

ПРАВА ДОСТУПУ, const, ПОКАЖЧИКИ, ПОСИЛАННЯ

Лабораторна робота №2

Мета:

- отримати навички при передаванні об'єктів у класи із застосування прав доступу та const-модифікаторів.

1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Розподілити в Window права доступу private, public.

Реалізувати клас GraphScreen основна задача котрого полягає у більш багатому відображенні даних Window із застосуванням псевдографіки для наочного відображення пов'язаного об'єкта.

Оновити Window для збереження функціональності цього класу при роботі із оновленим Window.

1	Віконні елементи	Вікно (Window)	private: field1_data1 = ID вікна; public: field2_data1=x1, public: field3_data1=y1, public: field4_data1=x2, public: field5_data1=y2	Экран (GraphScreen), текстово- псевдографічне відображення даних field1_view2=const CWindow* iWindow;	<Data1>::Width() – визначення ширини вікна <Data1>::Height() – визначення висоти вікна
---	------------------	----------------	---	--	---

2 РОЗРОБКА ПРОГРАМИ

Для реалізації програми було розроблено ще один клас: GraphScreen.

2.1 Ієрархія та структура класів

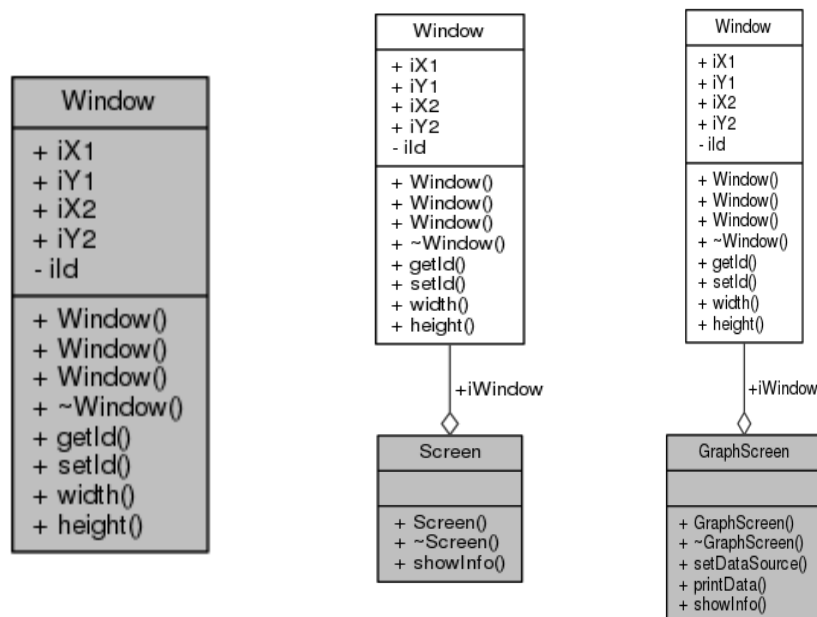


Рисунок 2.1 — Ієрархія класів

2.2 Опис програми

На рис.2.2 наведена структура розробленого проекту

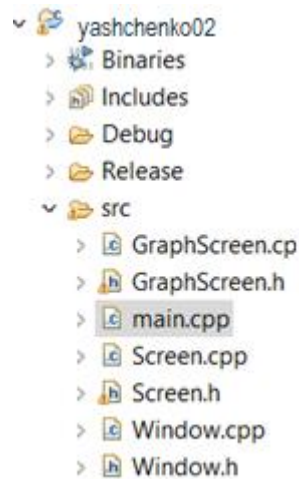


Рисунок 2.2 — Структура проекту

Призначення спроектованих класів наведено на рис. 2.3.




 GraphScreen	Prints information about window with pseudographics
 Screen	Shows information about window
 Window	Storing information about window

Рисунок 2.3 — Призначення класів, створене за допомогою Javadoc

2.3 Фрагменти програми

2.3.1 Файл Window.h

```
private:
    int iId; ///< Window identification number
    ...

/**
 * Returns Window::iId.
 * @return iId
 */
const int getId() const;
    ...

/**
 * Sets id value.
 */
void setId();

/**
 * Counts width of window.
 * @return width
 */
const int width() const;
    ...

/**
 * Counts height of window.
 * @return height
```

```
*/  
const int height() const;
```

2.3.2 Файл Window.cpp

```
const int Window::getId() const{  
    return ild;  
}  
  
const int Window::width() const {  
    return iX2-iX1;  
}  
  
const int Window::height() const {  
    return iY2-iY1;  
}
```

2.3.3 GraphScreen.h

```
/**  
 * @file GraphScreen.h  
 * Declaration of class GraphScreen  
 * @author Яценко Олександр  
 * @version 1.0.0  
 * @date 2017.10.01  
 */  
  
#ifndef GRAPHSCREEN_H_  
#define GRAPHSCREEN_H_  
  
#include "Window.h"  
/**  
 * Prints information about window with pseudographics.  
 */  
class GraphScreen {  
public:  
    /**  
     * Constructor.  
     * @param window source object  
     */  
    GraphScreen(Window* window);  
  
    /**  
     * Empty destructor.  
     */  
    ~GraphScreen();  
  
    const Window* iWindow; ///< Information about window  
  
    /**  
     * Sets data source object.  
     * @param iWindow source object  
     */  
    void setDataSource(const Window* window);  
  
    /**  
     * Prints information about window using pseudographics.  
     * @param window  
     */  
    void printData(const Window* window);
```

```

    /**
     * Shows data from Screen::iWindow
     */
    void showInfo();
};

#endif /* GRAPHSCREEN_H_ */

```

2.3.3 GraphScreen.cpp

```

/**
 * @file GraphScreen.cpp
 * Implementantion of class GraphScreen
 * @author Яценко Олександр
 * @version 1.0.0
 * @date 2017.10.01
 */

#include "Headers/GraphScreen.h"
#include <iostream>

using namespace std;

GraphScreen::GraphScreen(Window* window) :
    iWindow(window) {
}

GraphScreen::~GraphScreen() {
}

void GraphScreen::setDataSource(const Window* window) {
    iWindow = window;
}

void GraphScreen::printData(const Window* window) {
    cout << "=====" << endl;
    cout << "* ID=" << window->getId() << endl;
    cout << "* X1=" << window->iX1 << endl;
    cout << "* Y1=" << window->iY1 << endl;
    cout << "* X2=" << window->iX2 << endl;
    cout << "* Y2=" << window->iY2 << endl;
    cout << "* Width=" << window->width() << endl;
    cout << "* Height=" << window->height() << endl;
    cout << "=====" << endl;
}

void GraphScreen::showInfo() {
    cout << "=====" << endl;
    cout << "* ID=" << iWindow->getId() << endl;
    cout << "* X1=" << iWindow->iX1 << endl;
    cout << "* Y1=" << iWindow->iY1 << endl;
    cout << "* X2=" << iWindow->iX2 << endl;
    cout << "* Y2=" << iWindow->iY2 << endl;
    cout << "* Width=" << iWindow->width() << endl;
    cout << "* Height=" << iWindow->height() << endl;
    cout << "=====" << endl;
}

```

2.3.4 Файл Main.cpp

```

/**
 * @file main.cpp
 * Implementation of main() function
 * @author Яценко Олександр
 * @version 0.0.1
 * @date 2017.09.15
 */

#include "Headers/Screen.h"
#include "Headers/GraphScreen.h"

/**
 * Entry point.
 * @param argc number of command line parameters
 * @param argv array of command line parameters
 * @return exit code
 */
int main(int argc, char** argv) {
    Window window(1, 10, 20, 30, 40);
    Window *window2 = new Window(2, 15, 25, 35, 45);
    Window *window3 = new Window(3, 20, 30, 40, 50);

    Screen view1(window);
    GraphScreen view2(window2);

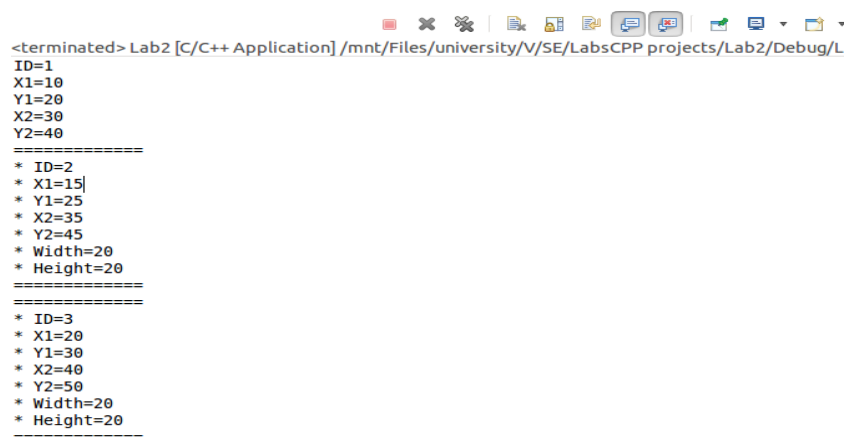
    view1.showInfo();

    view2.showInfo();
    view2.printData(window3);
    view2.setDataSource(window3);
    view2.showInfo();

    return 0;
}

```

3 РЕЗУЛЬТАТИ РОБОТИ



```

<terminated> Lab2 [C/C++ Application] /mnt/Files/university/V/SE/LabsCPP projects/Lab2/Debug/L
ID=1
X1=10
Y1=20
X2=30
Y2=40
=====
* ID=2
* X1=15
* Y1=25
* X2=35
* Y2=45
* Width=20
* Height=20
=====
* ID=3
* X1=20
* Y1=30
* X2=40
* Y2=50
* Width=20
* Height=20
=====

```

Рисунок 3.1 - Приклад роботи програми

ВИСНОВКИ

В розробленій програмі було створено додатково клас GraphScreen, котрий слугує для більш розширеного відображення інформації, також модифікував Window.