Today's announcements:

Course policies: http://cs.illinois.edu/class/cs225

general assistance (ews, svn, etc.) - post to piazza

HW0 available, due 1/23 before lecture.

MP1 available, due 1/22, 11:59p.

Proficiency exam Saturday - signup instructions on web site.

Today's plan:

Ideas/concepts:

- Class definitions
- Class member functions - declaration and implementation
- Constructors
- Clients

OOP: C++

- I
- E
- P

Our first class…

sphere.h

```
class sphere{

};
```

What surprises you about this code?

main.cpp

```
#include "sphere.h"

int main(){
   sphere a;
}
```

1. Upon command > `g++ main.cpp` does this code compile?
2. Upon command > `./a.out` does it run?

Access control and encapsulation:

sphere.h

```
class sphere{
   double theRadius;
};
```

What surprises you about this code?

main.cpp

```
#include "sphere.h"
#include <iostream>
using namespace std;

int main(){
   sphere a;
   cout << a.theRadius << endl;
}
```

1. Upon command `> g++ main.cpp` does this code compile?
2. Upon command `> ./a.out` does it run?
3. In c++ class members are, by default, "private". Why would we want to hide our representation of an object from a client?
4. How many collaborators are you allowed to have for MPs in this class?

Structure of a class defn (cont):

```
class sphere{
// member fn and data
public:
sphere();
sphere(double r);
void setRadius(double newRad);
double getDiameter();

private:
double theRadius;
};
```

`sphere` functionality:

1. *Create a new one.*
2. *Change an existing one.*
3. *Get information about one.*

`sphere` representation:

radius

```
int main(){




};
```

Structure of a class defn (cont):

```
class sphere{
public:
sphere();
sphere(double r);
void setRadius(double newRad);
double getDiameter() const;
…


private:
double theRadius;
};
```

```
//constructor(s) (next page)

void sphere::setRadius(double newRad){


}

double sphere::getDiameter() const {


}
…
```

Asides:

_____:

__:

Constructors (intro): When you *declare* a sphere, a sphere class constructor is invoked.

Points to remember abt ctors:

1.

2.

3.

```
int main(){



}
```

```
…
//default constructor
sphere::sphere(){

}

//default constructor, alternative
sphere::sphere()
{

}

//constructor with given radius
sphere::sphere(double r){




}
...
```

Class Definition… where are we?

Today's plan:

Ideas/concepts:

- Class definitions
- Class function implementation
- Constructors
- Clients

OOP: we now understand how C++ achieves

- I
- Encapsulation
- P