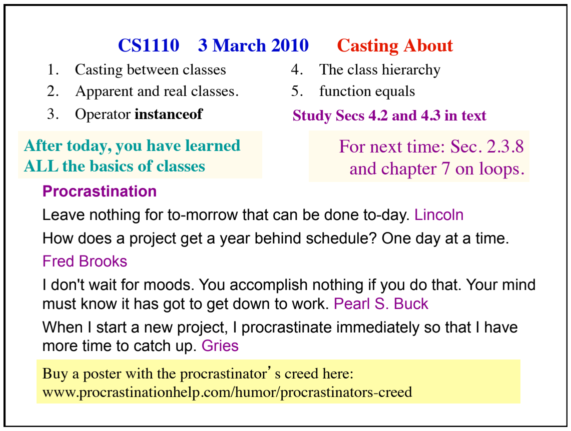


CS1110 3 March 2010 Casting About

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

For next time: Sec. 2.3.8 and chapter 7 on loops.

Procrastination

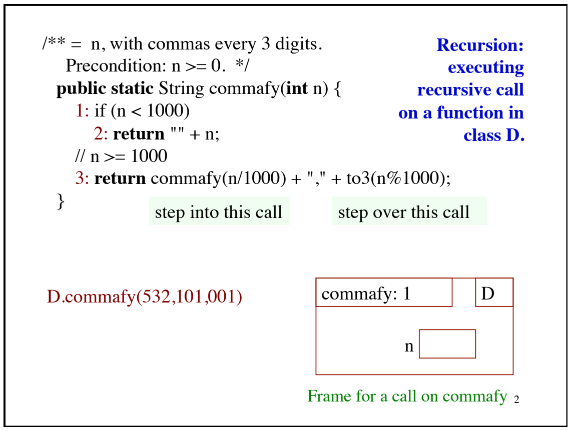
Leave nothing for to-morrow that can be done to-day. Lincoln

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:
www.procrastinationhelp.com/humor/procrastinators-creed



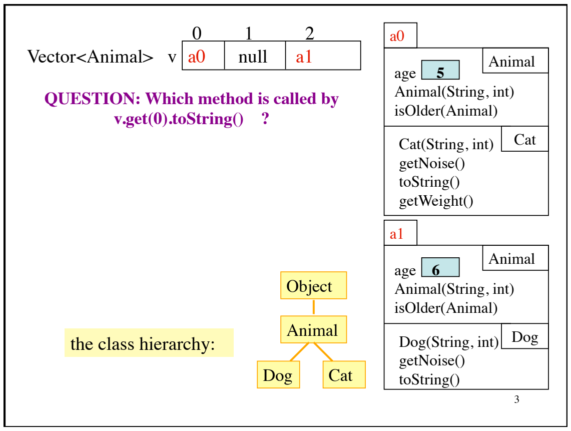
Recursion: executing recursive call on a function in class D.

```
/** =  n, with commas every 3 digits.
    Precondition: n >= 0.  */
  public static String commafy(int n) {
     1: if (n < 1000)
          2: return "" + n;
       // n >= 1000
       3: return commafy(n/1000) + "," + to3(n%1000);
  }
```

step into this call   step over this call

D.commafy(532,101,001)

Frame for a call on commafy 2



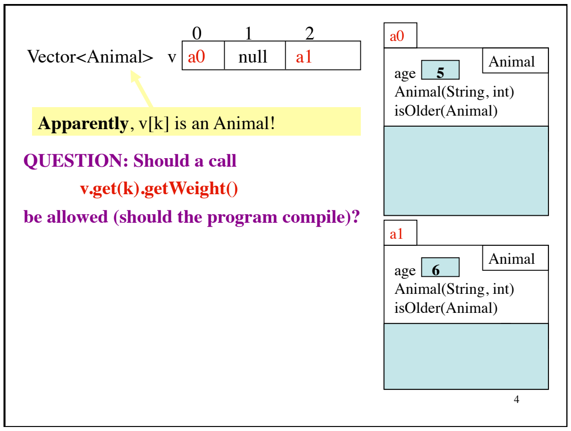
Vector<Animal> v

0	1	2
a0	null	a1

QUESTION: Which method is called by v.get(0).toString() ?

the class hierarchy:

3



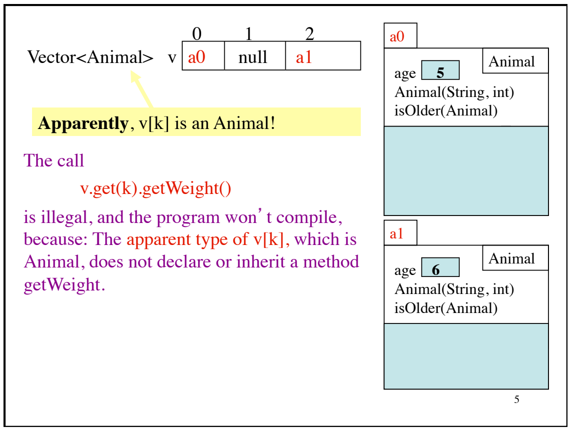
Vector<Animal> v

0	1	2
a0	null	a1

Apparently, v[k] is an Animal!

QUESTION: Should a call v.get(k).getWeight() be allowed (should the program compile)?

4



Vector<Animal> v

0	1	2
a0	null	a1

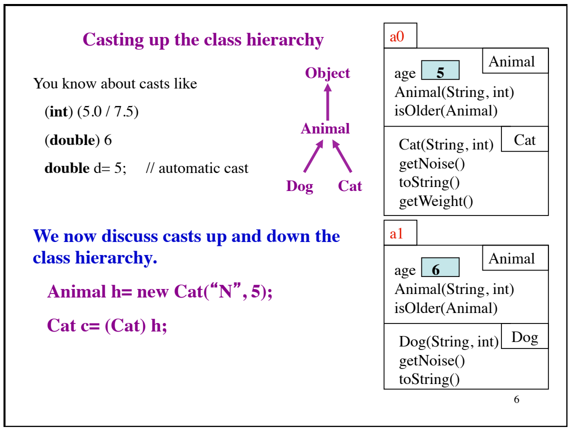
Apparently, v[k] is an Animal!

The call

v.get(k).getWeight()

is illegal, and the program won't compile, because: The apparent type of v[k], which is Animal, does not declare or inherit a method getWeight.

5



Casting up the class hierarchy

You know about casts like

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

(**double**) 6

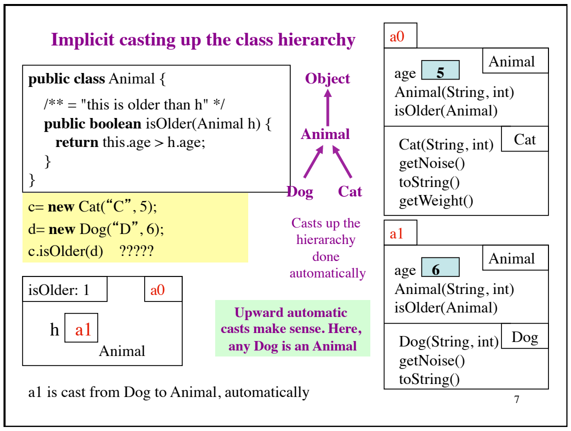
double d= 5; // automatic cast

We now discuss casts up and down the class hierarchy.

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

Cat c= (Cat) h;

6



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)    ?????
```

isOlder: 1 a0

h a1 Animal

Casts up the hierarachy done automatically

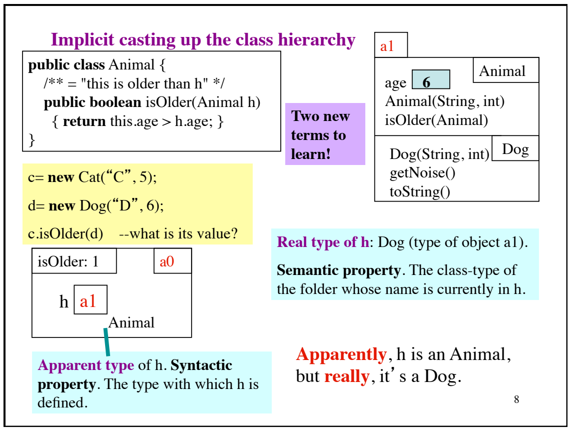
Upward automatic casts make sense. Here, any Dog is an Animal

a0
Animal: age 5, Animal(String, int), isOlder(Animal)
Cat: Cat(String, int), getNoise(), toString(), getWeight()

a1
Animal: age 6, Animal(String, int), isOlder(Animal)
Dog: Dog(String, int), getNoise(), toString()

a1 is cast from Dog to Animal, automatically

7



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!

a1
Animal: age 6, Animal(String, int), isOlder(Animal)
Dog: Dog(String, int), getNoise(), toString()

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

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

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

isOlder: 1 a0

h a1 Animal

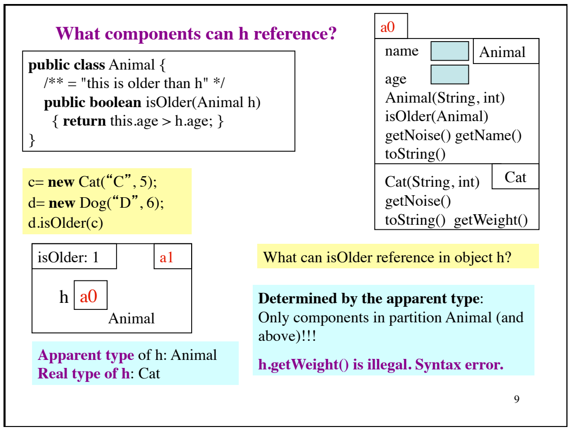
Real type of h: Dog (type of object a1).

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

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

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

8



What components can h reference?

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

a0
Animal: name, age, Animal(String, int), isOlder(Animal), getNoise() getName(), toString()
Cat: Cat(String, int), getNoise(), toString() getWeight()

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

isOlder: 1 a1

h a0 Animal

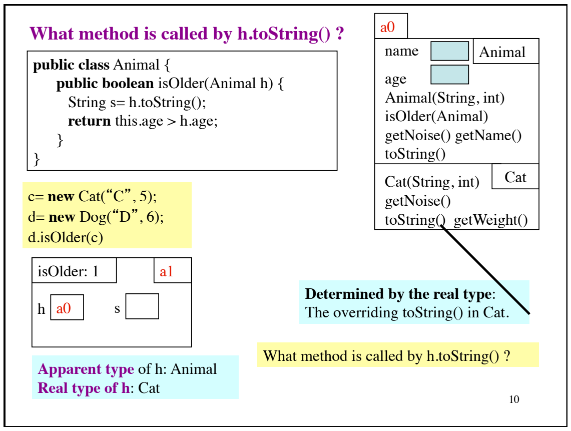
What can isOlder reference in object h?

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

h.getWeight() is illegal. Syntax error.

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

9



What method is called by h.toString() ?

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

a0
Animal: name, age, Animal(String, int), isOlder(Animal), getNoise() getName(), toString()
Cat: Cat(String, int), getNoise(), toString() getWeight()

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

isOlder: 1 a1

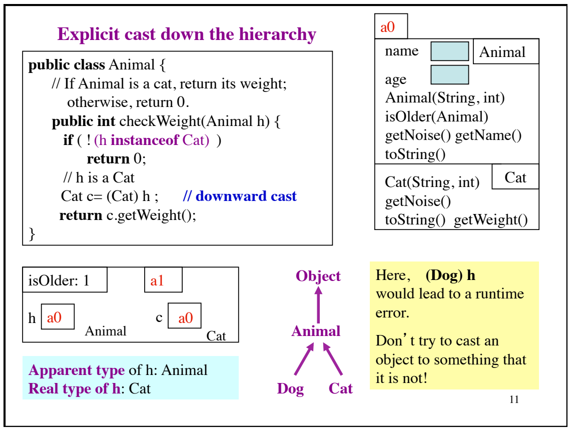
h a0 s

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

What method is called by h.toString() ?

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

10



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
        Cat c= (Cat) h ;      // downward cast
        return c.getWeight();
}
```

a0
Animal: name, age, Animal(String, int), isOlder(Animal), getNoise() getName(), toString()
Cat: Cat(String, int), getNoise(), toString() getWeight()

isOlder: 1 a1

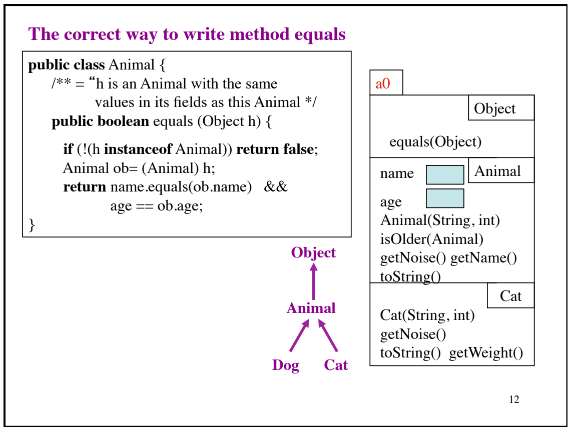
h a0 Animal c a0 Cat

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

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

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

11



The correct way to write method equals

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

a0
Object: equals(Object)
Animal: name, age, Animal(String, int), isOlder(Animal), getNoise() getName(), toString()
Cat: Cat(String, int), getNoise(), toString() getWeight()

12