CS 225 Spring 2019 :: TA Lecture Notes
1/14 Intro

By Wenjie

Thanks to all of you who pointed out typos in the notes, and gave formatting suggestions:)
But the google doc seems auto-reject comments from time to time - please lmk if you ran into this!

- **Variables in C++**
  - **Properties of each variable**
    - ➢ **Type** : primitive vs user-defined

- **Type**
  - **Primitive type variables**
    - Similar to what you have seen in Java
    - int/char/double/boolean/float/pointer (See more in wikipedia)
  - **User-defined variable/complex variable/object**
    - We use classes to define new variable types

- **Encapsulation**
  - The point of encapsulation is to separate the **interface** from **implementation** but still keep them as a cohesive unit
    - **Interface/the API** - **what** is the class supposed to do
    - **Implementation** - **how** is the class supposed to do it
  - With such separation between interface and implementation, we could update our coding implementation of function without update our interface
  - In C++ convention, normally we put the interface in a file with ".h" extension and the implementation in a file with ".cpp" extension.

cube.h		cube.cpp	
1 2 3	Interface/API	1 2 3	Implementation

CS 225 Spring 2019 :: TA Lecture Notes
1/14 Intro

By Wenjie

- Classes are like containers of holding variables and methods. Therefore, to **define** class is to specify its components - all its member including variables and methods.
  - This is within the .h file. Technically, defining a class is creating its API.
- On the other side, the member variables and methods are said to be **declared** within the ".h" file. This means that we have determined their components - their return type, name, and parameters.
  - A class is **defined** by specifying its members, however, methods and variables are **declared** when we specify their components.

• Inclusion guards

- **"#pragma once"** is like sending a message to the compiler such that this particular file will be included only once within this single compilation.
- Either way works. So pick one and be consistent. However it seems that now we are moving more towards "#pragma once" style.

cube.h

```
1  #ifndef CUBE_H_
2  #define CUBE_H_
3
   class Cube {
        public:


        private:

   };

   #endif
```

Equivalent

cube.h

```
1  #pragma once
2
3  class Cube {
        public:


        private:

   };
```

CS 225 Spring 2019 :: TA Lecture Notes
1/14 Intro

By Wenjie

A class is **defined** by specifying its members. However, methods and variables are **declared** when we specify their components. We implement methods and initialize variables within the .cpp file.
For example:

cube.h

```
#pragma once

class Cube {
     public:
           Double
getVolumn();
     ...
}
```

cube.cpp

```
#include "Cube.h"

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

- **Scope resolution**
  - **Example**
    Cube::getVolume() means the getVolume() method of class Cube

To be continued..