📘 Adding Methods to Classes in C++

In C++, class methods (member functions) can be **declared** inside the class and **defined** outside the class body. This separation improves code organization, readability, and maintainability—especially in large-scale software systems.

🔹 Declaration vs. Definition

Declaration

- Introduces the function's **signature** (return type, name, parameters).
- Placed **inside the class definition**, typically in a **header file** (`.h` or `.hpp`).
- Does **not** include the function body.

Header File: `Rectangle.h`

```cpp
class Rectangle {
private:
    double width;
    double height;

public:
    Rectangle(double w, double h);  // Constructor declaration
    double area() const;            // Method declaration
    void scale(double factor);      // Method declaration
};
```

Definition

- Provides the **actual code** for the function.
- Placed **outside the class**, typically in a **source file** (`.cpp`).
- ❗ Must use the **class scope resolution operator** (`::`) to indicate that the function belongs to the class.

Source File: `Rectangle.cpp`

❗ Make sure to put the `ClassName::` on the method name so that the compiler knows what class that the method belongs to. In the code below, `Rectangle::` was added to the method names.

```cpp
#include "Rectangle.h"

// Constructor definition using class scope
Rectangle::Rectangle(double w, double h) : width(w), height(h) {}

// Method definition using class scope
double Rectangle::area() const {
    return width * height;
}

// Another method definition using class scope
void Rectangle::scale(double factor) {
    width *= factor;
    height *= factor;
}
```

🔍 Why Use the Class Scope (`ClassName::`)?

The **scope resolution operator** `::` is essential when defining methods outside the class because:

- It **binds the function definition to the class** it belongs to.
- Without it, the compiler would treat the function as a **free-standing function**, not a class member.
- It maintains **clear modularity** between interface (declaration) and implementation (definition).

✅ Best Practices

- Always use `ClassName::` when defining methods outside the class.
- Keep class declarations in header files and definitions in source files.
- Use `const` for methods that do not modify the object's state.
- Inline short methods only when performance is critical and the logic is trivial.

🎯 Quiz!

Here's a short quiz on the topic: quiz

Footer Separator

🔭 Markdown Viewer

How to view the markdown files in a browser...

- Markdown Viewer

🧠 Lecture Practices

Here are the lecture Practices...

- Day 1
- Day 2
- Day 3

🔍 Lecture Quizzes

Here are the lecture quizzes...

- Day 1
- Day 2
- Day 3

⚙️ Weekly Topics

Here are the topics for the week...

- Calling Methods
- Calling Methods 2
- Creating Methods
- Iterators
- Vectors
- References
- Const
- Erasing in a Loop
- Default Parameters