Inheritance

Prof. Wade Fagen-Ulmschneider

Inheritance allows for a class to inherit all member functions and data from a **base class** into a **derived class**.

Generic to Specialized

A **base class** is a generic form of a specialized, **derived class**.

Shape ➔ Cube

cpp-inheritance/Shape.h

```
8  #pragma once
9
10 class Shape {
11   public:
12     Shape();
13     Shape(double width);
14     double getWidth() const;
15
16   private:
17     double width_;
   };
```

cpp-inheritance/Cube.h

```
8  #pragma once
9
10 #include "Shape.h"
11 #include "HSLAPixel.h"
12
13 namespace uiuc {
14   class Cube : public Shape {
15     public:
16       Cube(double width, uiuc::HSLAPixel color);
17       double getVolume() const;
18
19     private:
20       uiuc::HSLAPixel color_;
21   };
22 }
```

Initialization

When a derived class is initialized, the derived class **must** construct the base class:

- `Cube` must construct `Shape`
- By default, uses default constructor
- Custom constructor can be used with an initialization list

cpp-inheritance/Cube.cpp

```cpp
#include "Cube.h"
#include "Shape.h"

namespace uiuc {
  Cube::Cube(double width, uiuc::HSLAPixel color) : Shape(width) {
    color_ = color;
  }

  double Cube::getVolume() const {
    // Cannot access Shape::width_ due to it being `private`
    // ..instead we use the public Shape::getWidth(), a public function:

    return getWidth() * getWidth() * getWidth();
  }
}
```

Access Control

When a base class is inherited, the derived class:

- Can access all **public** members of the base class
- Can **not** access **private** members of the base class

Initializer List

The syntax to initialize the base class is called the initializer list and can be used for several purposes:

- Initialize a base class
- Initialize the current class using another constructor
- Initialize the default values of member variables

cpp-inheritance/Cube.cpp

```
8  #include "Shape.h"
9
10 Shape::Shape() : Shape(1) {
11   // Nothing.
12 }
13
14 Shape::Shape(double width) : width_(width) {
15   // Nothing.
16 }
17
18 double Shape::getWidth() const {
19   return width_;
20 }
```