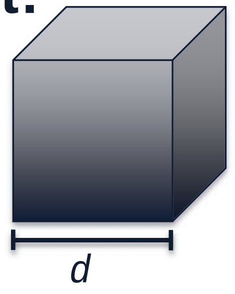
C++ Classes

Prof. Wade Fagen-Ulmschneider

ILLINOIS

C++ classes encapsulate data and associated functionality into an **object**:

Object:



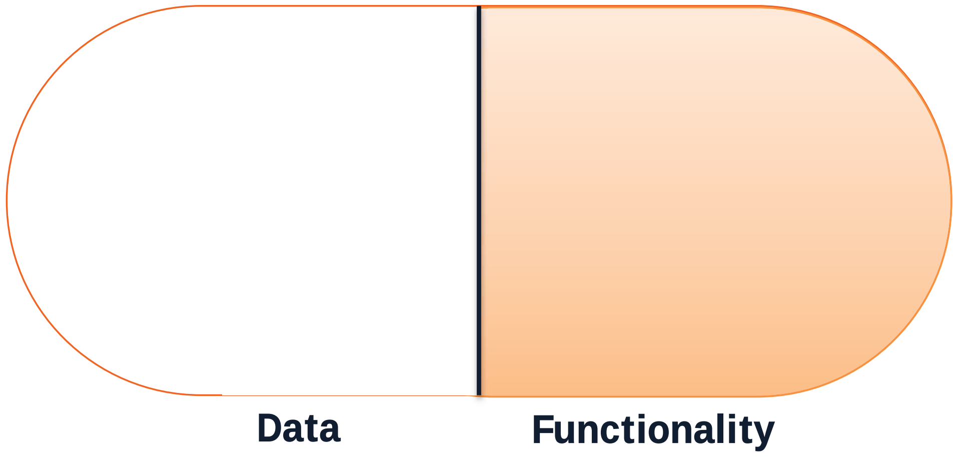
C++ class:

```
class Cube {
  public:
    double getVolume();
    // ...

  private:
    double length_;
};
```

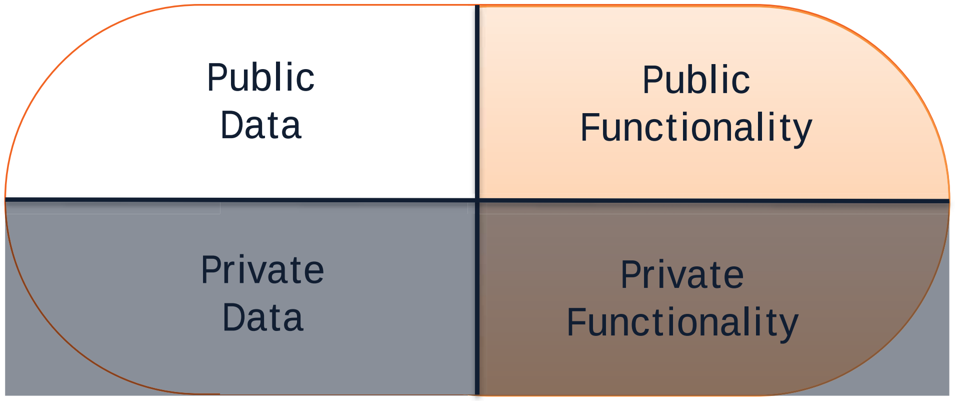
Encapsulation

Encapsulation encloses data and functionality into a single unit (called a **class**):



Encapsulation #1

In C++, data and functionality are separated into two separate protections: **public** and **private**.



Public vs. Private

The protection level determines the access that “client code” has to the member data or functionality:

- **Public** members can be accessed by client code.

- **Private** members cannot be accessed by client code (only used within the class itself).

Encapsulation #2

In C++, the **interface** (.h file) to the class is defined separately from the **implementation** (.cpp file).

C++ Header File (.h)

A header file (.h) defines the interface to the class, which includes:

- Declaration of **all** member variables
- Declaration of **all** member functions

cpp-class/Cube.h

```cpp
#pragma once
…
class Cube {
  public:
    double getVolume();
    double getSurfaceArea();
    void setLength(double length);

  private:
    double length_;
};
```

C++ Implementation File (.cpp)

An implementation file (.cpp) contains the code to implement the class (or other C++ code).

cpp-class/Cube.cpp

```cpp
#include "Cube.h"

double Cube::getVolume() {
  return length_ * length_ * length_;
}

double Cube::getSurfaceArea() {
  return 6 * length_ * length_;
}

void Cube::setLength(double length) {
  length_ = length;
}
```

cpp-class/main.cpp

```cpp
#include "Cube.h"

int main() {
  Cube c;

  c.setLength(3.48);
  double volume = c.getVolume();
  std::cout << "Volume: " << volume << std::endl;

  return 0;
}
```