CS100J 28 September 2006 Casting About

1. Casting between classes
2. Apparent and real classes.
3. Operator **instanceof**
4. The class hierarchy
5. function equals

Study Secs 4.2 and 4.3 in text

After today, you have learned all the basics of classes, and done extremely well. Be proud of yourselves.

Procrastination

Leave nothing for to-morrow that can be done to-day. Lincoln

How does a project get a year behind schedule? One day at a time.

Fred Brooks

I don't wait for moods. You accomplish nothing if you do that. Your mind must know it has got to get down to work. Pearl S. Buck

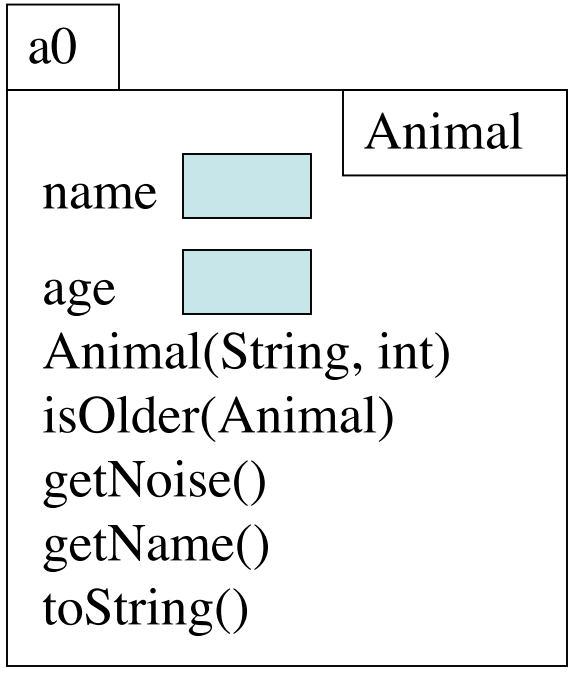
When I start a new project, I procrastinate immediately so that I have more time to catch up. Gries

Buy a poster with the procrastinator's creed here:
http://www.art.com/asp/sp-asp/_/pd--10001845/Procrastinators_Creed.htm

Class Animal

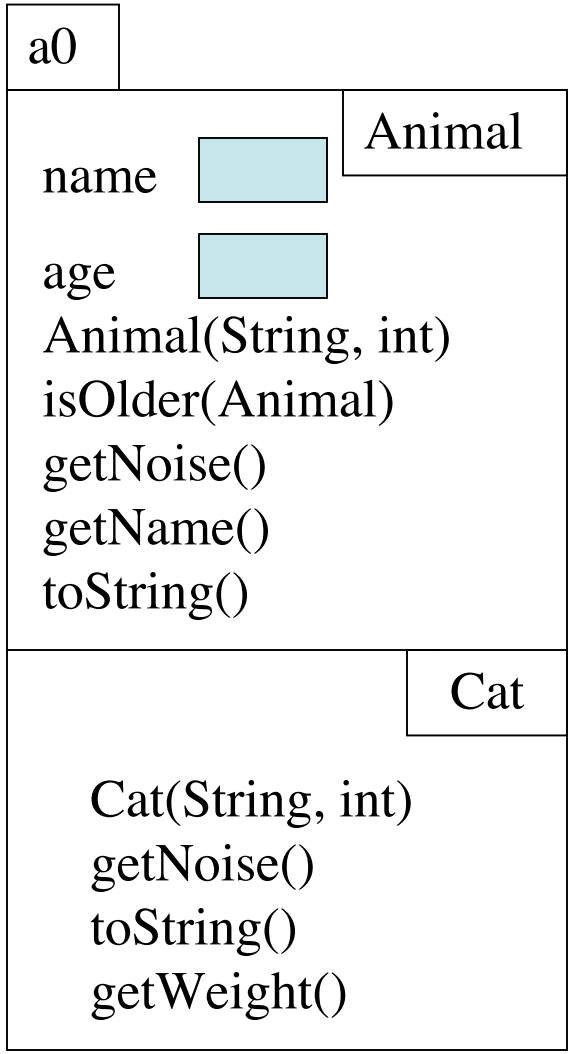
We put each method on one line to save space on the slide. Don't do it in your program.

```
public class Animal {
   private String name;  // name of the animal
   private int age;         // age of animal

   /** Constructor: an Animal with name n, age a  */
   public Animal(String n, int a) { name= n; age= a; }

   /** = "this Animal is older than h" */
   public boolean isOlder(Animal h)
     { return this.age > h.age; }

   /** = the noise that the animal makes --
          "" in class Animal */
   public String getNoise () { return ""; }

   /** = the name of this Animal */
   public String getName() { return name; }

   /** = a description of this Animal */
   public String toString() { return "Animal "  + name + ", age " + age; }
}
```



Class Cat

```
/** An instance is a cat */
public class Cat extends Animal {
   /** Constructor: a Cat with name n and age a */
   public Cat(String n, int a) { super(n, a); }

   /** = the noise this cat makes */
   public String getNoise() { return "meow"; }

   /** = a description of this Cat */
   public String toString() {
      return super.toString() + ", noise " + getNoise();
   }

   /** = weight of Cat */
   public int getWeight() { return 20; }
}
```



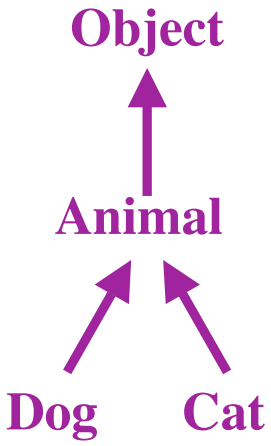
Casting up the class hierarchy

You know about casts like

(**int**) (5.0 / 7.5)

(**double**) 6

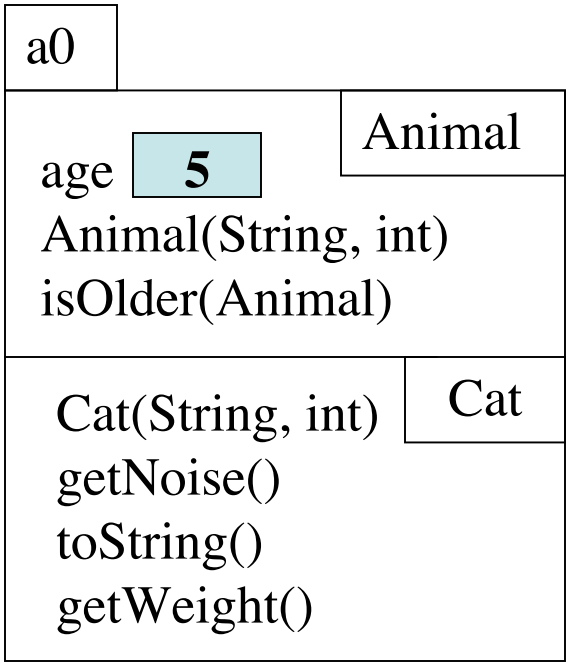
double d= 5; // automatic cast

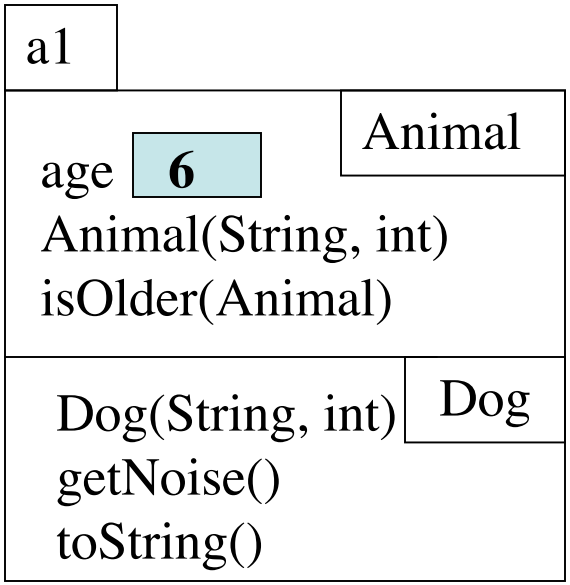


We now discuss casts up and down the class hierarchy.

Animal h= new Cat("N", 5);

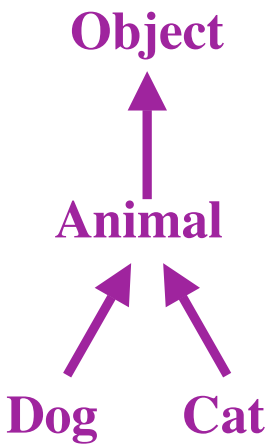
Cat c= (Cat) h;





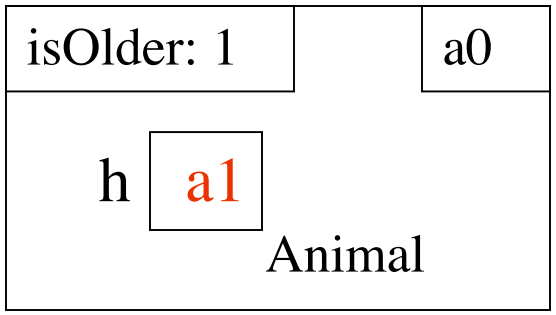
Implicit casting up the class hierarchy

```
public class Animal {

   /** = "this is older than h" */
   public boolean isOlder(Animal h)
     { return this.age > h.age; }
}
```



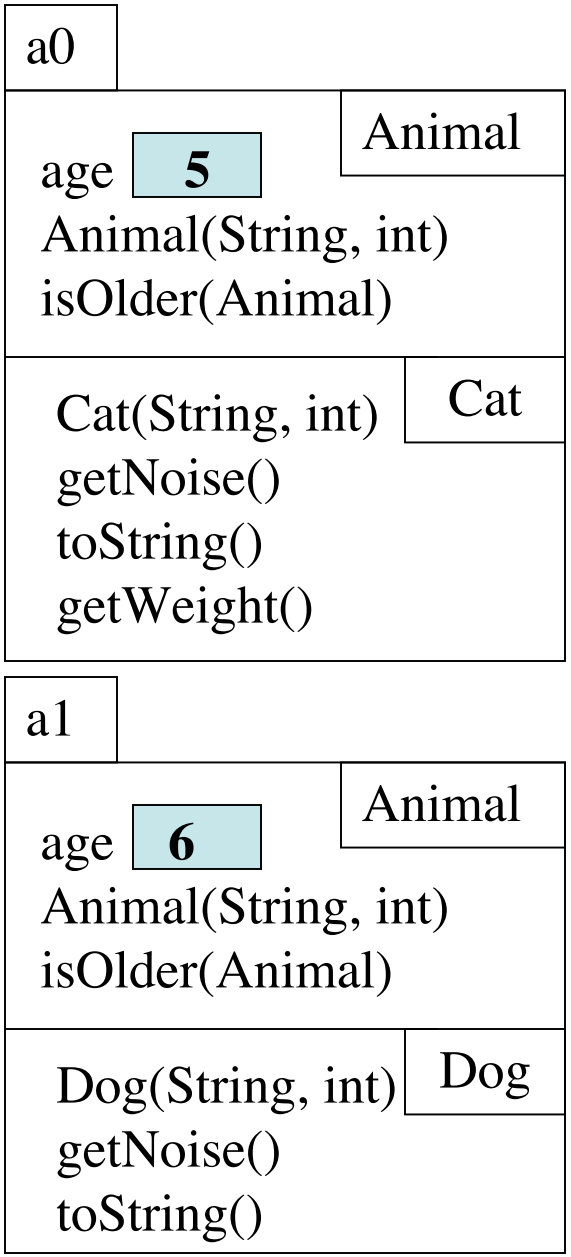

```
c= new Cat("C", 5);
d= new Dog("D", 6);
c.isOlder(d)    ?????
```

Casts up the hierarachy done automatically



Upward automatic casts make sense. Here, any Dog is an Animal

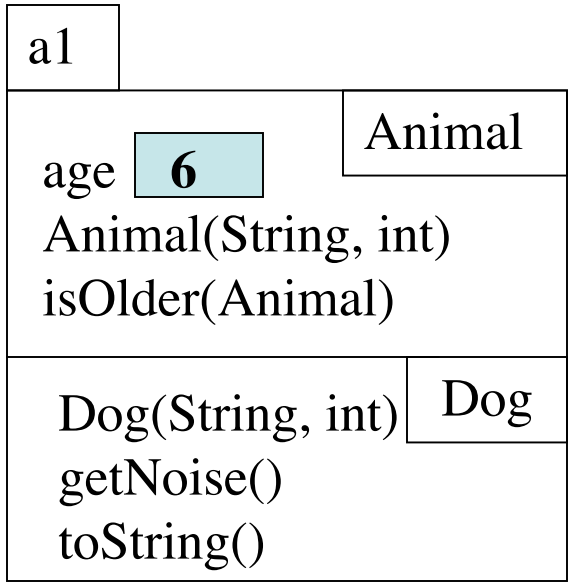
a1 is cast from Dog to Animal, automatically



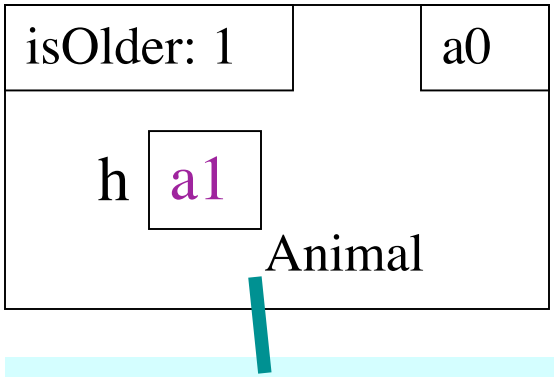
Implicit casting up the class hierarchy

```
public class Animal {
   /** = "this is older than h" */
   public boolean isOlder(Animal h)
     { return this.age > h.age; }
}
```

Two new terms to learn!




```
c= new Cat("C", 5);

d= new Dog("D", 6);

c.isOlder(d)    --what is its value?
```



Apparent type of h. **Syntactic property**. The type with which h is defined.

Real type of h: Doc (type of object a1).

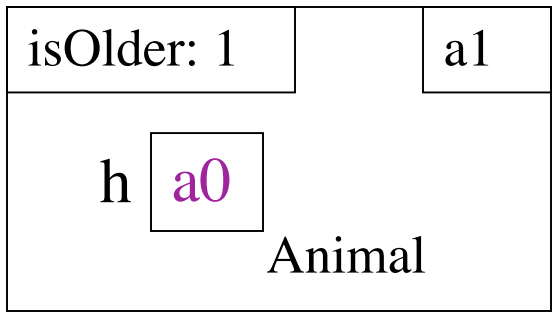
Semantic property. The class-type of the folder whose name is currently in h.

Apparently, h is an Animal, but **really**, it's a Dog.

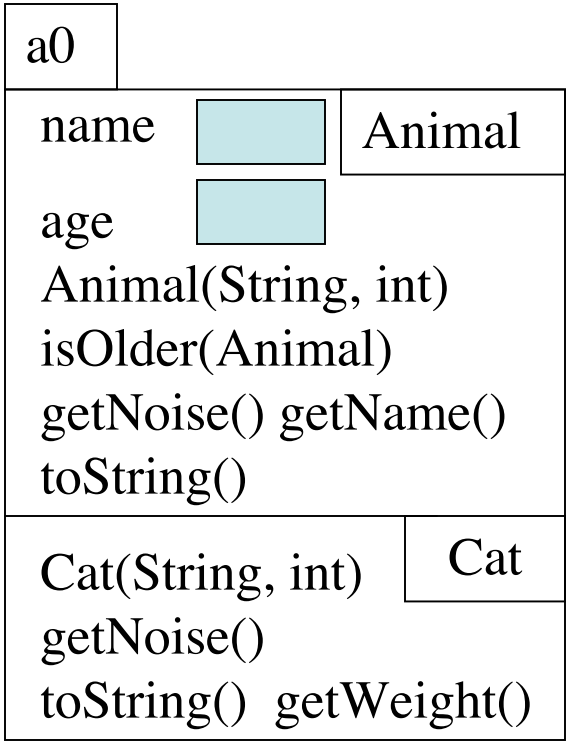
What components can h reference?

```
public class Animal {
   /** = "this is older than h" */
   public boolean isOlder(Animal h)
     { return this.age > h.age; }
}
```

```
c= new Cat("C", 5);
d= new Dog("D", 6);
d.isOlder(c)
```



Apparent type of h: Animal
Real type of h: Cat



What can isOlder reference in object h?

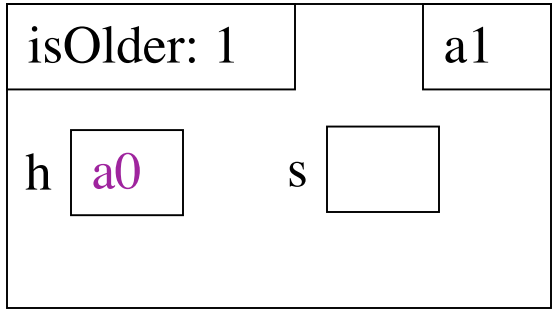
Determined by the apparent type:
Only components in partition Animal (and above)!!!

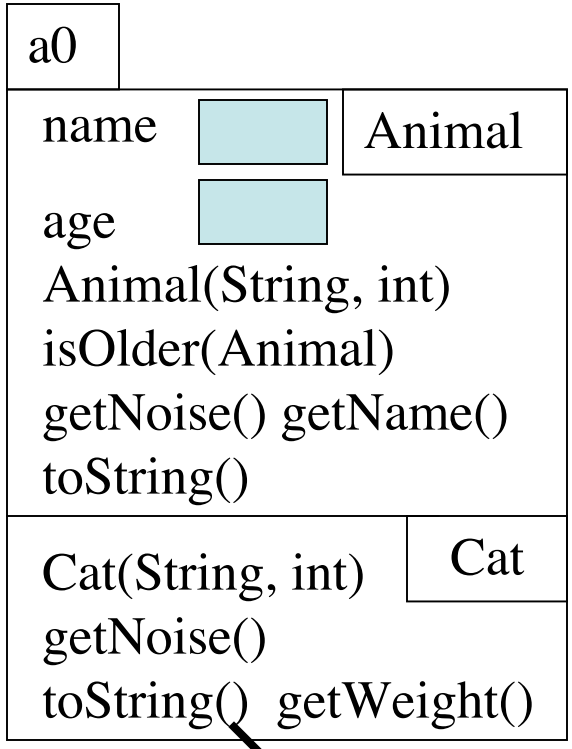
h.getWeight() is illegal. Syntax error.

What method is called by h.toString() ?

```
public class Animal {
    public boolean isOlder(Animal h) {
        String s= h.toString();
        return this.age > h.age;
} }
```

```
c= new Cat("C", 5);
d= new Dog("D", 6);
d.isOlder(c)
```





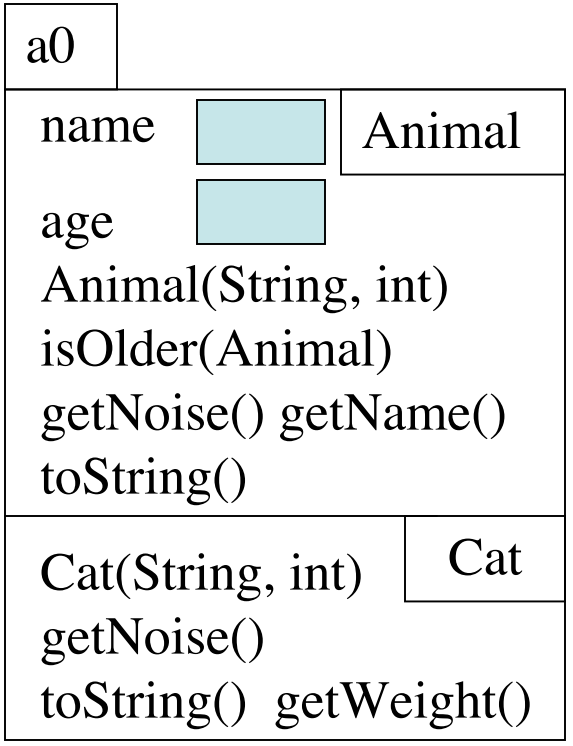
Determined by the real type:
The overriding toString() in Cat.

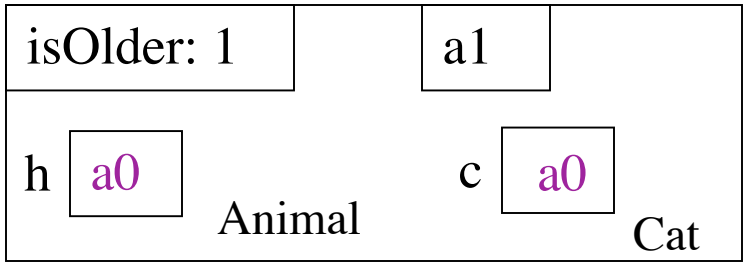
Apparent type of h: Animal
Real type of h: Cat

What method is called by h.toString() ?

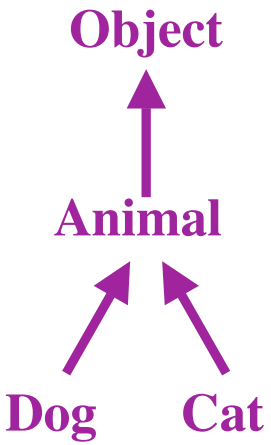
Explicit cast down the hierarchy

```
public class Animal {
      // If Animal is a cat, return its weight;
          otherwise, return 0.
      public int checkWeight(Animal h) {
        if ( ! (h instanceof Cat)  )
              return 0;
        // h is a Cat
        int c= (Cat) h ;       // downward cast
        return c.getWeight();
}
```





Apparent type of h: Animal
Real type of h: Cat



Here, **(Dog) h** would lead to a runtime error.

Don't try to cast an object to something that it is not!

The correct way to write method equals

```
public class Animal {
    /** = "h is a non-null Animal with the same
            values in its fields as this Animal */
    public boolean equals (Object h) {
      if (h == null)  return false;
      if (!(h instanceof Animal)) return false;
      Animal ob= (Animal) h;
      return this.name.equals(ob.name)   &&
                this.age == ob.age;
}
```