Министерство образования Республики Беларусь
Учреждение образования
“Брестский Государственный технический университет”
Кафедра ИИТ

Лабораторная работа №5

По дисциплине “Проектирование программного обеспечения интеллектуальных систем”

Тема: “Иерархии классов. Наследование”

Выполнил:
Студент 2 курса
Группы ИИ-21
Кирилович А. А.
Проверил:
Монтик Н. С.

Брест 2022

Цель работы:

1. Изучение правил наследования классов.
2. Реализация одиночного наследования классов.
3. Изучение управления методами и свойствами производных классов через объекты производных классов и через указатели на объекты производных классов.
4. Изучение правил описания наследования и диаграмм классов в языке UML.

Ход работы: Разработать три пользовательских класса в соответствии с целью.

```
#include <iostream>

class USBDevice {
private:
    int m_id;
    int m_type;
public:
    USBDevice() {
        m_id = 0;
        m_type = 0;
    }
    USBDevice(int id, int type)
         : m_id(id), m_type(type) {}
    USBDevice(const USBDevice& other)
         : m_id(other.m_id), m_type(other.m_type) {}
    ~USBDevice() {}
    void SetId(int id) { m_id = id; }
    void SetType(int type) { m_type = type; }
    int GetId() { return m_id; }
    int GetType() { return m_type; }
    void Print() {
        std::cout << "USBDevice: id = " << m_id << ", type = " << m_type << std::endl;
    }
};

class USBHub {
private:
    int m_id;
    int m_type;
    int m_ports;
public:
    USBHub() {
        m_id = 0;
        m_type = 0;
        m_ports = 0;
    }
    USBHub(int id, int type, int ports)
         : m_id(id), m_type(type), m_ports(ports) {}
    USBHub(const USBHub& other)
         : m_id(other.m_id), m_type(other.m_type), m_ports(other.m_ports) {}
    ~USBHub() {}
    void SetId(int id) { m_id = id; }
    void SetType(int type) { m_type = type; }
    void SetPorts(int ports) { m_ports = ports; }
    int GetId() { return m_id; }
    int GetType() { return m_type; }
    int GetPorts() { return m_ports; }
    void Print() {
        std::cout << "USBHub: id = " << m_id << ", type = " << m_type << ", ports = " << m_ports << std::endl;
    }
};

class USBDeviceHub : public USBDevice, public USBHub {
public:
    USBDeviceHub() {
        USBHub::SetId(0);
        USBHub::SetType(0);
        USBHub::SetPorts(0);
    }
    USBDeviceHub(int id, int type, int ports)
         : USBDevice(id, type), USBHub(id, type, ports) {}
    USBDeviceHub(const USBDeviceHub& other)
         : USBDevice(other), USBHub(other) {}
    ~USBDeviceHub() {}
```

```
    void SetId(int id) { USBDevice::SetId(id); USBHub::SetId(id); }
    void SetType(int type) { USBDevice::SetType(type); USBHub::SetType(type); }
    void SetPorts(int ports) { USBHub::SetPorts(ports); }
    int GetId() { return USBDevice::GetId(); }
    int GetType() { return USBDevice::GetType(); }
    int GetPorts() { return USBHub::GetPorts(); }
    void Print() {
        std::cout << "USBDeviceHub: id = " << USBDevice::GetId() << ", type = " << USBDevice::GetType() << ",
ports = " << USBHub::GetPorts() << std::endl;
    }
};

int main() {
    USBDevice device(1, 2);
    USBHub hub(3, 4, 5);
    USBDeviceHub devicehub(6, 7, 8);
    device.Print();
    hub.Print();
    devicehub.Print();
}
```