Seminar 05
Static members and methods, =default, =delete

1. = default and = delete.

- = default marks a constructor or operator= to be defined by the compiler using the **default implementation** (shallow copy).
  Example:
```
ClassName() = default;
```

- = delete marks a constructor or operator= as **non existent** for this class.
  I.e. this class will not have the marked constructor/operator=.
  Example:
```
// This prohibits the class from being copied
ClassName(const ClassName& other) = delete;
ClassName& operator=(const ClassName& other) = delete;
```

2. Static members and methods.

- Members and methods marked as static are linked to the **class** and not to an object of that class. And thus are accessed by specifying the class first, followed by :: and then the name of the member/method. *(ClassName::member)*
- We can think of static members as global variables for the class.
  Example:

Person.h

```
class Person
{
public:
    static int publicMember;
    static const int MAX_SOMETHING;
    Person();
    static int getNumOfPeople() { return Person::numOfPeople; }
private:
    static int numOfPeople;    // Used as people counter
};
```

Person.cpp

```
#include "Person.h"

int Person::publicMember = 42; // Default values for the data members
int Person::numOfPeople  = 0;  // are defined in the source file

Person::Person()
{
    numOfPeople++;
}
```

Source.cpp

```cpp
#include "Person.h"
int main()
{
    Person p1;
    std::cout << Person::getNumOfPeople();  // 1
    Person p2;
    std::cout << Person::getNumOfPeople();  // 2
    Person p3;
    Person p4;
    Person p5;
    std::cout << Person::getNumOfPeople();  // 5

    // This static member cannot be accessed since it's private
    ~~std::cout << Person::numOfPeople;~~

    // But this statc member is public, so it can be accessed
    std::cout << Person::publicMember;

    // As well as changed
    Person::publicMember = 5;

    return 0;
}
```