# Object Oriented programming and software engineering – Lab 8

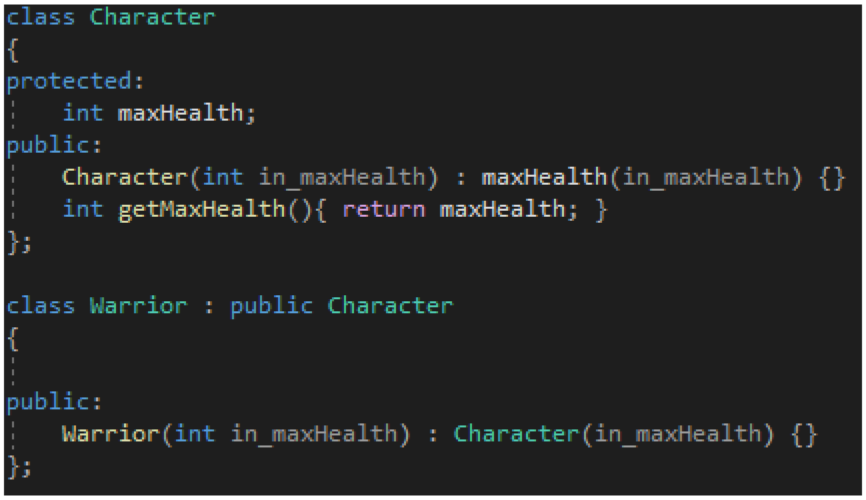
Adam Korytowski – 2025

## 1. Inheritance

Inheritance is one of the fundamental concepts of Object-Oriented Programming (OOP) in C++. It allows one class (child/derived class) to inherit properties and behavior from another class (parent/base class). This promotes code reuse and hierarchical relationships between classes.

| **Base Class Access** | **Public Inheritance** | **Protected Inheritance** | **Private Inheritance** |
|---|---|---|---|
| ***public* members** | *public* in derived | *protected* in derived | *private* in derived |
| ***protected* members** | *protected* in derived | *protected* in derived | *private* in derived |
| ***private* members** | Not inherited | Not inherited | Not inherited |

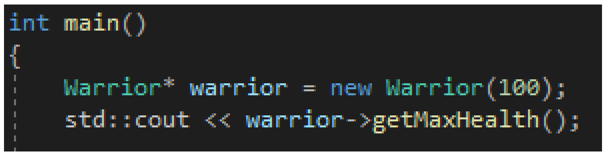
Example:



```cpp
class Character
{
protected:
    int maxHealth;
public:
    Character(int in_maxHealth) : maxHealth(in_maxHealth) {}
    int getMaxHealth(){ return maxHealth; }
};

class Warrior : public Character
{

public:
    Warrior(int in_maxHealth) : Character(in_maxHealth) {}
};
```



```cpp
int main()
{
    Warrior* warrior = new Warrior(100);
    std::cout << warrior->getMaxHealth();
```

## 2. Header and .cpp files

In C++, .h (header) files and .cpp (source) files are used to organize code, improve maintainability, and enable modular programming. Here's why they are used:

- Header files (.h): Contain declarations (function prototypes, class definitions, constants, macros).
- Source files (.cpp): Contain implementations (actual function definitions and logic).
- Header files allow you to reuse code by including them in multiple .cpp files without rewriting functions or classes.
- Instead of recompiling everything, only the changed source files (.cpp) are recompiled, improving compilation speed.
- Splitting code into multiple files makes it easier to manage large projects and work collaboratively.

## 3. Complete the tasks, having code in separate .h and .cpp files. Header files – class declarations, .cpp files – methods' definitions (one .h and one .cpp file per class) (2 pts each)

- Create 2 child classes for each of 2 existing classes. Create member variables for each deriving class.
- Add methods to each of the derived classes that interact with the base class's attributes. This method should alter or use those attributes in meaningful way.
- Create a new class that uses the existing classes, such as having instances of existing classes as member variables. Implement a method that interacts with the created objects.
- Modify constructors of two of the derived classes to accept additional parameters and use them to initialize both the base class and the derived class's attributes.