All dogs are animals.  
All beagles are dogs.

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

Inheritance

```
Class Animal
  Date birth
  Date death
  Place home
  eat()

Class Dog
  int serialNumber
  Person owner
  bark()

Class Beagle
  hunt()
```

Beagle is a subclass of Dog.  
Dog is a superclass of Beagle.

Subclasses inherit the fields and methods of their superclass.

Overloading

- same method name but different parameter types and/or number, (return type can be same or different)
- method signature - name + formal parameters + return type
- overloading can occur within a class or between classes
class Animal {
    Place home;
    Animal() { ... }
    Animal(Place home) {
        this.home = home;
    }
}

class Dog extends Animal {
    String owner;
    Dog() { ... }  // automatically calls super() which creates fields inherited from superclass i.e. Animal
    Dog(Place home, String owner) {
        super(home);
        this.owner = owner;
    }
    void bark(){
        System.out.println("Woof!");
    }
}

Example: (overriding)

```
// More methods...
```

Constructors are not inherited.

(Each object belongs to a unique class which determined by the constructor when the object is constructed.)

Object class

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

Constructors chaining
class Object

    boolean equals(Object)
    int hashCode()
    Object clone()
    String toString()

Object.equals(object)

    Object obj1, obj2
    :
    if (obj1.equals(obj2)) :
    // true if and only if obj1 and obj2 refer to the same object
    // i.e. obj1 == obj2

String.equals(String)

WARNING: Careful when comparing strings using ==

    String s1 = "hello"
    String s2 = "hello"
    // s1 == s2 is true
    String s3 = new String("hello")
    // s1.equals(s3) is true
    // s1 == s3 is false

See more examples in lecture notes.

Recall lecture 22 (Interfaces)

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

    al. compareTo(a2) returns 0 if and only if
    al. equals(a2) returns true.

class Object

- hashCode() returns an int (often 24 bits)
- typically, two distinct objects return two distinct hashcodes (since typical implementation is to use the starting address of object in JVM memory)
Object.clone()

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

If you override this method
eg. HockeyStick.clone()
then you should require:
• x.clone() == x is false
• x.equals(x.clone()) is true

Object.toString()

• prints the class name and hash code of object (in hexadecimal)
  eg. Doberman => 7434f9

x.toString()

returns "Bauer, Vapour, 60 in, composite"

ADVANCED TOPIC:
not on final exam
Classes vs. Objects
(descriptors vs. instances)

Class descriptors are objects!
They are instances of the class Class.
(In my opinions, it would have been better to call it Class Descriptor)