Modern C++ Course

Gammasoft

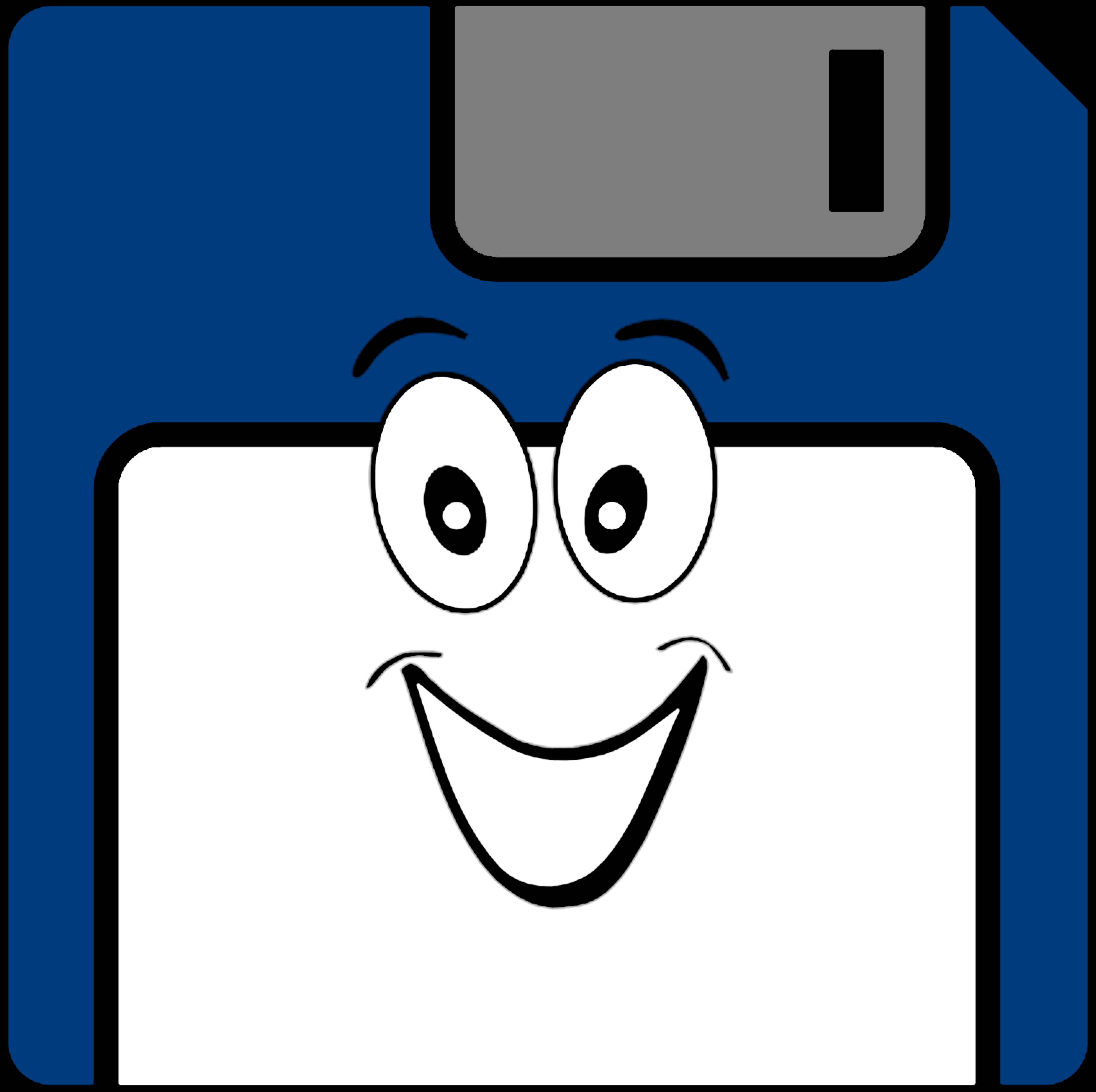
Who am I ?

Gammasoft

Gammasoft aims to make c++ fun again.

About

- Gammasoft is the nickname of Yves Fiumefreddo.
- More than thirty years of passion for high technology especially in development (c++, c#, objective-c, ...).
- Object-oriented programming is more than a mindset.
- more info see my GitHub : https://github.com/gammasoft71



Outline

Outline

Outline

Outline

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Classes (or “user-defined types”)

- structs on steroids
    - with inheritance
    - with access control
    - including methods (aka. member functions)

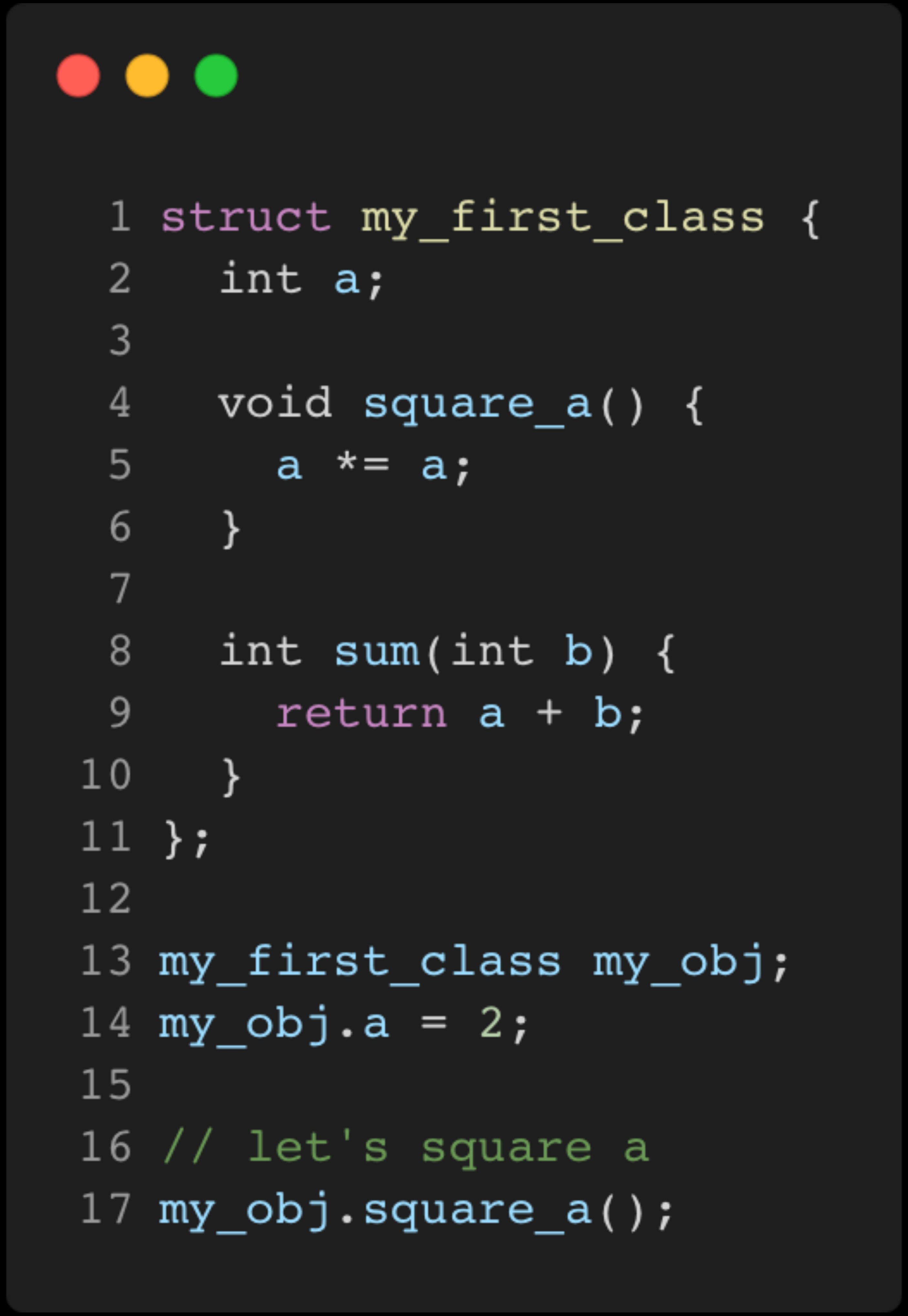
Objects

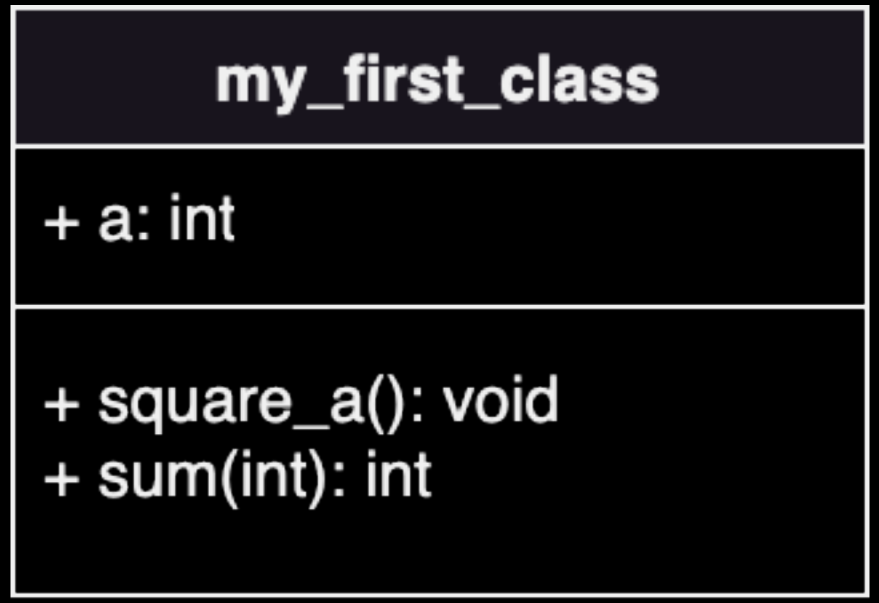
- instances of classes

A class encapsulates state and behavior of “something”

- shows an interface
- provides its implementation
    - status, properties
    - possible interactions
    - construction and destruction

My first class


```cpp
struct my_first_class {
  int a;

  void square_a() {
    a *= a;
  }

  int sum(int b) {
    return a + b;
  }
};

my_first_class my_obj;
my_obj.a = 2;

// let's square a
my_obj.square_a();
```



Separating the interface

Header: my_class.hpp

```cpp
#pragma once

struct my_class {
  int a;

  void square_a();
};
```

User 1: main.cpp

```cpp
#include "my_class.hpp"

int main() {
  my_class mc;
  //...
}
```

Implementation: my_class.cpp

```cpp
#include "my_class.hpp"

void my_class::square_a() {
  a *= a;
}
```

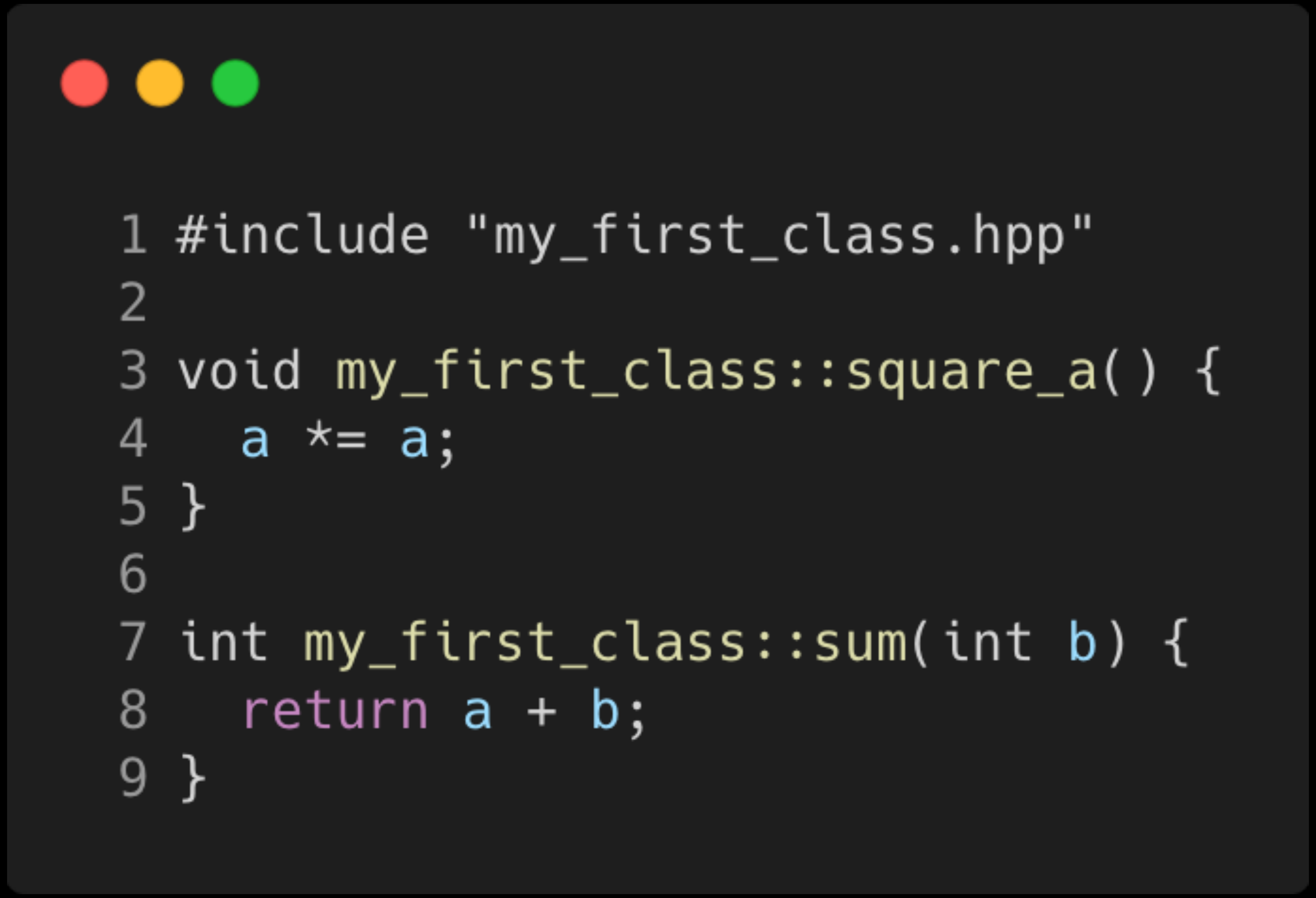
User 2: fun.cpp

```cpp
#include "my_class.hpp"

void fun(my_class& mc) {
  mc.square_a();
}
```

Implementing methods

Good practice

- usually in .cpp, outside of class declaration
- using the class name as “namespace”
- short member functions can be in the header
- some functions (templates, constexpr) must be in the header

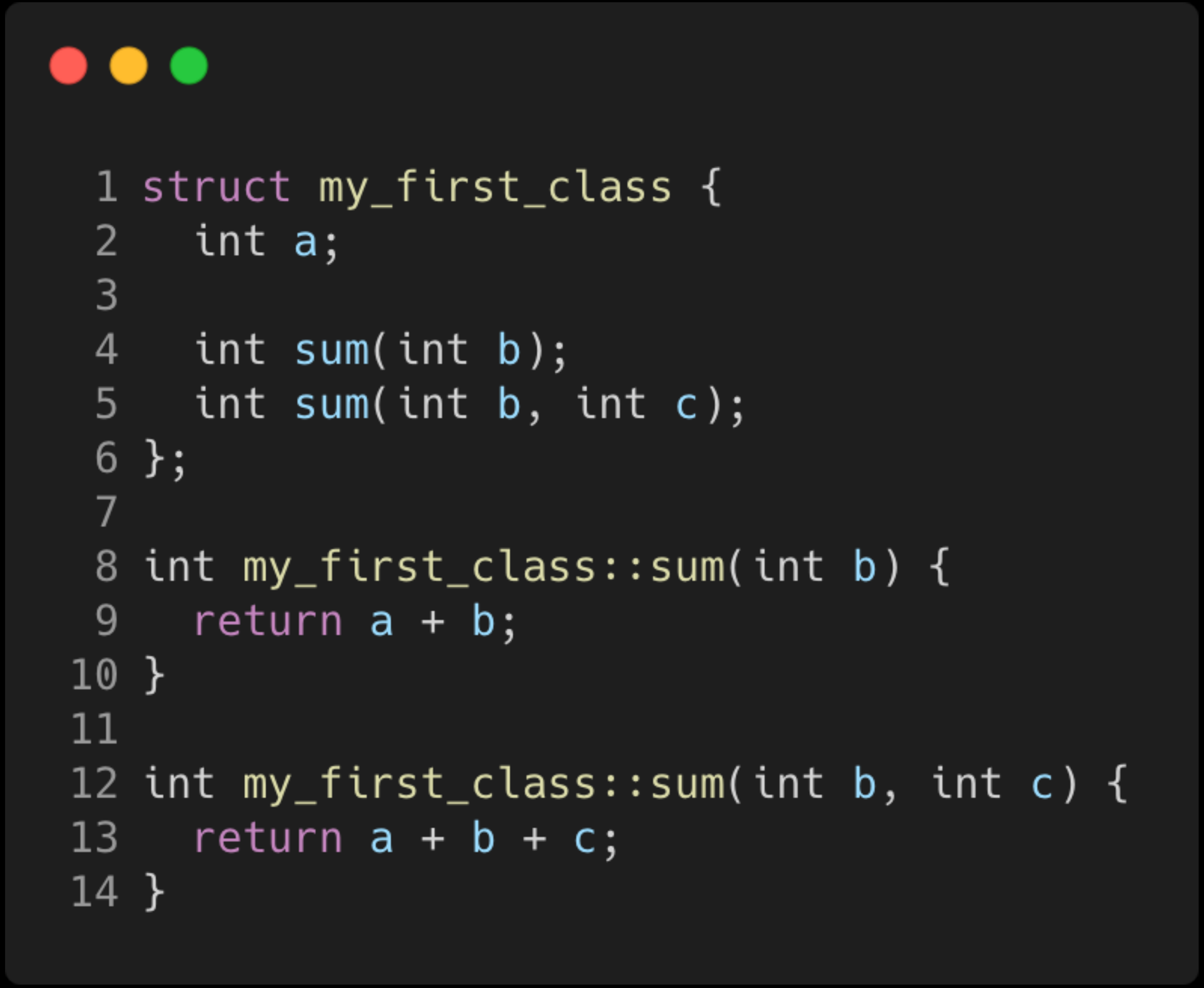


```cpp
#include "my_first_class.hpp"

void my_first_class::square_a() {
  a *= a;
}

int my_first_class::sum(int b) {
  return a + b;
}
```

Method overloading

The rules in C++

- overloading is authorized and welcome
- signature is part of the method identity
- but not the return type



```cpp
struct my_first_class {
  int a;

  int sum(int b);
  int sum(int b, int c);
};

int my_first_class::sum(int b) {
  return a + b;
}

int my_first_class::sum(int b, int c) {
  return a + b + c;
}
```

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Objects Oriented Programming (OOP)

- Objects and classes
- Inheritance
- Constructors / Destructors
- Static members
- Allocating objects
- Advanced Object Oriented
- Type casing
- Operator overloading
- Function objects
- Name Lookups

Outline

End