СТАТИЧНІ МЕТОДИ, ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ ТА МЕТОДІВ

Лабораторна робота №4

Мета:

- Навчитись доречно використовувати статичні методи а також використовувати перевантаження методів та операторів.

1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Необхідно визначити у класі GraphScreen статичний метод onTimerAction(). Цей метод відображатиме на екрані заданий нащадок Window.

Обрати для Win32- таймера власний інтервал повторних викликів. Встановити реалізовний метод onTimerAction() на виклик у таймері. Таймер повинен спрацювати лише 4 рази. Метод повинен виводити на екран дані про поточний асоційований об'єкт даних.

Реалізувати перевантажені оператори та методи згідно індивідуального завдання.

1	Віконні елементи	SetData(const Field1_data2_type&); SetData(const Field1_data2_type&, const Field2_data2_type&);	<Data1>::operator>(const <Data1>&) – повертає true, якщо площа вікна об'єкта більша за площу об'єкта-аргумента <Data1>::operator+=(const <Data1>&) – збільшує розміри об'єкта на Height() та Width() аргумента

РОЗРОБКА ПРОГРАМИ

2.1 Ієрархія та структура класів

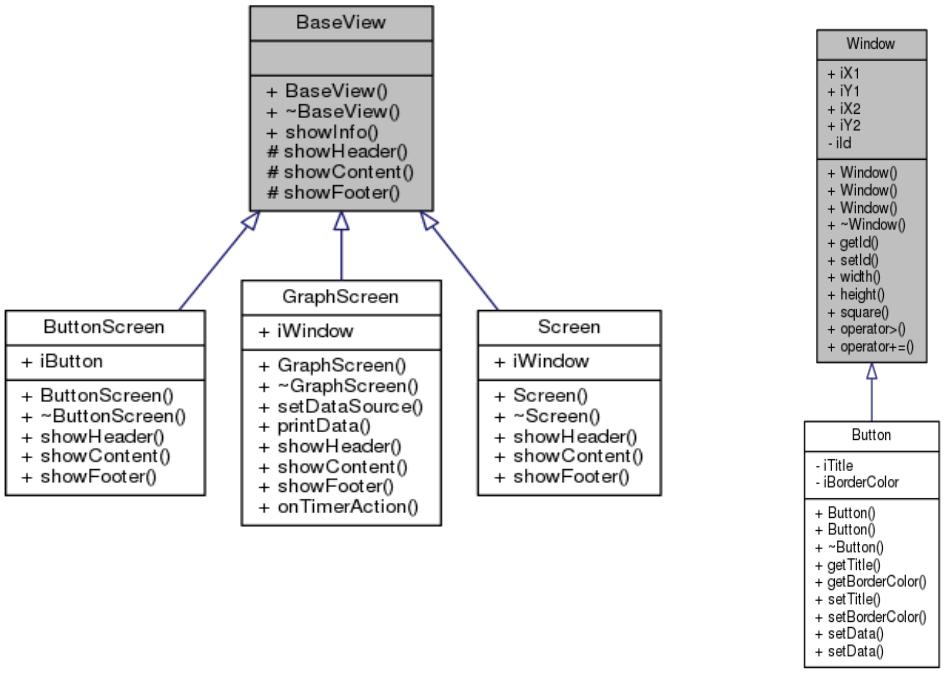

Рисунок 2.1 — Ієрархія класів

2.2 Опис програми

На рис.2.2 наведена структура розробленого проекту

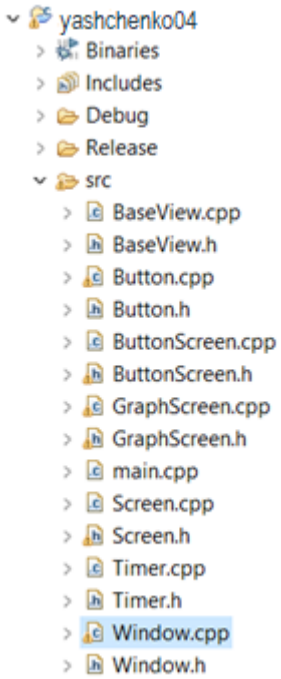

Рисунок 2.2 — Структура проекту

Призначення спроектованих класів наведено на рис. 2.3.

BaseView	Base View class
Button	Stores information about button
ButtonScreen	Shows information about button
GraphScreen	Prints information about window with pseudographics
Screen	Shows information about window
Timer	Doing some actions each interval of time
Window	Storing information about window

Рисунок 2.3 — Призначення класів, створене за допомогою Javadoc

2.3 Фрагменти програми

2.3.1 Button.h

```
/**
 * Sets Button::iTitle.
 * @param aTitle text to be set in iTitle
 */
void setData(const char* aTitle);
/**
 * Sets Button::iTitle and Button::iBorderColor.
 * @param aTitle text to be set in iTitle
 * @param aBorderColor text to be set in iBorderColor
 */
void setData(const char* aTitle, const char* aBorderColor);
```

2.3.2 Button.cpp

```
void Button::setData(const char* aTitle) {
```

```
		strcpy(iTitle, aTitle);
}
void Button::setData(const char* aTitle, const char* aBorderColor) {
		strcpy(iTitle, aTitle);
		strcpy(iBorderColor, aBorderColor);
}
```

2.3.3 Window.h

```
/**
 * Counts square of window.
 * @return square
 */
const int square() const;
/**
 * Checking if current object bigger than argument.
 * @param aWindow link to Window object
 * @return true or false
 */
bool operator>(const Window & aWindow);
/**
 * Adding width and heigth of argument to current object.
 * @param aWindow link to Window object
 */
void operator+=(const Window & aWindow);
```

2.3.4 Window.cpp

```
const int Window::square() const {
		return width() * height();
}
bool Window::operator >(const Window & aWindow) {
		if (square() > aWindow.square())
				return true;
		else
				return false;
}
void Window::operator +=(const Window & aWindow) {
		iY2 += aWindow.height();
		iX2 += aWindow.width();
}
```

2.3.5 GraphScreen.h

```
/**
 * Doing some actions every timer interval.
 * @param window pointer to Window object
 */
static void onTimerAction(Window *window);
```

2.3.6 GraphScreen.cpp

```
void GraphScreen::onTimerAction(Window * window) {
		cout << "onTimerAction output!" << endl;
}
```

2.3.6 Timer.h

```
/**
 * @file Timer.h
 * Declaration of class Timer.
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.27
 */
#ifndef SCREEN_H_
#define SCREEN_H_

#define _WIN32_WINNT 0x0600

#include "Windows.h"
```

```
/**
 * Doing some actions each interval of time.
 */
class Timer{
public:
	/**
	 * Runs the timer.
	 * @param aInterval interval of time.
	 * @param aCntActions number of actions to do.
	 * @return result code.
	 */
	int start(UINT aInterval, int aCntActions);
};
#endif
```

2.3.7 Timer.cpp

```
#include "Timer.h"
#include "GraphScreen.h"
#include <iostream>
using namespace std;

int Timer::start(UINT aInterval, int aCntActions) {
	// Set the timer.
	UINT_PTR timer = SetTimer(NULL, NULL, aInterval,
			(TIMERPROC) GraphScreen::onTimerAction);
	if (timer == NULL) {
		cout << "SetTimer failed (%lu)\n" << GetLastError() << endl;
		return 1;
	}

	MSG msg;

	while (aCntActions--) {
		GetMessage(&msg, 0, 0, 0);
		DispatchMessage(&msg);
	}

	if (0 == KillTimer(NULL, timer)) {
		return 1;
	}
	return 0;
}
```

2.3.8 main.cpp

```
int main(int argc, char** argv) {
	Button button(3, 30, 50, 80, 90, "Nice button", "red");
	Button button2(5, 15, 15, 35, 35, "Beautiful button", "green");
	BaseView * view1 = new ButtonScreen(button);
	BaseView * view11 = new ButtonScreen(button2);
	GraphScreen gscreen(&button);
	view1->showInfo();
	cout << endl;
	if (button > button2)
		cout << "Button > button2" << endl;
	cout << endl;
	view11->showInfo();
	cout << endl;
	button.setData("Red button");
	button2.setData("BIG button", "black");
	button += button2;
	BaseView * view2 = new ButtonScreen(button);
	BaseView * view22 = new ButtonScreen(button2);
	view2->showInfo();
	cout << endl;
	view22->showInfo();
	Timer timer1;
	timer1.start(2000, 4);
	delete view1;
	delete view11;
	delete view2;
```

```
        delete view22;
        return 0;
}
```

3 РЕЗУЛЬТАТИ РОБОТИ

```
Button info:
ID=3
X1=30
Y1=50
X2=80
Y2=90
Width=50
Height=40
Button text=Nice button
Border Color=red
ButtonScreen output...

Button > button2

Button info:
ID=5
X1=15
Y1=15
X2=35
Y2=35
Width=20
Height=20
Button text=Beautiful button
Border Color=green
ButtonScreen output...

Button info:
ID=3
X1=30
Y1=50
X2=100
Y2=110
Width=70
Height=60
Button text=Red button
Border Color=red
ButtonScreen output...

Button info:
ID=5
X1=15
Y1=15
X2=35
Y2=35
Width=20
Height=20
Button text=BIG button
Border Color=black
ButtonScreen output...
onTimerAction output!
onTimerAction output!
onTimerAction output!
onTimerAction output!
```

Рисунок 3.1 - Приклад роботи програми

ВИСНОВКИ

В розробленій програмі було реалізовано перевантажений метод setData() для встановлення даних об'єктів класу Button, перевантажено оператори > та += у класі Window, а також метод onTimerAction() для

поінтервального виклику цього методу у таймері.