CS 225

Data Structures

Wade Fagen-Ulmschneider

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H

class Sphere {
  public:
    double getRadius();





  private:


};

#endif
```

sphere.cpp

```
#include "sphere.h"

double Sphere::getRadius() {


}
```

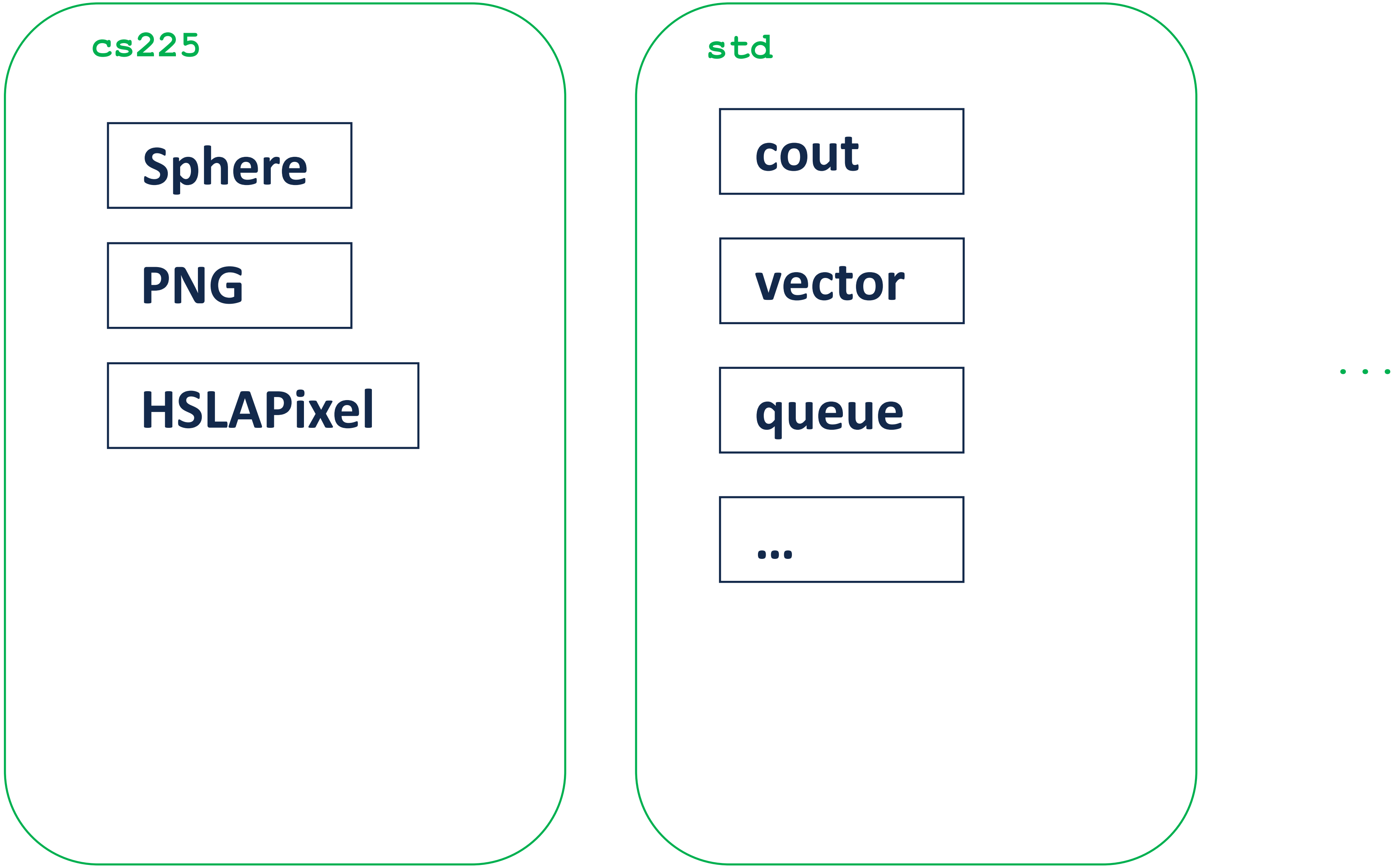
main.cpp

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

int main() {
  cs225::Sphere s;
  std::cout << "Radius: " << s.getRadius() << std::endl;
  return 0;
}
```

Namespaces

Namespaces



cs225

Sphere

PNG

HSLAPixel

std

cout

vector

queue

...

. . .

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H

namespace cs225 {
  class Sphere {
    public:
      double getRadius();
      double getVolume();




    private:
      double r_;

  };
}

#endif
```

sphere.cpp

```
#include "sphere.h"

namespace cs225 {
  double Sphere::getRadius() {
    return r_;
  }

  double Sphere::getVolume() {
    return (4 * r_ * r_ * r_ *
            3.14159265) / 3.0;
  }
}
```

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H

namespace cs225 {
  class Sphere {
    public:
      double getRadius();
      double getVolume();








}

#e

```

sphere.cpp

```
#include "sphere.h"

namespace cs225 {
  double Sphere::getRadius() {
    return r_;
  }

  double Sphere::getVolume() {
    return (4 * r_ * r_ * r_ *
                3.14159265) / 3.0;
  }
}
```

main.cpp

```
#include "sphere.h"
#include <iostream>

int main() {
  cs225::Sphere s;
  std::cout << "Radius: " << s.getRadius() << std::endl;
  return 0;
}
```

main.cpp

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

int main() {
  cs225::Sphere s;
  std::cout << "Radius: " << s.getRadius() << std::endl;
  return 0;
}
```

main.cpp

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

int main() {
  cs225::Sphere s;
  std::cout << "Radius: " << s.getRadius() << std::endl;
  return 0;
}
```

main.cpp

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

int main() {
  cs225::Sphere s;
  std::cout << "Radius: " << s.getRadius() << std::endl;
  return 0;
}
```

CS 225 – Office Hours

Lab Sections – Meet with your CAs and TAs each week!

Open Office Hours – Held in the basement of Siebel Center by TAs and CAs, ramping up over the next week. Available open office hours are posted on the course website.
(With the basement of Siebel having many labs, great place for both conceptual and programming questions!)

Faculty Office Hours –

Wade's Office Hours: Wednesdays, 12:20-1:40pm, 4034 ECEB
(Between the two lectures, held right here in ECEB!)

Eric's Office Hours: Thursdays, 2-3pm, 2221 Siebel Center

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/sp2018/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

sphere.h

```cpp
#ifndef SPHERE_H
#define SPHERE_H
namespace cs225 {
  class Sphere {
    public:
      Sphere();
      double getRadius();
      double getVolume();




    private:
      double r_;

  };
}

#endif
```

sphere.cpp

```cpp
#include "sphere.h"
namespace cs225 {
  Sphere::Sphere() {


  }

  double Sphere::getRadius() {
    return r_;
  }

  double Sphere::getVolume() {
    return (4 * r_ * r_ * r_ *
                3.14159265) / 3.0;
  }
}
```

sphere.h

```cpp
#ifndef SPHERE_H
#define SPHERE_H
namespace cs225 {
  class Sphere {
    public:
      Sphere(double r);
      double getRadius();
      double getVolume();




    private:
      double r_;

  };
}

#endif

```

sphere.cpp

```cpp
#include "sphere.h"
namespace cs225 {
  Sphere::Sphere(double r) {


  }

  double Sphere::getRadius() {
    return r_;
  }

  double Sphere::getVolume() {
    return (4 * r_ * r_ * r_ *
            3.14159265) / 3.0;
  }
}
```

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H
namespace cs225 {
  class Sphere {
    public:
      Sphere(double r);
      double getRadius();
      double getVolume();








}

#
```

sphere.cpp

```
#include "sphere.h"
namespace cs225 {
  Sphere::Sphere(double r) {
     r_ = r;

  }

  double Sphere::getRadius() {



                                  () {
                                  r_ *
                                  3.0;
```

main.cpp

```
#include "sphere.h"
#include <iostream>

using namespace std;
using namespace cs225;

int main() {
  Sphere s;
  cout << "Radius: " << s.getRadius() << endl;
  return 0;
}
```

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H
namespace cs225 {
  class Sphere {
    public:

      Sphere(double r);
      double getRadius();
}

#endif
```

sphere.cpp

```
#include "sphere.h"
namespace cs225 {




  Sphere::Sphere(double r) {
    r_ = r;


                                        () {



                                        () {
                                        r_ *
                                        3.0;
```

main.cpp

```
#include "sphere.h"
#include <iostream>

using namespace std;
using namespace cs225;

int main() {
  Sphere s;
  cout << "Radius: " << s.getRadius() << endl;
  return 0;
}
```

sphere.h

```
#ifndef SPHERE_H
#define SPHERE_H
namespace cs225 {
  class Sphere {
    public:

      Sphere(double r);
      double getRadius();
```

```
}

#endif
```

sphere.cpp

```
#include "sphere.h"
namespace cs225 {




  Sphere::Sphere(double r) {
    r_ = r;
```

```
() {
```

```
() {
r_ *
3.0;
```

main.cpp

```
#include "sphere.h"
#include <iostream>

using namespace std;
using namespace cs225;

int main() {
  Sphere s;
  cout << "Radius: " << s.getRadius() << endl;
  return 0;
}
```

Pointers and References

Pointers and References

A variable containing an instance of an object:

```
Sphere s1;
```

A reference variable of a Sphere object:

```
Sphere & s1;
```

A variable containing a pointer to a Sphere object:

```
Sphere * 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 alias to an existing variable.

```
#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;
}
```

Reference Variable

Three facts about reference variables:

1.

2.

3.

CS 225 – Things To Be Doing

Exam 0 starts on Tuesday, Jan 23rd

Ensure you sign up for your CBTF timeslot for Exam 0!

lab_intro is due this Sunday (Jan. 21st)

Make sure to attend your lab section every week – they're worth points!

MP1 is released today!

Due: Monday, Jan. 29th (about 10 days from now)

Ensure you are on our Piazza

Details on the course website: https://courses.engr.illinois.edu/cs225/

See you Monday!