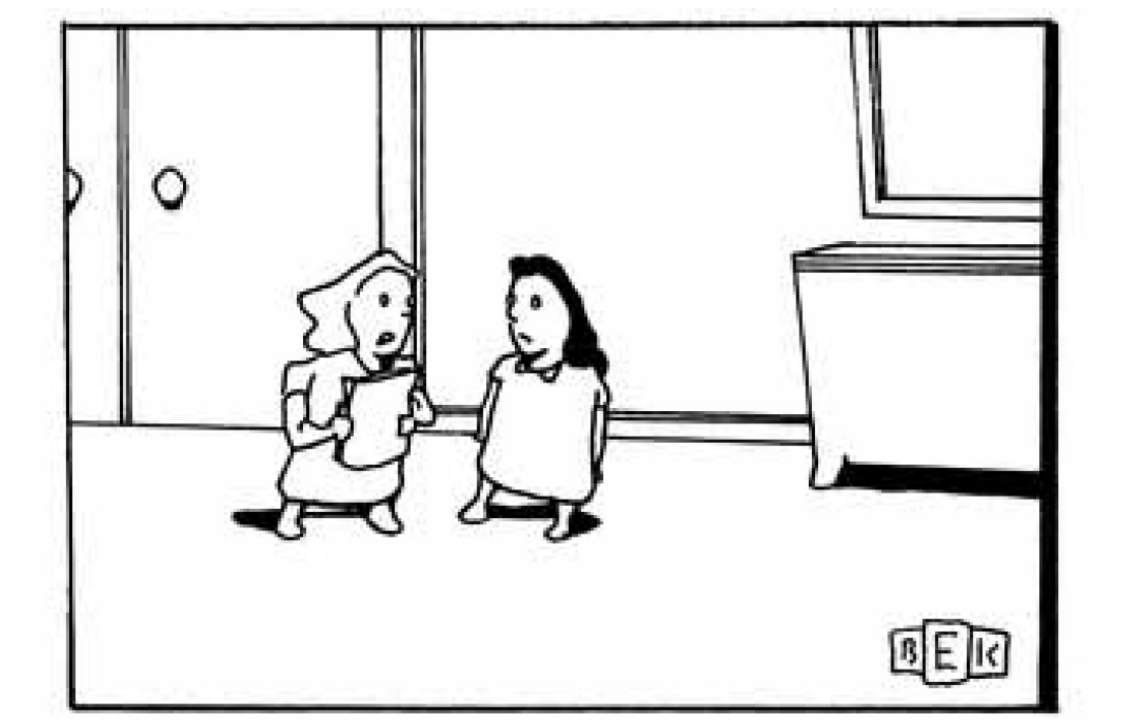
CS 32

Lecture 6: Scoping



"You know, it's a picture book. Needless to say, it doesn't have a lot of narrative scope."

DWEM

“To the strongest and quickest mind it is far easier to learn than to invent. The principles of arithmetic and geometry may be comprehended by a close attention in a few days; yet who can flatter himself that the study of a long life would have enabled him to discover them, when he sees them yet unknown to so many nations, whom he cannot suppose less liberally endowed with natural reason, than the Grecians or Egyptians?” —Samuel Johnson, 1751

Our Mission

- To the strongest and quickest mind it is far easier to learn than to invent.
  - Right, that's why we're here (UCSB)
  - Stand on the shoulders of giants
  - Could have been a snub to Hooke

Learning is Easier Than...

- “The principles of arithmetic and geometry may be comprehended by a close attention in a few days...”
  - He is being metaphorical
  - A few days means “less than a lifetime”

...Doing It Yourself

- Yet who can flatter himself that the study of a long life would have enabled him to discover them,
  - Yeah, like we'd have discovered DNA
  - Or gravity waves

Level Playing Field

- “…so many nations, whom he cannot suppose less liberally endowed with natural reason, than the Grecians or Egyptians?”
  - The Greeks and Egyptians were smart, but…
  - …so was everybody else
    - Australasians and Aztec and Indonesian and Polynesian and …
  - And they still are

Scoping?

- Does not mean we are scoping something out
- Scopes are environments of variables
- Often nested
- Levels of privacy for variables
- But PS 10.2 was about classes, why is the lecture called “Scoping”?

Scoping!

- Keep internal details hidden
- Present a simple interface
- When you declare a class, you are creating a scope
  - That's why we need a scope resolution operator
  - What's that?

Where's the Meat?

- Programmers want separate interface & implementation sections
- So the .h and .cpp files
- But this means you write the "guts" somewhere else, outside the scope of the class declaration

Scope Resolved

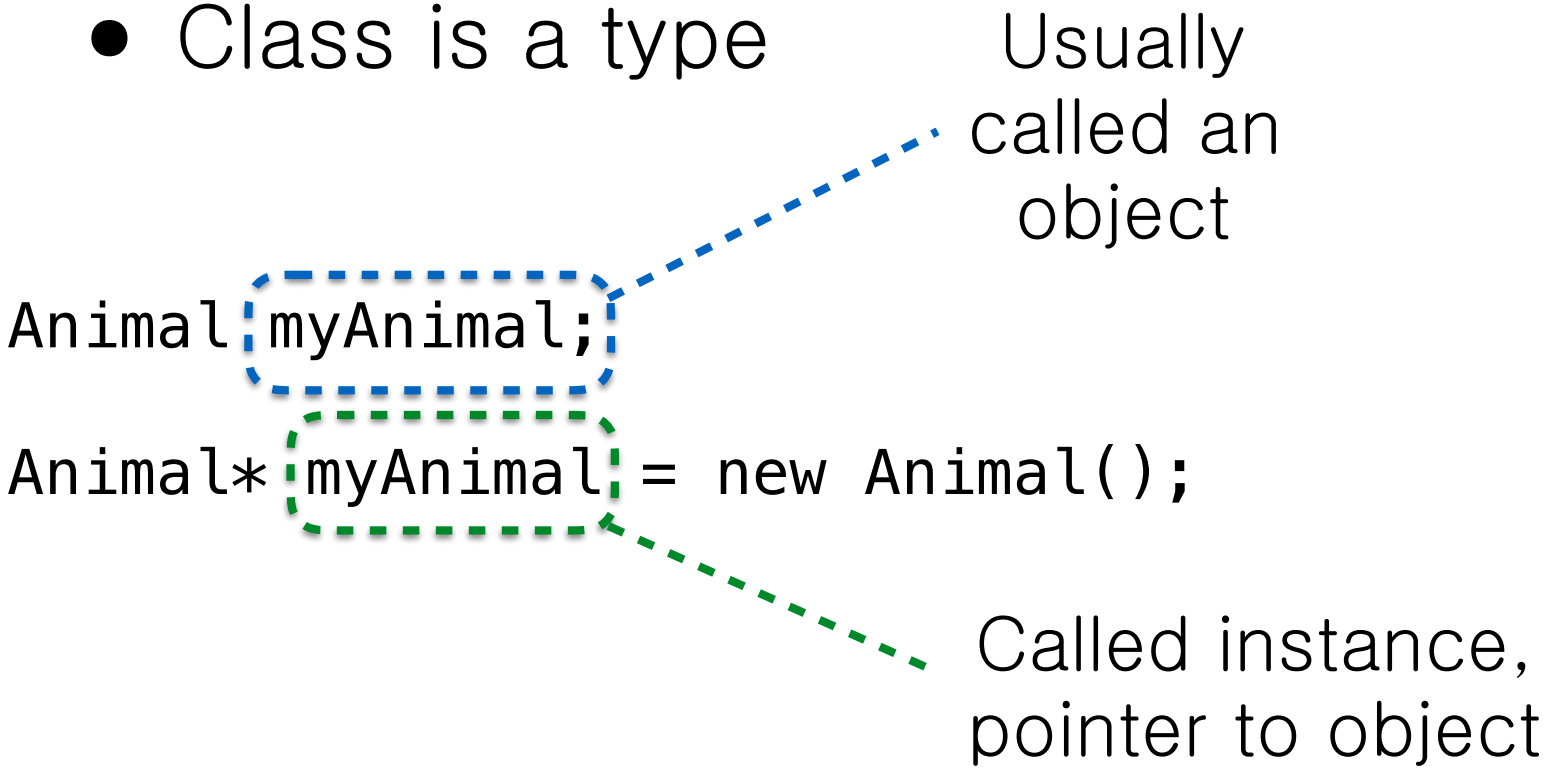
- After some chain of header files, the compiler gets to your code
- You want to implement a method
- Compiler needs to know which class (scope) you mean
  - `void MyClass::myMethod() {…}`

Analogous to .

- Dot is used after an object name
- :: is used after a class name

Same Difference?

- Object is a value
- Class is a type



Encapsulation

- Brief mention, encapsulated in the text
- Most important benefit of OO
- Can benefit from this without inheritance hierarchy
- So there's a subset relation here...

Based

- Object-Based Programming
  - Using classes just for encapsulation
- Object-Oriented Programming
  - Using inheritance also

More Scoping

- Of a kind
- Can declare members
  - `public`
  - `private`
- Oh wait, what are members?

Membership

- Just as with structs, you have member variables
  - They hold values, of course
- Classes are supercharged with member methods too

Best Practice

- Structs got methods too
- Struct vs. Class?
  - All members of struct are public, and…
  - …that's it!
- Recommendation: use structs only for objects with public variables and no methods.

Privacy Rights

- The default visibility level for classes is private
- Best to be explicit
- Private from whom?
  - Clients
  - Descendants too!

Accessors

- If data members are private, how to read/write?
- Accessors
  - Sometimes custom-made
  - Sometimes inlined

Get and Set

```
class Point {
  private:
    int x, y;
};

int Point::getX() { return this->x; }
void Point::setX(int newVal) { … }
```

- Traditional names
  - Objective-C generates them automatically
- A lot can go on in those {…}

Mutator?

- Setters also called mutators
- Want to constrain how the state of the program is changed
- For garbage-collectors, the whole program (except the GC) is the mutator

Assignment

- For plain objects, plain assignment means plain member copying
- For object instances, it means pointer sharing
  - Internal details — love 'em or hate 'em

Creation Myth

- If members are private, then even creating new objects requires some convention or process
- Constructors
  - An endless topic…

Constructors

- Provide some generic and some specific constructors
- Have generic ones call specific ones with “magic numbers”
- C++, Obj-C have perfected this

Special Constructors

- Two special-purpose constructors
  - Default constructor
  - Copy constructor
- Sometimes the “System” creates them
  - I.e. there’s a default default.
  - Proper term is “implicit”

Assignment

- And there's the assignment (=) operator
- It has its implicit version
- How does it differ from the copy constructor?

The Big Three

- Used in different circumstances
- Shows C++ awareness of memory
  - A handoff is different…
  - …from a new creation

```
MyClass a1, a2;   // Call default constructor
a2 = a1;          // Call assignment operator
MyClass a3 = a1;  // Call copy constructor
```

Default Constructor

- Implicit one has same effect as a user-defined constructor with empty body and empty initializer list
  - I.e. calls default constructors of base classes and instance variables

Assignment Operator

- Invoked when a pre-existing object (not a pointer) is assigned to another pre-existing one

```
void operator = (const MyClass &that ) {
  this->m1 = that.m1;
  //etc.
}
```

Copy Constructor

- Invoked when a pre-existing object (not a pointer) is assigned to a new one
- Has at least one parameter (the “other”)
  - Other params must have default arguments

```
MyClass::MyClass(const MyClass &other) {
   this->m1 = other.m1;
}
```

Why?

- Why not use implicit versions?
- Looks like we are just copying members
- A: pointers (mwahahaha!)
- When class members hold memory, copying gets trickier

Read!

- Data Structures 12.2
  - Which is Hashing
  - Next homework will cover it!