

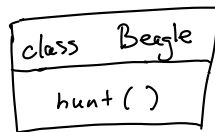
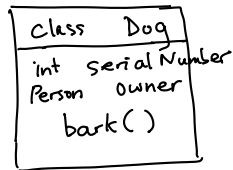
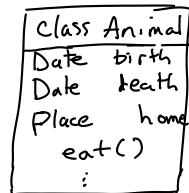
# lecture 28

## inheritance

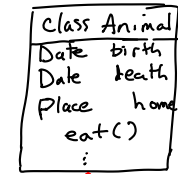
All beagles are dogs.  
All dogs are animals.

Dogs bark.  
Beagles chase rabbits.  
Animals are born (date).

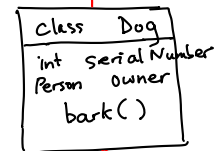
### Class Hierarchy



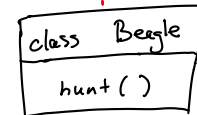
### Inheritance Diagram



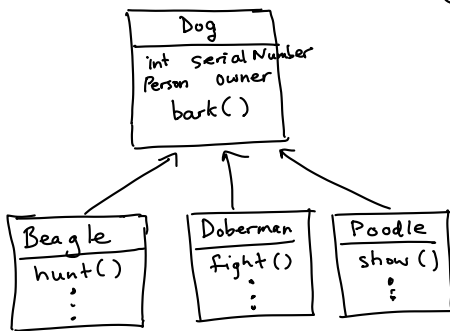
↑ extends



↑ extends



### Inheritance Terminology



Beagle is a subclass of Dog.

Dog is a superclass of Beagle.

Beagle.bark() overrides Dog.bark() for Beagle objects.

Subclasses inherit  
the fields and methods  
of their superclass.

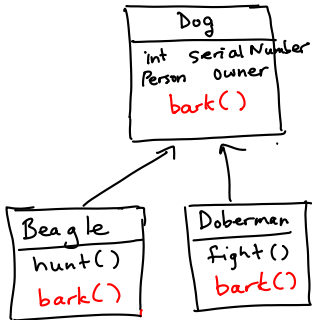
## Overloading (review)

- same method name but **different parameter types and/or number**, and optionally different return type
- **signature** - name + formal parameters + return type
- can occur within a class or between classes

## Overriding a method

- subclass method **overrides** superclass method
- the **signatures are the same** (name + parameter types). otherwise it is overloading.

### Example: (overriding)



```

class Animal {
    ...
}
class Dog extends Animal {
    void bark() {
        print("woof")
    }
}
class Beagle extends Dog {
    void bark() {
        print("aa woowoooo")
    }
}
class Doberman extends Dog {
    void bark() {
        print("GRRR.. wo wo!")
    }
}
  
```

## Object class

```

class Object {
    boolean equals()
    int hashCode()
    Object clone()
    String toString()
}
  
```

extends ↑ (automatic)

```

class Animal {
    Date birth
    Date death
    Place home
    eat()
    ...
}
  
```

extends

```

class Dog {
    int serialNumber
    Person owner
    bark()
}
  
```

extends

```

class Beagle {
    hunt()
    bark()
}
  
```

```

class Object {
    boolean equals()
    int hashCode()
    Object clone()
    String toString()
}
  
```

## Object.equals()

```
Object obj1, obj2
```

⋮

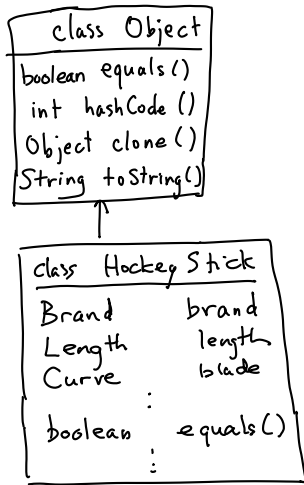
```
if (obj1.equals(obj2))
```

⋮

// true if and only if obj1 and obj2  
// reference the same object

// i.e. obj1 == obj2

When to override equals() ?



When are two hockey sticks equal ?

### String.equals()

String.equals() overrides Object.equals()  
WARNING: careful when comparing strings using ==

```

String s1 = "hello"
String s2 = "hello"
// s1 == s2 is true

String s3 = new String("hello")
// s1 == s3 is false
// s1.equals(s3) is true
  
```

See more examples in lecture notes.

### Recall lecture 22 (Interfaces)

If a class implements Comparable, then it is recommended that:

a1.compareTo(a2) returns 0 if and only if

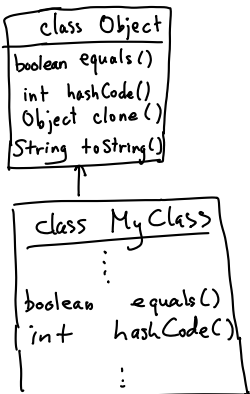
a1.equals(a2) returns true.

### Object.hashCode()

You sometimes read that this returns the memory address of the object. However, this is not quite true. Memory addresses are 32 bits. But Object.hashCode() is a 24-bit number. Thus, at most  $2^{24}$  objects. Why the restriction?

Memory is partitioned into various reserved regions. (including also the call stack, and instructions).

### Overriding equals() and hashCode()



MyClass x,y :

x.equals(y) is true

✓ ↓ ~~✗~~ not necessarily

x.hashCode() == y.hashCode() is true

### Object.clone()

Make a new object that has the same fields as the original.

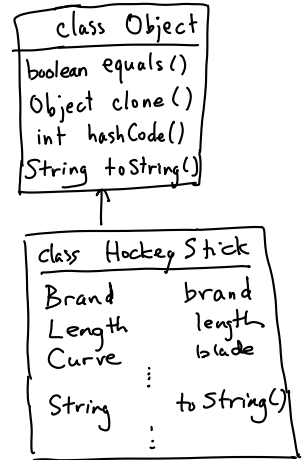
When you override this method (eg. HockeyStick class) you should explicitly copy all fields, and ensure:

- x.clone() == x is false
- x.equals(x.clone()) is true

## Object.toString()

- prints the class name  
+ the hash code  
(in hexadecimal, see Exercises 1)
- *eg. Doberman @ 7d34fa*
- Often you want to override this method  
(print out various fields of your class)

Example:  
overriding  
toString()



x.toString() returns "Bauer, Vapour, 60 in, composite  
....."