# Тема 4: Статичні методи, перевантаження операторів та методів.

## Мета

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

## 1. Загальне завдання

Необхідно визначити у класі *View1* статичний метод *OnTimerAction()*. Цей метод відображатиме на екрані заданий нащадок *Data1*. Обрати для Win32-таймера власний інтервал повторних викликів. Встановити реалізований метод *View1::OnTimerAction()* на виклик у таймері. Таймер повинен спрацювати лише 4 рази. Метод повинен виводити на екран дані про поточний асоційований об'єкт даних. Додаткові 4 бали додаються пр иокремій реалізації класу для роботи із таймером. Реалізувати перевантажені опреатори та методи згідно індивідуального завдання.
Варіант 3.

1. Предметна область: Мишки.
2. Перевантаженні методи SetData();
3. Перевантажені оператори:

- bool Data1::operator == (const Data1&), котрий повертає true, якщо тип пристрою та кількість кнопок співпадає.
- bool Data2::operator == (const Data2&), котрий повертає true, якщо усі параметри порівнюваних об'єктів співпадають.
- Data2::operator = (int) встановлює тип сенсора, який допустимий для цього класу. У разі недопустимого значення присвоєння не відбувається і залишається попереднє значення.

## 2. Розробка програми

### 2.1. Засоби ООП

У розробленій програмі були використані наступні засоби ООП:

- інкапсуляція
- спадкування
- поліморфізм

### 2.2. Ієрархія та структура класів

На рис. 2.1 наведена ієрархія зв'язків класу **BaseView**, на рис. 2.2 - **Manipulator**, а на рисунку 2.3 - **Event**.

**Hierarchy of BaseView class**

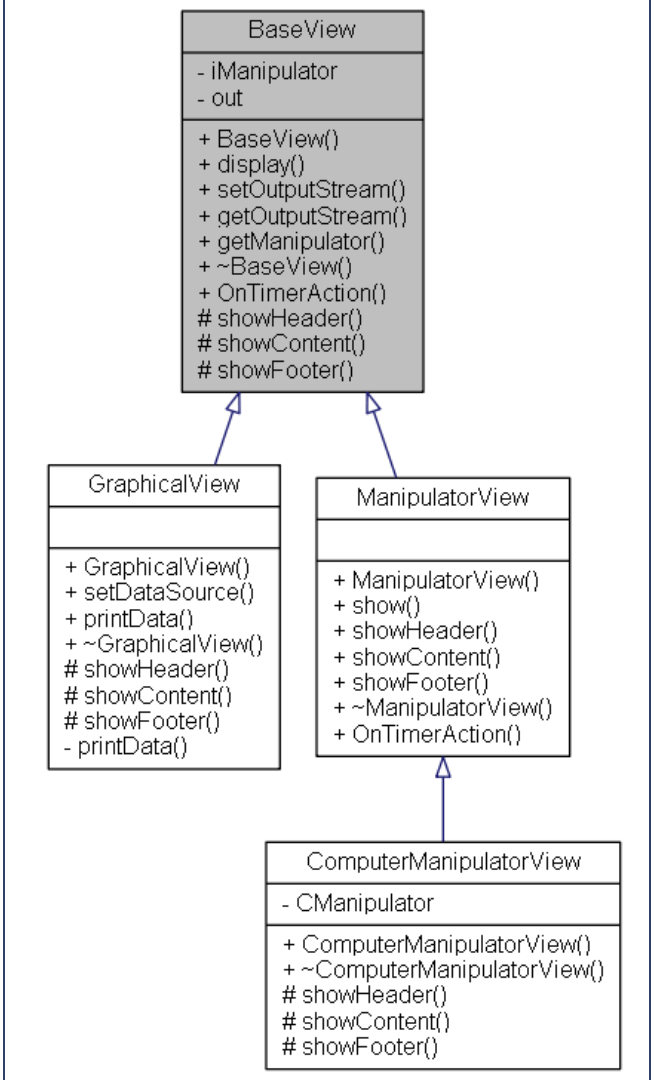


Рисунок 2.1 - Ієрархія зв'язків класу **BaseView**

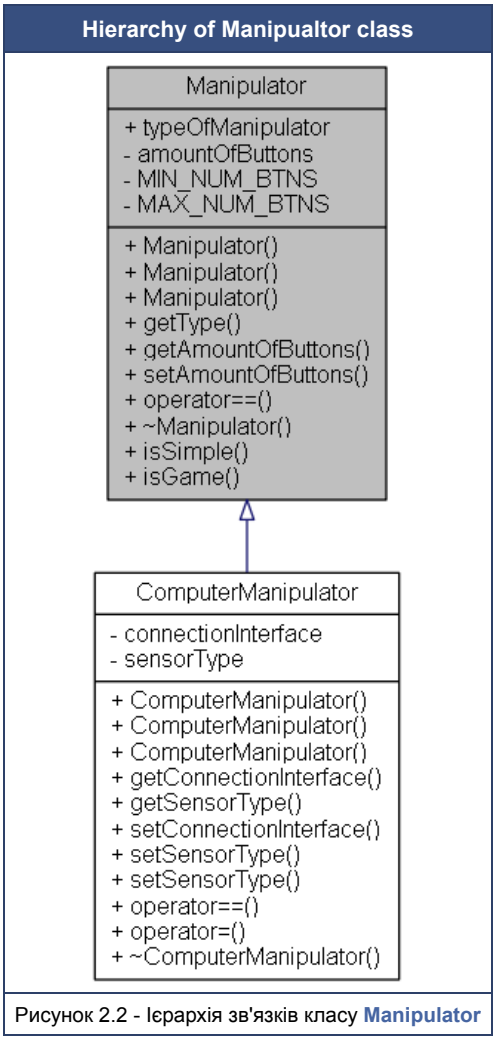


Рисунок 2.2 - Ієрархія зв'язків класу **Manipulator**

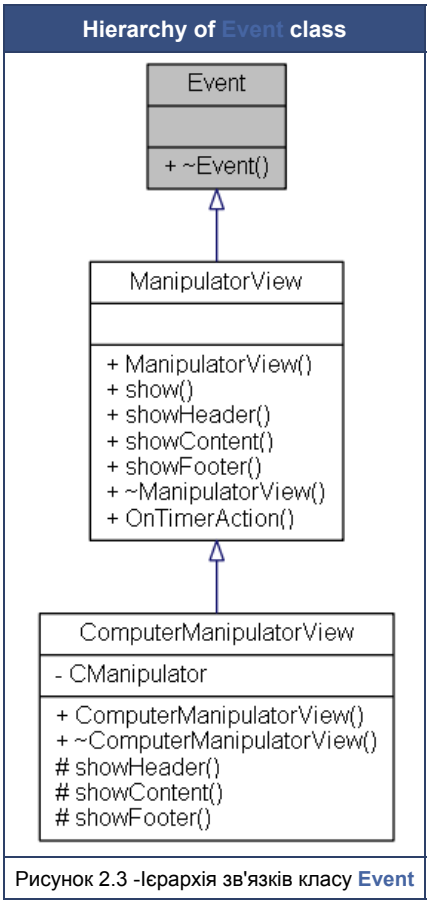


Рисунок 2.3 -Ієрархія зв'язків класу **Event**

### 2.3. Опис програми

На рисунку 2.4 наведена структура розробленого проекту:

**Project structure**

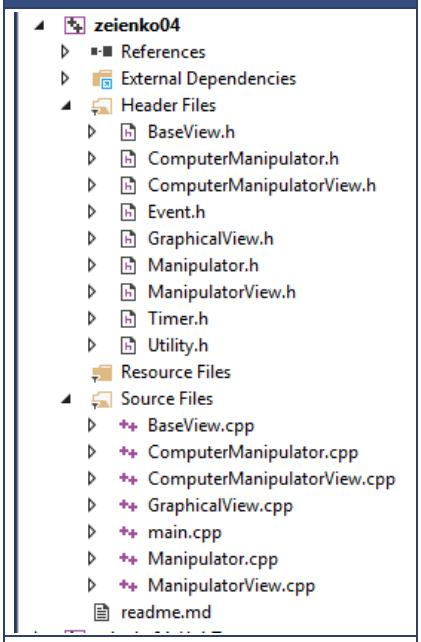


Рисунок 2.4 - Структура проект

### 2.4. Важливі фрагменти програми

Демонстрація перевантажених операторів та спрацьовування таймера наведена на рисунку 2.5 та рисунку 2.6 відповідно.

**Демонстрація перевантажених операторів**

```
 // Operator overloading demo
void operatorOverloadingDemonstration() {
    Manipulator* manipulator = new Manipulator;
    ComputerManipulator* computerManipulator = new ComputerManipulator;
    printf("Compare manipulators by two parameters: amount of buttons and its type.\n");
    printf("Testing 2 objects on equality: ");
    if (*manipulator == *computerManipulator) {
        printf("manipualtors are equal. \n");
    }
    else {
        printf("manipualtors are different. \n");
    }
    printf("Try to assign number to ComputerManipulator's object and change its \n sensor type to Ballng.\n");
    printf("Sensor type before assigning: %s\n", computerManipulator->getSensorType());
    *computerManipulator = 0;
    printf("Sensor type after assigning: %s\n", computerManipulator->getSensorType());
    delete manipulator;
    delete computerManipulator;
}
```

Рисунок 2.5. – Функція operatorOverloadingDemonstration()

**Демонстрація спрацьовування таймера**

```
void timerDemonstration() {

    ComputerManipulator* compManip = new ComputerManipulator();
    ComputerManipulatorView* compManipView = new ComputerManipulatorView(*compManip, &(std::cout));

    printf("Creating timer object \n");
    Timer<ManipulatorView>* timer = new Timer<ManipulatorView>(compManipView, &ComputerManipulatorView::OnTimerAction);
    printf("Setting is_running to true \n");
    timer->setIsRunning(true);
    printf("Starting the timer \n");
    timer->startTheThread();

    timer->Join();
    printf("Done\n ");
    delete compManip;
    delete compManipView;
    delete timer;
}
```

Рисунок 2.6. – Функція timerDemonstration()

Для даного проекту були розроблені модульні тести за допомогою GoogleTest Framework. Ці тести перевіряють працювання перевантажених операторів. Результат роботи тестів зображений на рисунку 3.2 п. Результат роботи

Призначення спроектованих класів наведено на рис. 2.7.

**Predestination structure**

## Класи

Класи, структури, об'єднання та інтерфейси з коротким описом.

| | |
|---|---|
| C BaseView | The base class for all other classes which are responsible for output |
| C ComputerManipulator | This class represents computer`s manipulators |
| C ComputerManipulatorView | This class provides output of the CpmputerMnipulator`s information |
| C Event | This class is used to work with timer those classes which will inherit this one |
| C GraphicalView | Representation of text-pseudo-graphic display of data |
| C Manipulator | Represents abstraction of mouse periphery |
| C ManipulatorView | This class provides an output information about manipulator |
| C Timer | This class is the representation of the timer |

Рисунок 2.7. – Призначення спроектованих класів

# 3. Результат работи

Результат роботи програми зображений на рисунку 3.1.

| Output to the console |
|---|
| img_result |

Рисунок 3.1. – Результат виконання програми

Результат виконання всіх модульних тестів зображений на рисунку 3.2. Тести відсортировані за класом тестування.

**Unit Tests Result**

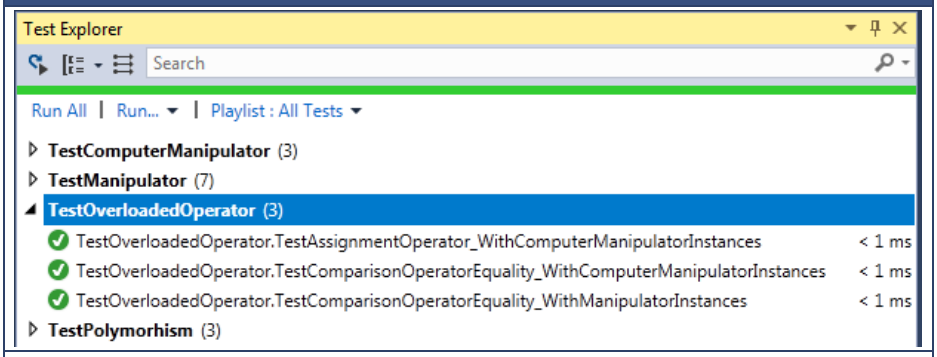


Рисунок 3.2. – Результат тестування

Було усунуто витоки пам'яті. Код завершення програми зображений на рисунку 3.3

**Program completion code**

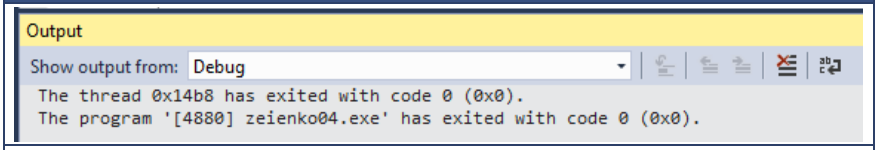


Рисунок 3.3. – Результат тестування

## Висновок

В ході виконання лабораторної роботи були отримані навички використовування статичних методи, а також використовування перевантаженних методів та операторів.