EIT, 306387, Gabriel Ćwiek, MAPT 2, lab 3, https://github.com/Astaree/MAPT_2

1. Ex. 0:
   a. Created object calls chairman desc. As we overridden Student function.

```
#include <iostream>
#include <string>
using namespace std;

class Student {
public:
  string description = "A student of a group";
  Student();
  void printDescription();
};

Student::Student(){
    cout << "Creating student object class named: " << description << endl;
  }

class Chairman : private Student {
public:
  string description = "A chairman of a group";
  void printDescription();
};

void Student::printDescription() {
  cout << "Object description: " << description << endl;
}

void Chairman::printDescription(){
  cout << "Object description: " << description << endl;

}
int main() {
    Chairman ch;
    ch.printDescription();
}

Output:
Creating student object class named: A student of a group
Object description: A chairman of a group
```

2. Ex. 1:
    a. Added mth., edited members, cannot access "inform…" functions because they are protected and are not "friend" functions, but we can call it from constructor

```cpp
Chairman::Chairman(string sname_surname, unsigned int sid_number, string semail)
    :Student(sname_surname, sid_number), email(semail)
{
    cout << "Creating an object of the Chairman class named: "
         << description << endl;
}

Student::Student(string sname_surname, unsigned int sid_number)
    : name_surname(sname_surname)
{
    id_number = sid_number;
    cout << "Creating an object of the Student class named: "
         << description << endl;
    Coordinator::informGroup();
    Coordinator::informTeacher();
}

void Student::printDescription()
{
    cout << "Description: " << description << endl;
}

void Coordinator::informGroup(){
    cout<<"Informed group\n";
}
void Coordinator::informTeacher(){
    cout<<"Informed teacher\n";
}

int main()
{
    Student stud("Jan Kowalski", 7);
    stud.printDescription();
    Chairman chair("Aleksandra Nowak", 999, "mail@nomail.dot");
    chair.printDescription();

    // cant acces w/o making component "friend"
    //chair.informGroup();
    //chair.informTeacher();

}
```

3. Ex. 2:
    a. Changing just to virtual method doesn't change anything in base "call" but assigning it to 0 makes it impossible to compile as method if purely virtual

```cpp
#include <iostream>
#include <string>
using namespace std;

class Student
{
protected:
    string name_surname = "NO_NAME";
    unsigned int id_number = 0;

public:
    Student(string sname_surname, unsigned int sid_number);
    string description = "student of group";
    //virtual void printDescription();
    //Doesnt work -- method is purely virtual,
    virtual void printDescription() = 0;
    void printData()
    {
        cout << " Method printData of the Student class" << endl;
        cout << " name and surname " << name_surname << endl;
        cout << " id number " << id_number << endl;
    }
};

class Coordinator
{
    public:
    string description="coordinator of group";

    void printDescription();

    protected:
    void informGroup();
    void informTeacher();

};

class Chairman : public Student,Coordinator
{
private:
    string email = "no@noemail";
public:
    Chairman(string sname_surname, unsigned int sid_number, string semail);
    string description = "chairman of a group";
    void printDescription();
};


Chairman::Chairman(string sname_surname, unsigned int sid_number, string
semail)
    :Student(sname_surname, sid_number), email(semail)
{
    cout << "Creating an object of the Chairman class named: "
         << description << endl;
    Coordinator::informGroup();
    Coordinator::informTeacher();

}

Student::Student(string sname_surname, unsigned int sid_number)
    : name_surname(sname_surname)
{
    id_number = sid_number;
    cout << "Creating an object of the Student class named: "
         << description << endl;
}

void Student::printDescription()
{
    cout << "Description: " << description << endl;
}

void Coordinator::printDescription()
{
    cout << "Description: " << description << endl;
}
void Chairman::printDescription()
{
    cout << "Description: " << description << endl;
}
void Coordinator::informGroup(){
    cout<<"Informed group\n";
}
void Coordinator::informTeacher(){
    cout<<"Informed teacher\n";
}
```

```cpp
int main()
{
    Student stud("Jan Kowalski", 7);
    stud.printDescription();
    Chairman chair("Aleksandra Nowak", 999, "mail@nomail.dot");
    chair.printDescription();

    // cant acces w/o making component "friend"
    //chair.informGroup();
    //chair.informTeacher();

}
```

4. Ex. 4:
    a. Doesn’t compile as “Device u” is purely virtual,
    b. Destructor – I guess,

```
#include <iostream>
#include <sstream>
using namespace std;

class Device
{
public:
    Device(){};
    ~Device(){};
    virtual int write(int id, string data) = 0;
    virtual string read(int id) = 0;
};

class Disc : public Device
{
private:
    int id_;
    string data_;

public:
    Disc(int id);
    int write(int id, string data);
    string read(int id);
};

Disc::Disc(int id)
{

    cout << "Creating an object of the Disc class   " << endl;
}

int Disc::write(int id, string data)
{
    if (id >= 0)
    {
        id_ = id;
        data_ = data;
        cout << "writing data: " << data << endl;
        return 1;
    }
    cout << "-1\n";
    return -1;
}
```

```
string Disc::read(int id)
{
    cout << "reading data: " << data_ << endl;
    return data_;
}

int main()
{
    //all members of class are virtual
    //Device u;
    Disc d1(0);
    d1.write(7, "test 11");
    d1.read(7);
    system("pause");
}

Output for input „0”:
-1
writing data: test 11
reading data: test 11

Output for input “7”:
Creating an object of the Disc class
1
writing data: test 11
reading data: test 11
```