Абстрактні класи, інтерфейси,серіалізація

Мета

Навчитись застосовувати інтерфейси для роботи класів на прикладі задачі серіалізації.

1. Індивідуальне завдання

Реалізувати для кожного із класів даних своєї ієрархії можливість збереження та завантаження даних за допомогою класу CFileStorage, який видається до лабораторної роботи у вигляді бібліотеки. Показати у звіті бінарний дамп збереженого файлу та відмітити дані із власних об’єктів.

2. Розробка програми

2.1 Засоби ООП

В ході розробки програми були використані так засоби ООП:

- Абсракція – кожен об’єкт описує свою сутність, яка визначається його полями.
- Спадкування - механізм утворення нових класів на основі використання вже існуючих
- Інкапсуляція - поля об'єктів закриті для користувача,натомість ми даємо доступ до даних за допомогою геттерів та сеттерів, так користувач має можливість отримати готові дані, а не обробляти їх, для подпльшого вікористання.
- Поліморфізм - властивість, яка дозволяє одне і те саме ім’я використовувати для вирішення декількох технічно різних задач, тобто основною метою поліморфізму є використання одного імені для задання загальних класу дій.

2.2 Ієрархія та структура класів

На рис 2.1 дивись ієрархію класів.

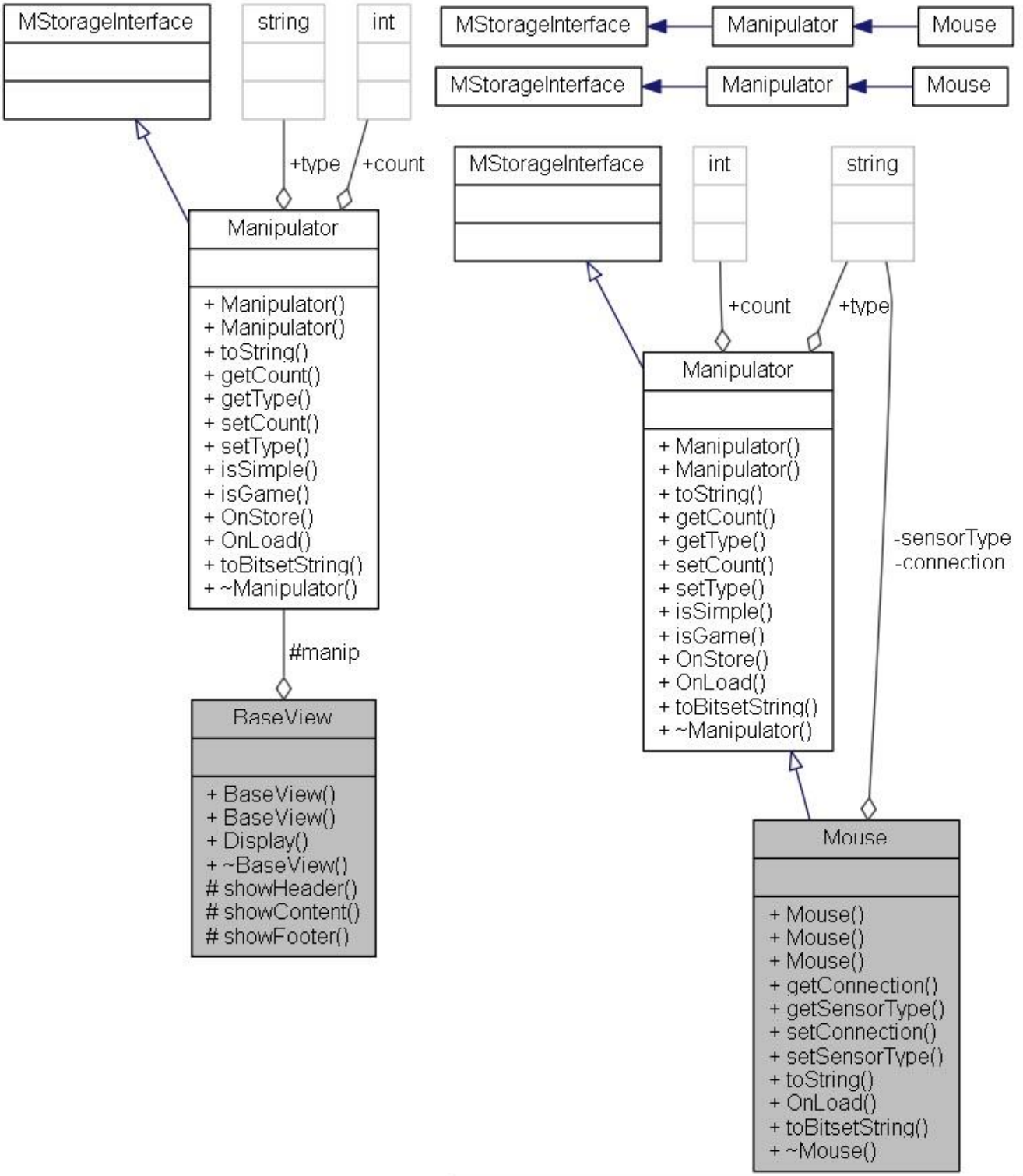

Рисунок 2.1 – Ієрархія класів

Опис програми

На рис. 2.2 дивись структуру проекту.

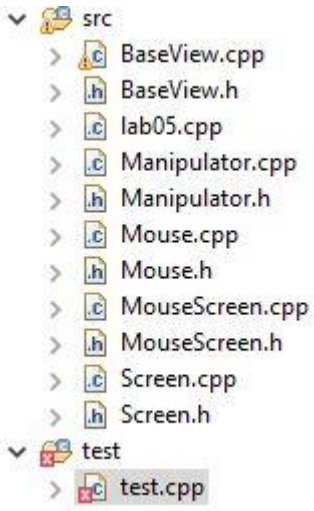

Рисунок 2.2 – Структура проекту

На рис. 2.3 дивись призначення класів.

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

BaseView	Базовий клас для Manipulator & Mouse
Manipulator	Клас опису маніпулятору
Mouse	Клас опису мишки
MouseScreen	Клас для відображення данних мишки
MStorageInterface	
Screen	Класс для відображення даних

Рисунок 2.3 – Призначення класів

Класи Manipulator та Mouse мають методи для збереження(OnStore) та завантаження данних(OnLoad). Та метод для переводу данних класу у послідовність бітів(toBitsetString).

2.4 Важливі фрагменті програми

У программі слід зауважити увагу на таких моментах:

Клас `Manipulator.cpp`:

```
void Manipulator::OnStore(std::ostream& aStream) {
        aStream << toBitsetString();
}

void Manipulator::OnLoad(std::istream& aStream) {

        bitset<32> input;
        aStream >> input;
        count = input.to_ulong();

        bitset<8> inputS;
```

```
        string tmpStr;
        while (aStream.get() != ' ') {
                        aStream >> inputS;
                        tmpStr += (char) inputS.to_ulong();
                }
                this->setType(tmpStr);

}

string Manipulator::toBitsetString() {

        string res;

        res += bitset<32>(this->getCount()).to_string();

        for (unsigned int i = 0; i < this->getType().length(); i++) {
                res += bitset<9>(this->getType().at(i)).to_string();
        }
                res += " ";
                return res;
}
#### Mouse.cpp:
void Mouse::OnLoad(std::istream& aStream) {

        Manipulator::OnLoad(aStream);
        string tmpStr;
        bitset<8> input;

        while (aStream.get() != ' ') {

                aStream >> input;
                tmpStr += (char) input.to_ulong();

        }
        this->setConnection(tmpStr);

        tmpStr.clear();
        while (aStream.get() != ' ') {
                aStream >> input;
                tmpStr += (char) input.to_ulong();
        }
        this->setSensorType(tmpStr);

}

string Mouse::toBitsetString() {

        string res = Manipulator::toBitsetString();

        for (unsigned int i = 0; i < this->getConnection().length(); i++) {
                res += bitset<9>(this->getConnection().at(i)).to_string();
        }
        res += " ";

        for (unsigned int i = 0; i < this->getSensorType().length(); i++) {
                res += bitset<9>(this->getSensorType().at(i)).to_string();
        }
        res += " ";
        return res;
}
```

3. Результати роботи

Результати роботи показано на рис.3.1

```
<terminated> (exit value: 0) lab05.exe [C/C++ Application] C:\workspace\lab05\Debug\lab05.exe (06.11.17, 15:20)
Количество кнопок: 5
Тип устройства : Joystick
Количество кнопок: 4
Тип устройства : Mouse
Тип соединения: Wireless
Тип сенсора: Laser
```

Рисунок 3.1 – Результати роботи

Бітова послідовність збережених файлів показано на рис. 3.2 - 3.3

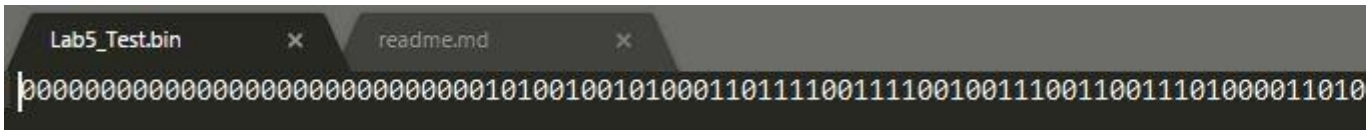

Рисунок 3.2 – Бітова послідовність Manipulator

Рисунок 3.3 – Бітова послідовність Mouse

Висновки

Отримав основні навики перевантаження операторів, методів та ознайомився зі статичними методами