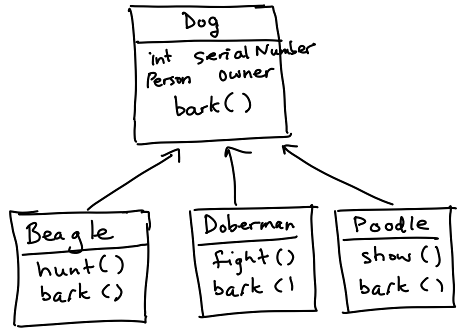
lecture 29

- inheritance (cont.)
- polymorphism

Inheritance



Beagle is a subclass of Dog.

Dog is a superclass of Beagle.

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

General question: when a method is defined in a sub- and super-class (overriding) which version gets applied?

- 'this'
- 'super'

} keywords

Example 1

```
class Dog {
    void showTeeth() {
        print("see these teeth")
    }
    void bark() {
        print("woof")
    }
    void threaten() {
        showTeeth()
        bark()
    }
}
```

↑ which bark?

(Hint: compiler substitutes 'this')

```
class Doberman extends Dog {
    void bark() {
        print("GRRR, WOOF")
    }
}
```

```
class Test {
    main() {
        Doberman dober =
            new Doberman()
        dober.threaten()
    }
}
```

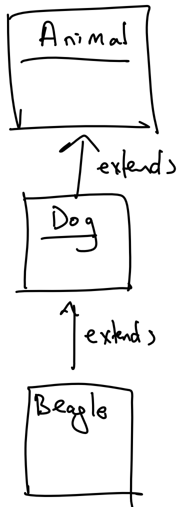
```
class Dog {
    void bark() {
        print("woof")
    }
    void showTeeth() {
        print("see these teeth")
    }
    void threaten() {
        this.showTeeth()
        this.bark()
    }
}
```

```
class Doberman extends Dog {
    void bark() {
        print("GRRR WOOF")
    }
    void barkLikeDog() {
        super.bark()
    }
}
```

'this' is a reference to the object that invokes the method (determined at run time).

'super' is a reference to a class.

Constructors cannot be inherited.
(Why not?)



Example 2 ("constructor chaining")

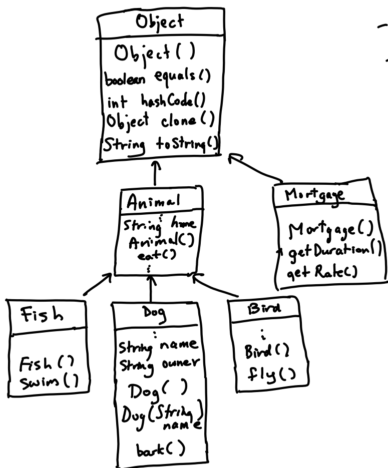
```
class Animal {
    Place home
    Animal() { }
    Animal(Place home){
        this. home = home
    }
}
class Dog extends Animal {
    :
    Dog(){// super();}  // automatic (unnecessary)  invokes
                        Animal's default constructor

    Dog(Place home, String owner){
        super (home);
        this.owner = owner;
    }
}
```

Classes vs. Objects

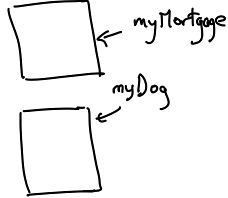
(descriptors vs. instances)

Class descriptors

- class name
- list of fields / types
- methods (names + code)
- reference to superclass
- static fields

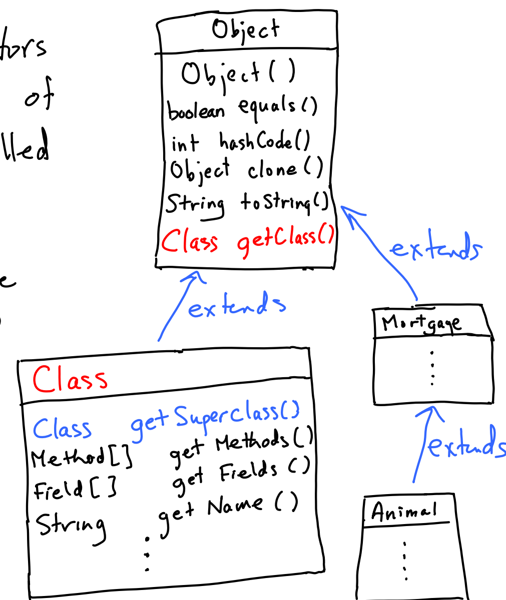


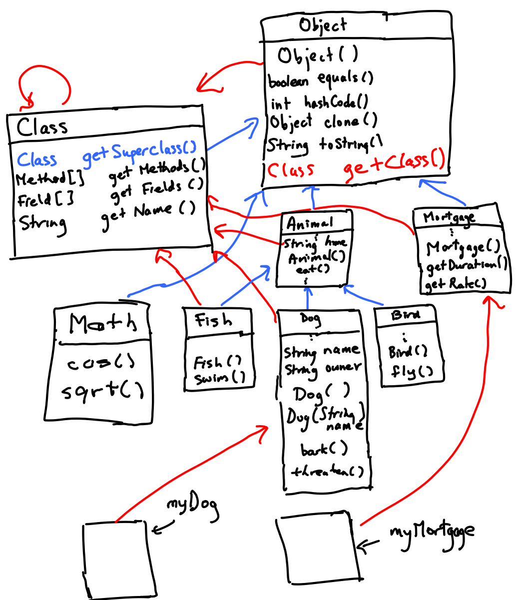
Object instances



Class descriptors are instances of a class called **Class**.

(It would have been better to call it Class Descriptor)





getClass (instance of)

getSuperClass (extends)

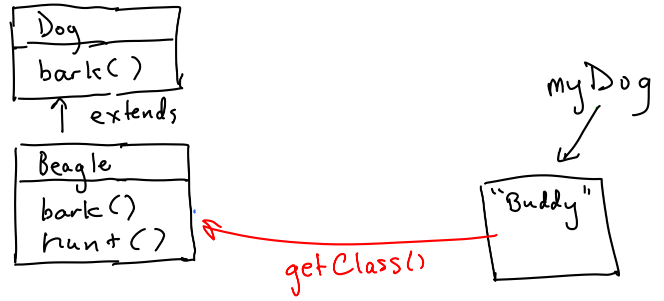
object instances

T.A.s often use the Class class to automate grading.

e.g. getMethods() and then test them

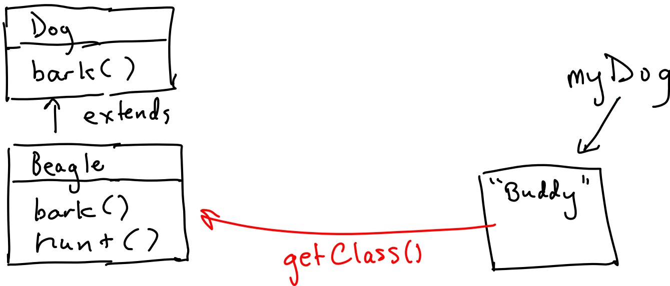
(Also see A3.TestSignatures.)

Compile time vs. Run time



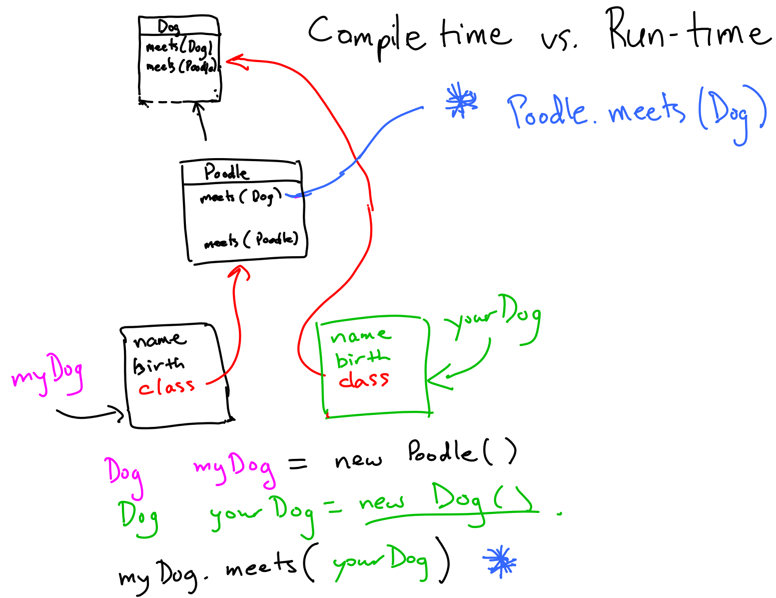
Dog myDog = new Beagle("Buddy")

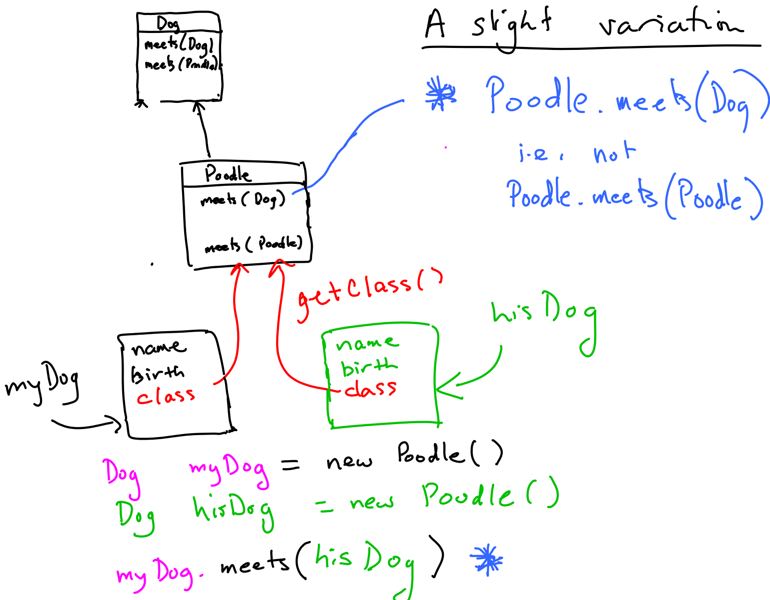
Does compiler allow this? (yes)



Dog myDog = new Beagle("Buddy")
myDog.hunt()

Does compiler allow this? (No)





"Polymorphism"

A reference variable has a declared type. It can reference any object of that type or subtype i.e. subclass.

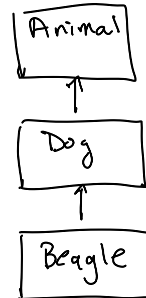
Polymorphism (cont.)

A method name in an instruction will cause an action that is defined by the class of object that invokes the method.

Often called "dynamic dispatch"
or
"dynamic binding"
(of the method to the object)

How to choose the method?

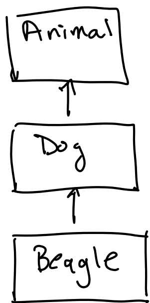
Compile time
- reference variable has a declared type
- check that method signature exists




```
Dog myDog

myDog.bark()
myDog.hashCode()
```

Run time
- reference variable points to an object

Polymorphism - object type (class) can be a subtype (subclass) of reference variable type




```
Dog myDog
myDog = new Beagle()
myDog.bark()
myDog.hashCode()
```