробити ієрархію класів відповідно до варіанту
2. Створити базовий, похідні класи.
3. Використати public, protected наслідування.
4. Використати множинне наслідування (за необхідності).
5. Виконати перевантаження функції print() в базовому класі, яка друкує назву відповідного класу, перевизначити її в похідних. В проекті при натисканні кнопки виведіть на форму назви всіх розроблених класів.
6. Реалізувати методи варіанта та результати вивести на форму і у файл. При записі у файл використати різні варіанти аргументів конструктора.
7. Оформити звіт до лабораторної роботи. Включити у звіт Uml-діаграму розробленої ієрархії класів.

**Варіант 4**

Розробити ієрархію класів для сутності: облік спожитої електроенергії.

Розробити такі моделі обліку:

- Звичайну (вартість спожитої електроенергії обчислюється за фіксованою ціною)

- Пільгову (вартість спожитої електроенергії обчислюється за пільговою ціною у випадку, коли обсяг не перевищує нормованого)

- Багатозонну (вартість спожитої електроенергії обчислюється за підвищеною ціною в час-пік, за зниженою ціною в нічний час і за стандартною в іншому випадку).

Кожен клас повинен мати можливість обліковувати спожиту електроенергію погодинно, обчислювати вартість спожитої електроенергії за заданий період, а також зберігати та завантажувати журнал обліку з файлу.

Набір полів і методів, необхідних для забезпечення функціональної зручності класів, визначити самостійно.

**Хід роботи**

1. Реалізовую програму:

**Файл accounting.cpp**

#include "accounting.h"

#include "ui\_accounting.h"

#include <QPalette>

#include <QComboBox>

#include <default.h>

#include <multizone.h>

#include <discount.h>

#include <QFileDialog>

#include <QDialog>

#include <QMessageBox>

Default A;

Discount B;

Multizone C;

Accounting::Accounting(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Accounting)

{

ui->setupUi(this);

ui->radioButton->hide();

ui->radioButton\_2->hide();

ui->radioButton\_3->hide();

History = this->ui->History;

this->setAutoFillBackground( true );

this->setStyleSheet("background-color: #fcde65;");

ui->comboBox->setStyleSheet("QComboBox { background-color: #6581fc; }");

}

Accounting::~Accounting()

{

delete ui;

}

void Accounting::on\_pushButton\_clicked()

{

switch(ui->comboBox->currentIndex()){

case 0:

A.setKvtH(ui->Kvth->toPlainText().toDouble());

A.calculatePrice();

A << \*History;

break;

case 1:

B.setKvtH(ui->Kvth->toPlainText().toDouble());

B.calculatePrice();

B << \*History;

break;

case 2:

if(ui->radioButton->isChecked() || ui->radioButton\_2->isChecked() || ui->radioButton\_3->isChecked()){

C.setMultiKvtH(ui->Kvth->toPlainText().toDouble());

if(ui->radioButton->isChecked()){

C.setnight();

C.calculatePrice();

C.clearZones();

}

else if(ui->radioButton\_2->isChecked()){

C.setrush();

C.calculatePrice();

C.clearZones();

}

else if(ui->radioButton\_3->isChecked()){

C.calculatePrice();

}

C << \*History;

break;

}

else{

QMessageBox::about(this, "Error" , "Choose option");

}

}

}

void Accounting::on\_comboBox\_currentIndexChanged(int index)

{

if(index == 2){

ui->radioButton->setEnabled(true);

ui->radioButton->show();

ui->radioButton\_2->setEnabled(true);

ui->radioButton\_2->show();

ui->radioButton\_3->setEnabled(true);

ui->radioButton\_3->show();

}

else{

ui->radioButton->setEnabled(false);

ui->radioButton->hide();

ui->radioButton\_2->setEnabled(false);

ui->radioButton\_2->hide();

ui->radioButton\_3->setEnabled(false);

ui->radioButton\_3->hide();

}

}

void Accounting::on\_pushButton\_2\_clicked()

{

switch(ui->comboBox->currentIndex()){

case 0:{

QString str1 = QString("Сума за тарифом \"Звичайний\" = " + QString::number(A.gettotalPrice()));

ui->History->addItem(str1);

break;}

case 1:{

QString str2 = QString("Сума за тарифом \"Пільговий\" = " + QString::number(B.gettotalPrice()));

ui->History->addItem(str2);

break;}

case 2:{

QString str3 = QString("Сума за тарифом \"Багатозонний\" = " + QString::number(C.gettotalPrice()));

ui->History->addItem(str3);

break;}

}

}

void Accounting::on\_pushButton\_3\_clicked()

{

int months = ui->textEdit->toPlainText().toInt();

switch(ui->comboBox->currentIndex()){

case 0:{

QString strsum1 = QString("Сума за останні " + QString::number(months) + " місяці за тарифом \"Звичайний\" = " + QString::number(A.calculatePeriod(months)));

ui->History->addItem(strsum1);

break;}

case 1:{

QString strsum2 = QString("Сума за останні " + QString::number(months) + " місяці за тарифом \"Пільговий\" = " + QString::number(B.calculatePeriod(months)));

ui->History->addItem(strsum2);

break;}

case 2:{

QString strsum3 = QString("Сума за останні " + QString::number(months) + " місяці за тарифом \"Багатозонний\" = " + QString::number(C.calculatePeriod(months)));

ui->History->addItem(strsum3);

break;}

}

}

void Accounting::on\_pushButton\_4\_clicked()

{

QString fileName = QFileDialog::getSaveFileName(this, tr("Save File"), "",

tr("Text File (\*.txt);;All Files (\*)"));

QFile file(fileName);

if (file.open(QIODevice::WriteOnly | QIODevice::Text)) {

QTextStream stream(&file);

for (int i = 0; i < ui->History->count(); i++) {

QListWidgetItem \*item = ui->History->item(i);

stream << item->text() << '\n';

}

file.close();

}

}

void Accounting::on\_pushButton\_5\_clicked()

{

QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"), "",

tr("Text File (\*.txt);;All Files (\*)"));

QFile file(fileName);

if (file.open(QIODevice::ReadOnly | QIODevice::Text)) {

QTextStream in(&file);

while(!in.atEnd()) {

QString line = in.readLine();

ui->History->addItem(line);

}

file.close();

}

}

void Accounting::on\_pushButton\_6\_clicked()

{

ui->History->clear();

A.clear();

B.clear();

C.clear();

}

**Файл default.cpp**

#include "default.h"

Default::Default()

{

}

void Default::calculatePrice(){

monthPrice = KvtH \* pricePerKvtH;

totalPrice += monthPrice;

}

void Default::operator<<(QListWidget &item){

QString str;

str = "Обсяг спожитої енергії: " + QString::number(KvtH) + " за тарифом \"Звичайний\" (" + QString::number(pricePerKvtH) + " грн за кВт/год); " + "Сума = " + QString::number(monthPrice);

item.addItem(str);

history[step] = monthPrice;

step++;

}

double Default::calculatePeriod(int Months){

double Sum = 0;

int id = step;

for (int i = 0; i < Months; i++){

Sum += history[id];

id--;

}

return Sum;

}

void Default::clear(){

this->step = 0;

this->sumPrice=0;

this->monthPrice = 0;

this->KvtH = 0;

this->totalPrice = 0;

for(int i = 0; i < 50; i++){

this->history[i] = 0;

}

}

**Файл discount.cpp**

#include "discount.h"

Discount::Discount()

{

}

void Discount::calculatePrice(){

double temp = 0.0;

if (getKvtH() > 250){

setpricePerKvtH(1.68);

setmonthPrice(getKvtH() \* getpricePerKvtH());

temp += getmonthPrice();

settotalPrice(temp);

}

else{

setpricePerKvtH(1.44);

setmonthPrice(getKvtH() \* getpricePerKvtH());

temp += getmonthPrice();

settotalPrice(temp);

}

}

void Discount::operator<<(QListWidget &item){

QString str;

str = "Обсяг спожитої енергії: " + QString::number(getKvtH()) + " за тарифом \"Пільговий\" (" + QString::number(getpricePerKvtH()) + " грн за кВт/год); " + "Сума = " + QString::number(getmonthPrice());

item.addItem(str);

history[step] = getmonthPrice();

step++;

}

**Файл multizone.cpp**

#include "multizone.h"

Multizone::Multizone()

{

}

void Multizone::calculatePrice(){

double temp = 0.0;

if(rush){

setpricePerKvtH(1.68);

setmonthPrice(getKvtH() \* getpricePerKvtH());

temp += getmonthPrice();

settotalPrice(temp);

}

else if(night){

setpricePerKvtH(1.3);

setmonthPrice(getKvtH() \* getpricePerKvtH());

temp += getmonthPrice();

settotalPrice(temp);

}

else{

setpricePerKvtH(1.44);

setmonthPrice(getKvtH() \* getpricePerKvtH());

temp += getmonthPrice();

settotalPrice(temp);

}

}

void Multizone::operator<<(QListWidget &item){

QString str;

str = "Обсяг спожитої енергії: " + QString::number(getKvtH()) + " за тарифом \"Багатозонний\" (" + QString::number(getpricePerKvtH()) + " грн за кВт/год); " + "Сума = " + QString::number(getmonthPrice());

item.addItem(str);

history[step] = getmonthPrice();

step++;

}

void Multizone::setnight(){

this->night = true;

}

void Multizone::setrush(){

this->rush = true;

}

void Multizone::setMultiKvtH(double Kvth){

KvtH = Kvth;

}

void Multizone::clearZones(){

this->night = false;

this->rush = false;

}

double Multizone::gettotalPrice(){

return totalPrice;

}

double Multizone::calculatePeriod(int Months){

double Sum = 0;

int id = step;

for (int i = 0; i < Months; i++){

Sum += history[id];

id--;

}

return Sum;

}

void Multizone::clear(){

this->step = 0;

this->sumPrice=0;

this->monthPrice = 0;

this->KvtH = 0;

this->totalPrice = 0;

for(int i = 0; i < 50; i++){

this->history[i] = 0;

}

}

**Файл accounting.h**

#ifndef ACCOUNTING\_H

#define ACCOUNTING\_H

#include <QMainWindow>

#include <QListWidget>

QT\_BEGIN\_NAMESPACE

namespace Ui { class Accounting; }

QT\_END\_NAMESPACE

class Accounting : public QMainWindow

{

Q\_OBJECT

public:

Accounting(QWidget \*parent = nullptr);

~Accounting();

private slots:

void on\_pushButton\_clicked();

void on\_comboBox\_currentIndexChanged(int index);

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

void on\_pushButton\_5\_clicked();

void on\_pushButton\_6\_clicked();

private:

Ui::Accounting \*ui;

QListWidget \*History;

};

#endif // ACCOUNTING\_H

**Файл default.h**

#ifndef DEFAULT\_H

#define DEFAULT\_H

#include <QListWidget>

class Default

{

public:

Default();

void calculatePrice();

void setpricePerKvtH(double price){

this->pricePerKvtH = price;

}

void setKvtH(double Kvth){

this->KvtH = Kvth;

};

void setmonthPrice(double monthPrice){

this->monthPrice = monthPrice;

};

void setsumPrice(double sumPrice){

this->sumPrice = sumPrice;

};

void settotalPrice(double totalPrice){

this->totalPrice = totalPrice;

};

double calculatePeriod(int Months);

double getpricePerKvtH(){return pricePerKvtH;};

double getKvtH(){return KvtH;};

double getmonthPrice(){return monthPrice;};

double getsumPrice(){return sumPrice;};

double gettotalPrice(){

totalPrice+=monthPrice;

return totalPrice;};

void operator<<(QListWidget &item);

void clear();

protected:

int step = 0;

double sumPrice,

monthPrice,

pricePerKvtH = 1.44,

KvtH,

totalPrice,

history[50];

};

#endif // DEFAULT\_H

**Файл discount.h**

#ifndef DISCOUNT\_H

#define DISCOUNT\_H

#include <default.h>

class Discount :public Default

{

public:

Discount();

double totalPrice;

double gettotalPrice(){

totalPrice+=monthPrice;

return totalPrice;};

void calculatePrice();

void operator<<(QListWidget &item);

};

#endif // DISCOUNT\_H

**Файл multizone.h**

#ifndef MULTIZONE\_H

#define MULTIZONE\_H

#include <default.h>

class Multizone : protected Default

{

public:

Multizone();

bool rush, night;

double totalPrice;

void setMultiKvtH(double KvtH);

void calculatePrice();

void setnight();

void setrush();

void clearZones();

double gettotalPrice();

void operator<<(QListWidget &item);

double calculatePeriod(int Months);

void clear();

};

#endif // MULTIZONE\_H

**Виконання лабораторної роботи**

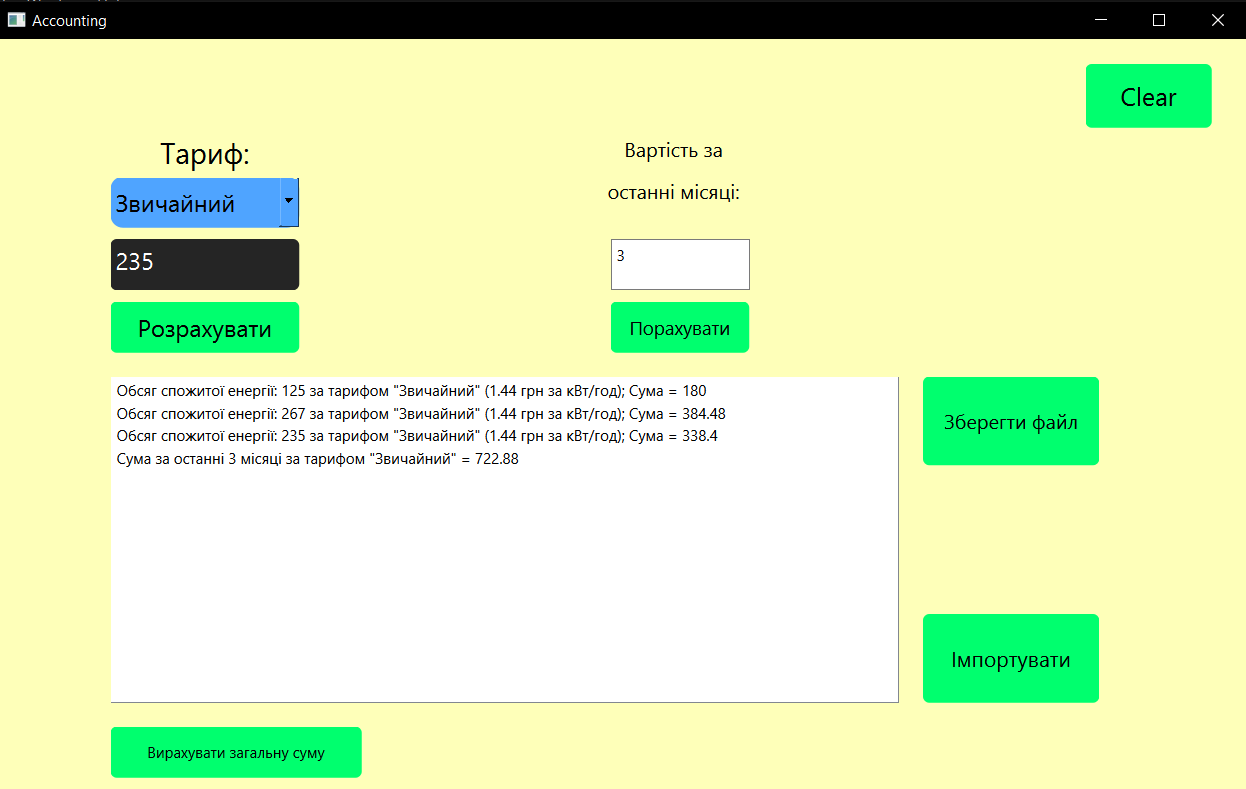


Рис. 1 Виконання програми

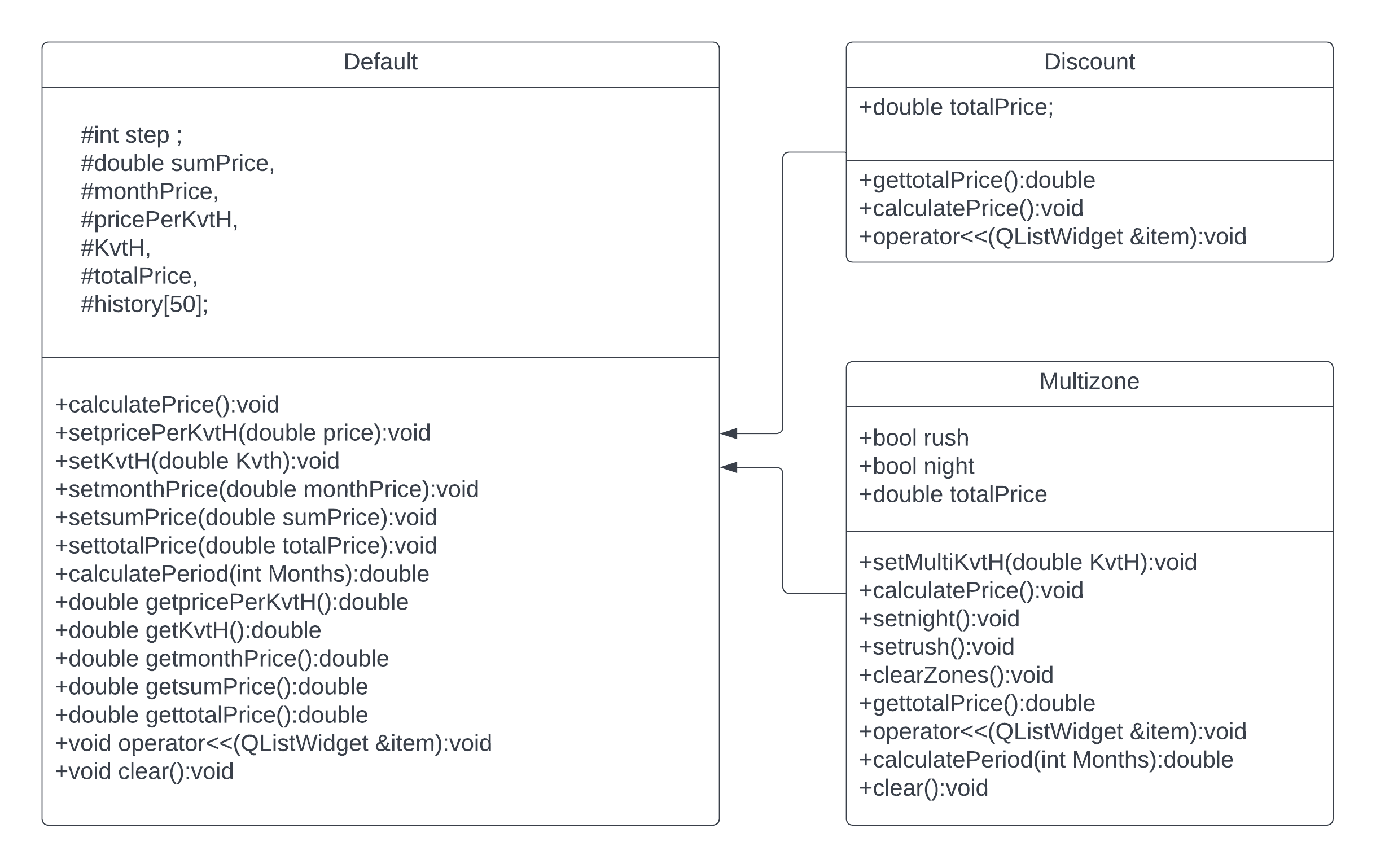


Рис.2 UML-діаграма

**Висновки**

На цій лабораторній роботі я навчився використовувати базові та похідні класи, використовувати наслідування різного типу доступу, опанувати принципи використання множинного наслідування. Ознайомився з перевизначенням методів в похідному класі.