#pragma once

class Cube {
public:
    double getVolume();
private:
};

#include "Cube.h"

double Cube::getVolume() {
}
Namespaces
Namespaces

cs225

Cube
PNG
HSLAPixel

std

cout
vector
queue
...

...
```cpp
#pragma once

namespace cs225 {
    class Cube {
        public:
            double getVolume();
            double getSurfaceArea();
    private:
            double length_;
    };
}
```

```cpp
#include "Cube.h"

namespace cs225 {
    double Cube::getVolume() {
        return length_ * length_ * length_;
    }

    double Cube::getSurfaceArea() {
        return 6 * length_ * length_;
    }
}
```
```cpp
#pragma once

namespace cs225 {
    class Cube {
    public:
        double getVolume();
        double getSurfaceArea();
    private:
        double length_;  
    };
}

#include "Cube.h"

namespace cs225 {
    double Cube::getVolume() {
        return length_ * length_ * length_;
    }

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

#include <iostream>

int main() {
    cs225::Cube c;
    std::cout << "Volume: " << c.getVolume() << std::endl;
    return 0;
}
```
```cpp
#include "Cube.h"
#include <iostream>

int main() {
    cs225::Cube c;
    std::cout << "Volume: " << c.getVolume() << std::endl;
    return 0;
}
```
```cpp
#include "Cube.h"
#include <iostream>

int main() {
    cs225::Cube c;
    std::cout << "Volume: " << c.getVolume() << std::endl;
    return 0;
}
```
```cpp
#include "Cube.h"
#include <iostream>

int main() {
    cs225::Cube c;
    std::cout << "Volume: " << c.getVolume() << std::endl;
    return 0;
}
```
Hate typing cout::__ and cs225::__ multiple times?

Useful Shortcut:
using std::__cout;  // Imports cout into global scope
using std::__endl;  // Imports endl into global scope
using cs225::__Cube; // Imports Cube into global scope

Strongly Discouraged Shortcut
using namespace std;  // Imports all of std::__ into
// global scope!
// ...THOUSANDS of things!
```cpp
#include "Cube.h"
using cs225::Cube;
#include <iostream>
using std::cout;
using std::endl;

int main() {
    Cube c;
    cout << "Volume: " << c.getVolume() << endl;
    return 0;
}
```
CS 225 – Office Hours

Lab Sections – Meet with your TA and CAs every week!

Open Office Hours – Held in the basement of Siebel Center by TAs and CAs, ramping up over the next week. First open office hours start this Thursday. (*Great place for both conceptual and programming questions!*)

Faculty Office Hours –
Craig’s Office Hours: This week, Thursday 9-11am in Siebel 4112
Wade’s Office Hours: TBA
CS 225 – Exam 0

First exam is coming up next week!

“Exam 0”

• Low-stress introduction to the CBTF exam environment.
• This exam is worth only 40 points
• Focuses primarily on foundational knowledge you have from your prerequisite classes.

Full Details:

https://courses.engr.illinois.edu/cs225/sp2019/exams/
CBTF-based Exams

All CS 225 exams are held in the Computer Based Testing Facility (CBTF):
- You can choose which day to take your exam within the exam window for a given exam.

- Sign up for your exam here: https://cbtf.engr.illinois.edu/
Constructor
```cpp
#pragma once

namespace cs225 {

class Cube {
public:
    Cube();
    double getVolume();
    double getSurfaceArea();

private:
    double length_;  
};
}

#include "Cube.h"

namespace cs225 {

Cube::Cube() {}  

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

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

}
```
#pragma once

namespace cs225 {

class Cube {
public:
  Cube(double length);
  double getVolume();
  double getSurfaceArea();

private:
  double length_;}

#include "Cube.h"

namespace cs225 {

Cube::Cube(double length) {
}

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

double Cube::getSurfaceArea() {
  return 6 * length_ * length_;
}
```cpp
#pragma once

namespace cs225 {

  class Cube {
  public:
    Cube(double length);
    double getVolume();
    double getSurfaceArea();
  
    double length_; 
  
  private:
  
  };

}

#include "Cube.h"

namespace cs225 {

  Cube::Cube(double length) {
    length_ = length;
  }

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

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

}

#include "Cube.h"
using cs225::Cube;
#include <iostream>
using std::cout;
using std::endl;

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

namespace cs225 {

class Cube {
    public:
        Cube(double length);
        double getVolume();
        double getSurfaceArea();

    private:
        double length_; // length of the cube
};
}

#include "Cube.h"

namespace cs225 {

    Cube::Cube(double length) {
        length_ = length; // Assign length to the length_ member variable
    }

    double Cube::getVolume() {
        return length_ * length_ * length_; // Calculate the volume of the cube
    }

    double Cube::getSurfaceArea() {
        return 6 * length_ * length_; // Calculate the surface area of the cube
    }
}

int main() {
    Cube c; // Create an instance of the Cube class
    cout << "Volume: " << c.getVolume() << endl; // Output the volume of the cube
    return 0;
}
#pragma once

namespace cs225 {

    class Cube {
        public:
            Cube(double length);
            double getVolume();
            double getSurfaceArea();
    }

    private:
        double length_{
    }
}

#include "Cube.h"

namespace cs225 {

    Cube::Cube(double length) {
        length_ = length;
    }

double Cube::getVolume() {

    }

double Cube::getSurfaceArea() {

    }

}

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

}
Pointers and References
Pointers and References

A variable containing an instance of an object:

```cpp
1 Cube s1;
```

A reference variable of a Cube object:

```cpp
1 Cube & s1;
```

A variable containing a pointer to a Cube object:

```cpp
1 Cube * s1;
```
Reference Variable

A reference variable is an alias to an existing variable.

*Key Idea: Modifying the reference variable modifies the variable being aliased.*
Reference Variable

A reference variable is an \textit{alias} to an existing variable.

\begin{verbatim}
#include <iostream>

int main() {
    int i = 7;
    int & j = i; // j is an alias of i
    j = 4;
    std::cout << i << " " << j << std::endl;
    i = 2;
    std::cout << i << " " << j << std::endl;
    return 0;
}
\end{verbatim}
Reference Variable

Three facts about reference variables:

1.

2.

3.
CS 225 – Things To Be Doing

Exam 0 starts on Thursday, Jan. 24th
Ensure you sign up for your CBTF timeslot for Exam 0!

lab_intro is due this Sunday (Jan. 20th)
Make sure to attend your lab section every week – they’re worth points and EC!

MP1 is released Friday!
Due: Monday, Jan. 28th (~10 days after release)

Ensure you are on our Piazza
Details on the course website: https://courses. engr.illinois.edu/cs225/

See you Friday!