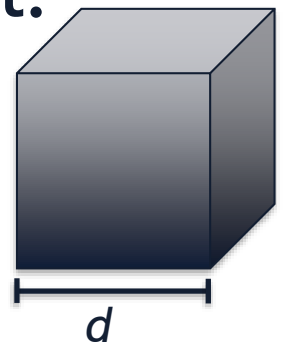
Class Constructors

Prof. Wade Fagen-Ulmschneider

When an instance of a class is created, the **class constructor** sets up the initial state of the class.

Object:



Default: Unit Cube (d=1)

Automatic Default Constructor

If we do not provide any custom constructors, the C++ compiler provides an **automatic default constructor** for our class for free!

The automatic default constructor will only initialize all member variables to their default values.

cpp-std/Cube.h

```
8  #pragma once
9
10 namespace uiuc {
11   class Cube {
12     public:
13       double getVolume();
14       double getSurfaceArea();
15       void setLength(double length);
16
17     private:
18       double length_;
19   };
20 }
```

Custom Default Constructor

The simplest constructor we can provide is a **custom default constructor** that specifies the state of the object when the object is constructed. We define one by creating:

- A member function with the same name of the class
- The function takes zero parameters.
- The function does not have a return type.

```
Cube::Cube()  // custom default constructor
```

cpp-ctor/ex1/Cube.h

```
1  #pragma once
2
3  namespace uiuc {
4    class Cube {
5      public:
6        Cube(); // Custom default constructor
7
8        double getVolume();
9        double getSurfaceArea();
10       void setLength(double length);
11
12     private:
13       double length_;
14   };
15 }
16
17
```

cpp-ctor/ex1/Cube.cpp

```
 8 #include "Cube.h"
 9
10 namespace uiuc {
11   Cube::Cube() {
12     length_ = 1;
13   }
14
...    ...
```

cpp-ctor/ex1/main.cpp

```cpp
#include "Cube.h"
#include <iostream>

int main() {
  uiuc::Cube c;
  std::cout << "Volume: " << c.getVolume() << std::endl;
  return 0;
}
```

Custom Constructors

We can also specify custom, non-default constructors that require client code to supply arguments:

```
Cube::Cube(double length)
 // one-argument ctor specifying initial length
```

cpp-ctor/ex2/Cube.h

```
8  #pragma once
9
10 namespace uiuc {
11   class Cube {
12     public:
13       Cube(); // Custom default constructor
14       Cube(double length); // One argument constructor
15
16       double getVolume();
17       double getSurfaceArea();
18       void setLength(double length);
19
20     private:
21       double length_;
22   };
23 }
```

cpp-ctor/ex2/Cube.cpp

```
 8 #include "Cube2.h"
 9
10 namespace uiuc {
11   Cube::Cube() {
12     length_ = 1;
13   }
14
15   Cube::Cube(double length) {
16     length_ = length;
17   }
18
...    ...
```

cpp-ctor/ex2/main.cpp

```cpp
#include "Cube.h"
#include <iostream>

int main() {
  uiuc::Cube c(2);
  std::cout << "Volume: " << c.getVolume() << std::endl;
  return 0;
}
```

Automatic Default Constructor

If any custom constructor is defined, an automatic default constructor is not defined.

cpp-ctor/ex3/main.cpp

```
 8 #include "Cube.h"
 9 #include <iostream>
10
11 int main() {
12   uiuc::Cube c; // !!!
13   std::cout << "Volume: " <<
       c.getVolume() <<
       std::endl;
14   return 0;
15 }
```

error

ex3/Cube.h

```
 8 #pragma once
 9
10 namespace uiuc {
11   class Cube {
12     public:
13       Cube(double length);
14
15       double getVolume();
16       double getSurfaceArea();
17       void setLength(double);
18
19     private:
20       double length_;
21   };
22 }
```