СПАДКУВАННЯ ТА ВІРТУАЛЬНІСТЬ

Лабораторна робота №3

Мета:

- Отримати основні навички розробки власних класів із використанням принципу розширення та віртуальності.

1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Створити клас Button використовуючи спадкування від Window із додавання нових полів. Роподілити та обгрунтувати права доступу до полів.

Виділити клас BaseView для Screen та GraphScreen із функцією BaseView::showInfo() та наступними віртуальними функціями, котрі викликаються з неї:

protected:
    BaseView::showHeader();
    BaseView::showContent();
    BaseView::showFooter();

Перенести основний функціонал відображення в базовий клас, реалізувавши специфічну поведінку у відповідних віртуальних методах. Після вказаних модифікацій Screen та GraphScreen повинні працювати аналогічно як у роботі 2 із обєктами класу Window та нащадками.

Створити клас ButtonScreen, котрий задає специфіку відображення для обєктів Button. Вибрати необхідне місце у ієрархії відображень для цього класу.

Показати роботу віртуальності на прикладі використання нащадка через показчик на базовий клас для обєкту Window та обєкту Button із використанням View-класів.

2 РОЗРОБКА ПРОГРАМИ

2.1 Ієрархія та структура класів

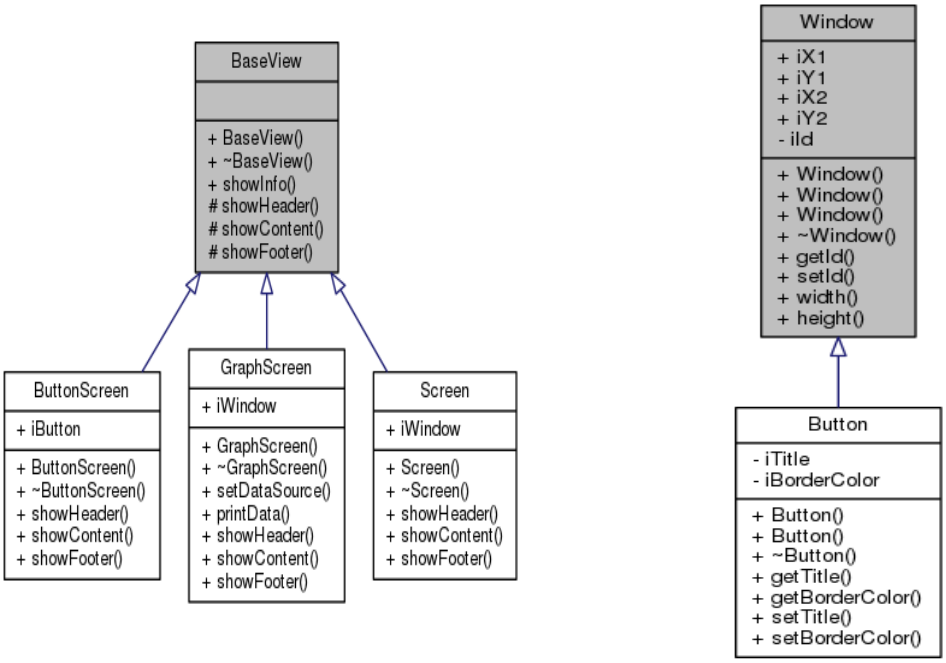


Рисунок 2.1 — Ієрархія класів

2.2 Опис програми

На рис.2.2 наведена структура розробленого проекту

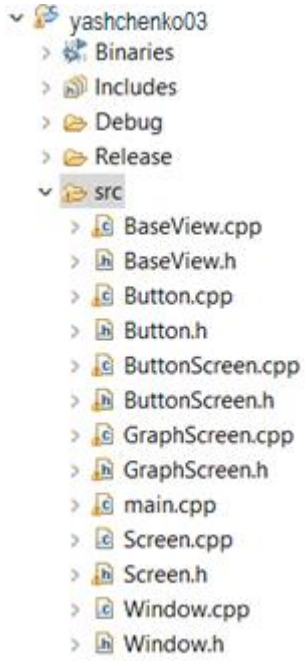
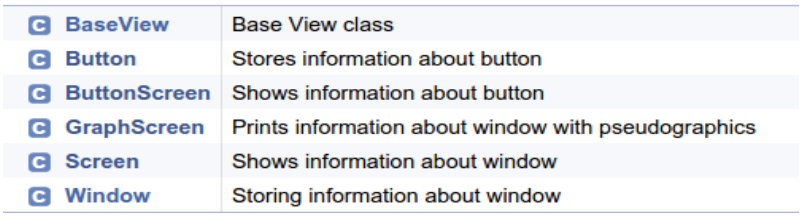


Рисунок 2.2 — Структура проекту

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
Window	Storing information about window

Рисунок 2.3 — Призначення класів, створене за допомогою Javadoc

2.3 Фрагменти програми

2.3.1 BaseView.h

```
/**
 * @file BaseView.h
 * Declaration of class BaseView
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.20
 */

#ifndef BASEVIEW_H_
#define BASEVIEW_H_
/**
 * Base View class.
 */
class BaseView {
public:
	/**
	 * Constructor.
	 */
	BaseView();
	/**
	 * Destructor.
	 */
	virtual ~BaseView();
	/**
	 * Shows full information about object.
	 */
	void showInfo();
protected:
	/**
	 * Shows header for information.
	 */
	virtual void showHeader();
	/**
	 * Shows main content about object.
	 */
	virtual void showContent();
	/**
	 * Shows footer of information.
	 */
	virtual void showFooter();
};

#endif /* BASEVIEW_H_ */
```

2.3.2 BaseView.cpp

```
/**
 * @file BaseView.cpp
 * Implementation of class BaseView
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.20
 */

#include "Headers/BaseView.h"

BaseView::BaseView() {

}

BaseView::~BaseView() {
}

void BaseView::showInfo() {
	showHeader();
	showContent();
	showFooter();
}

void BaseView::showHeader() {
}

void BaseView::showContent() {
}

void BaseView::showFooter() {
}
```

2.3.3 Button.h

```
/**
 * @file Button.h
 * Declaration of class Button.
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.20
 */

#ifndef BUTTON_H_
#define BUTTON_H_

#include "Window.h"

/**
 * Stores information about button.
 */
class Button: public Window {
public:
	/**
	 * Constructor.
	 */
	Button();

	/**
	 * Constructor with parameters.
	 * @param aId sets Button::iId
	 * @param aX1 sets Button::iX1
	 * @param aY1 sets Button::iY1
	 * @param aX2 sets Button::iX2
```

```
         * @param aY2 sets Button::iY2
         * @param aTitle sets Button::iTitle
         * @param aBorderColor sets Button::iBorderColor
         */
        Button(int aId, int aX1, int aY1, int aX2, int aY2, char * aTitle,
                                char * aBorderColor);

        /**
         * Destructor.
         */
        ~Button();
private:
        char iTitle[32]; ///< Text of button title
        char iBorderColor[32]; ///< Color of button border
public:
        /**
         * Returns Button::iTitle.
         * @return pointer to text of button title
         */
        const char* getTitle() const;

        /**
         * Returns Button::iBorderColor.
         * @return pointer to text of color of border of button
         */
        const char* getBorderColor() const;

        /**
         * Sets Button::iTitle.
         */
        void setTitle(char * aTitle);

        /**
         * Sets Button::iBorderColor.
         */
        void setBorderColor(char * aBorderColor);
};

#endif /* BUTTON_H_ */
```

2.3.4 Button.cpp

```
/**
 * @file Button.cpp
 * Implementation of class Button.
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.20
 */

#include "Headers/Button.h"
#include "Headers/Window.h"
#include "string.h"

Button::Button() {
        strcpy(iTitle, "Default");
        strcpy(iBorderColor, "Default");
}

Button::Button(int aId, int aX1, int aY1, int aX2, int aY2, char * aTitle,
                        char * aBorderColor) :
                        Window(aId, aX1, aY1, aX2, aY2) {
        strcpy(iTitle, aTitle);
```

```
	strcpy(iBorderColor, aBorderColor);
}

Button::~Button() {

}

const char* Button::getTitle() const {
	return iTitle;
}

const char* Button::getBorderColor() const {
	return iBorderColor;
}

void Button::setTitle(char * aTitle) {
	strcpy(iTitle, aTitle);
}

void Button::setBorderColor(char * aBorderColor) {
	strcpy(iBorderColor, aBorderColor);
}
```

2.3.5 ButtonScreen.h

```
/**
 * @file ButtonScreen.h
 * Decalaration of class ButtonScreen
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.20
 */

#ifndef BUTTONSCREEN_H_
#define BUTTONSCREEN_H_

#include "BaseView.h"
#include "Button.h"

/**
 * Shows information about button.
 */
class ButtonScreen: public BaseView {
public:
	ButtonScreen(Button button);
	~ButtonScreen();

	Button iButton; ///< Information about button

	/**
	 * Shows header for information.
	 */
	final void showHeader() ovveride;

	/**
	 * Shows main contentof information.
	 */
	final void showContent() ovveride;

	/**
	 * Shows footer of information.
	 */
	final void showFooter() ovveride;
};
```

```
#endif /* BUTTONSCREEN_H_ */
```

2.3.6 ButtonScreen.cpp

```
/**
 * @file ButtonScreen.cpp
 * Implementation of class ButtonScreen.
 * @author Ященко Олександр
 * @version 0.0.1
 * @date 2017.10.20
 */

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

using namespace std;

ButtonScreen::ButtonScreen(Button button) {
	iButton = button;
}

ButtonScreen::~ButtonScreen() {
}

void ButtonScreen::showHeader() {
	cout << "Button info:" << endl;
}

void ButtonScreen::showContent() {
	cout << "ID=" << iButton.getId() << endl;
	cout << "X1=" << iButton.iX1 << endl;
	cout << "Y1=" << iButton.iY1 << endl;
	cout << "X2=" << iButton.iX2 << endl;
	cout << "Y2=" << iButton.iY2 << endl;
	cout << "Width=" << iButton.width() << endl;
	cout << "Height=" << iButton.height() << endl;
	cout << "Button text=" << iButton.getTitle() << endl;
	cout << "Border Color=" << iButton.getBorderColor() << endl;
}

void ButtonScreen::showFooter() {
	cout << "ButtonScreen output..." << endl;
}
```

2.3.7 Screen.h

```
…
/**
 * Shows header for information.
 */
final void showHeader() ovveride;
/**
 * Shows main contentof information.
 */
final void showContent() ovveride;
/**
 * Shows footer of information.
 */
final void showFooter() ovveride;
…
```

2.3.8 Screen.cpp

```
…
void Screen::showHeader() {
        cout << "Window info:" << endl;
}

void Screen::showContent() {
        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;
}

void Screen::showFooter() {
        cout << "Screen output..." << endl;
}
…
```

2.3.9 GraphScreen.h

```
…
/**
 * Shows header for information.
 */
final void showHeader() ovveride;
/**
 * Shows main contentof information.
 */
final void showContent() ovveride;
/**
 * Shows footer of information.
 */
final void showFooter() ovveride;
…
```

2.3.10 GraphScreen.cpp

```
…
void GraphScreen::showHeader() {
        cout << "Window info:" << endl;
}
void GraphScreen::showContent() {
        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;
}

void GraphScreen::showFooter() {
        cout << "GraphScreen output..." << endl;
}
…
```

2.3.11 main.cpp

```
…
int main(int argc, char** argv) {
	Window window(1, 10, 20, 30, 40);
	Window *window2 = new Window(2, 15, 25, 35, 45);
	Button button(3, 30, 50, 80, 90, "Nice button", "red");

	BaseView * view1 = new Screen(window);
	BaseView * view2 = new GraphScreen(window2);
	BaseView * view3 = new ButtonScreen(button);

	view1->showInfo();
	cout << endl;
	view2->showInfo();
	cout << endl;
	view3->showInfo();
	return 0;
}
```

3 РЕЗУЛЬТАТИ РОБОТИ

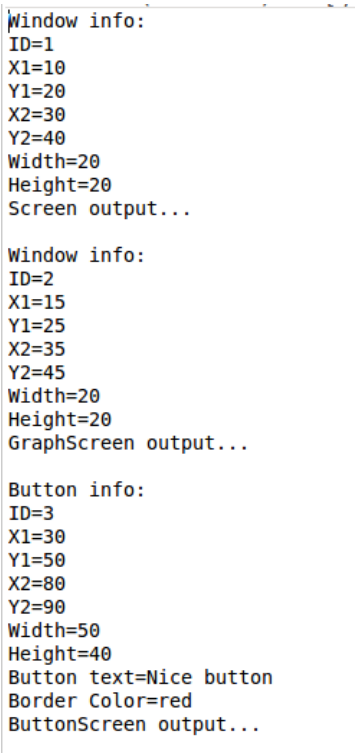


Рисунок 3.1 - Приклад роботи програми

ВИСНОВКИ

В розробленій програмі було реалізовано базовий клас відображень BaseView, клас реалізуючий сутність “кнопка” - Button та клас, що задає специфіку його відображення ButtonScreen. Були внесені зміни до інших класів відображення у звязку із використанням віртуальності.