Seminar 04

Classes. Declaration & definition. Members. Access specifiers.

- Classes. Why do we need them?
    - Abstraction, reusability, single encapsulated objects with interface.

- Methods and **this** pointer.
    - Methods are **functions** inside of a class' declaration. They all have access to the **this** pointer.
    - **this** is a **const pointer** referring to the object that's called the method.

- Constructors
    - Methods called when an object of a specific class is being created.
    - Constructors don't have a return type.
    - Default, parameterized and *copy* constructors.

      [More on copy constructors and destructors in the next lesson.]

- Access specifiers.
    - **public:**
      Everything after this modifier is visible by the "outside world".
    - ***protected:*** *[More on this specifier when we learn about **inheritance**.]*
      Everything after this modifier is visible by the children of the class.
    - **private:**
      Everything after this modifier is NOT visible by the "outside world".

- Differences with structs.
    - In **C++** almost none.
      Structs have public access specifier by default
      Classes - private by default.
    - In **C** structs **can't** have methods, static members, access specifiers and more.

- Selectors and Mutators (getters and setters).
    - Each **selector** is a method that returns a value of a data member.

      ```
      <member_type> get<Member_name>()
      {
          return <member_name>;
      }
      ```

      *For data members that are **arrays** the return type of the **selector** must be a **const** <type>*****, thus we don't break the encapsulation of the class.*
      We'll talk about a better way to return arrays later in the course.

    - Each **mutator** is a method that **modifies** a data member ONLY in the way we intend to. (i.e. we must validate the given parameter).

      ```
      void set<Member_name>(<member_type> value)
      {
          if (<validate the given value>) {
              <member_name> = value;
      ```

```
            }
        }
```

<u>Examples:</u>

Rectangle.h

```
#pragma once

class Rectangle
{
public:
    Rectangle();
    Rectangle(double width,
              double height);
    double CalcArea();

    void SetWidth(double width);
    double GetWidth();
    void SetHeight(double height);
    double GetHeight();

private:
    double width;
    double height;
};

// The setter and getter
// for height are similar
// to the setter and getter
// for width
```

Rectangle.cpp

```
#include “Rectangle.h”

Rectangle::Rectangle()
{
    width = 0;
    height = 0;
}

Rectangle::Rectangle(double width,
       double height) : Rectangle()
{
    SetWidth(width);
    SetHeight(height);
}

double Rectangle::CalcArea()
{
    return width * height;
}

void Rectangle::SetWidth(
                    double width) {
{
    if (width >= 0)
        this->width = width;
}

double Rectangle::GetWidth()
{
    return width;
}
```

Source.cpp

```
#include <iostream>
#include “Rectangle.h”

int main()
```

```cpp
{
    Rectangle rect(4, 6);           // ⟺ Rectangle rect{4, 6};
    std::cout << rect.calcArea();

    return 0;
}
```