Object-Oriented Programming with ActionScript 3

Kris Schultz

RESOURCEINTERACTIVE

Me

You?

What You'll Learn

- The *what* and *why* of OOP
- Four features of OO languages
- OOP terminology
- How to write OO code in AS3

What is OOP?

- Technique for developing software by constructing small, independent “objects” that collaborate with each other
- Each object is *independent* and has a focused set of *responsibilities*

Benefits

- Enables adaptivity
- Makes complex programs easier to think about
- Promotes code reuse
- Allows programming labor to be distributed among multiple people

Features of OO Languages

- *Modularity* - separation of roles and responsibilities
- *Inheritance* - code reuse
- *Polymorphism* - different behaviors with common name
- *Encapsulation* - hiding implementation details from other objects

Modularity

class

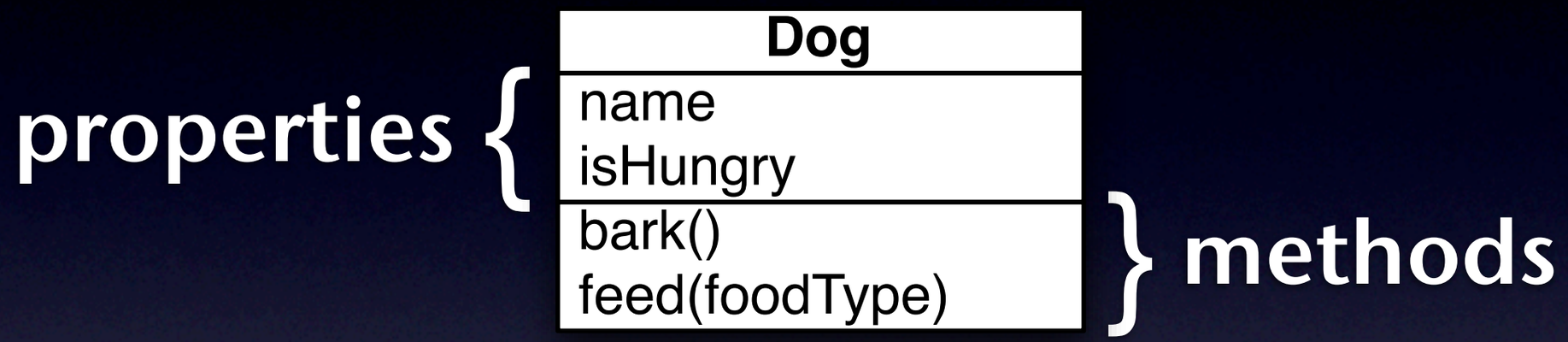
A blueprint that describes an object type.


VERANDAH
M BED
dwn
BRKFST
M
BATH
C
C
M
POW
LIVING
DINING
KIT
VERANDAH
PORTAL
PAN
UTL
M
PLANTER
SPA
dwn
BRIDGE
BREEZEWAY
POOL
DECK
AUTOCOURT
32

House class

instances of the House class

An example class...


properties
Dog
name
isHungry
bark()
feed(foodType)
methods

Writing a class...

```
package animals
{
    public class Dog
    {

        public var name:String = "";

        public var isHungry:Boolean = true;

        public function bark() :void
        {
            ...
        }

        public function feed(foodType:String) :void
        {
            ...
        }
    }
}
```

Creating an instance of the Dog class

```
var myDog:animals.Dog = new animals.Dog();
```

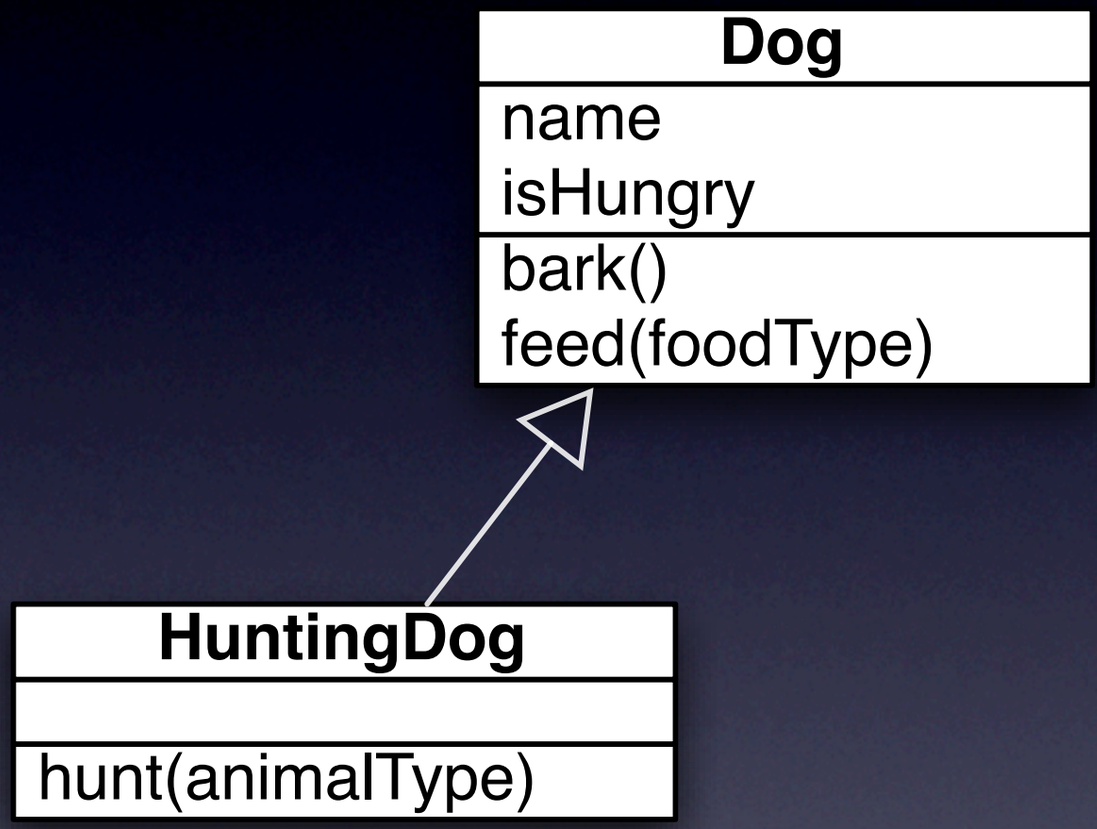
```
import animals.Dog;
var myDog:Dog = new Dog();
```

Demo

Inheritance

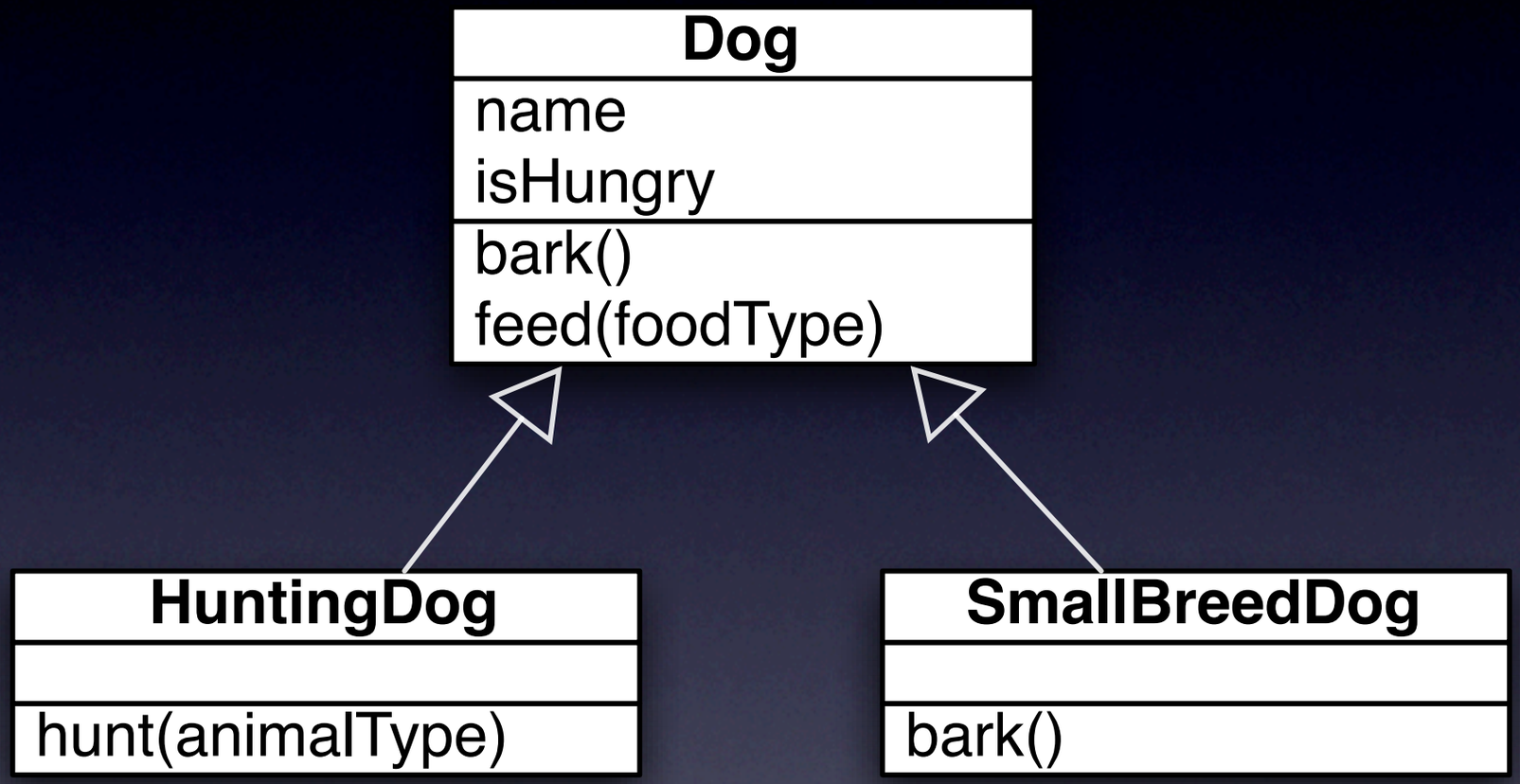
subclass / child class

A class that extends another class and adds to or changes the original's functionality.


Dog
name
isHungry
bark()
feed(foodType)
HuntingDog
hunt(animalType)

Demo

Polymorphism

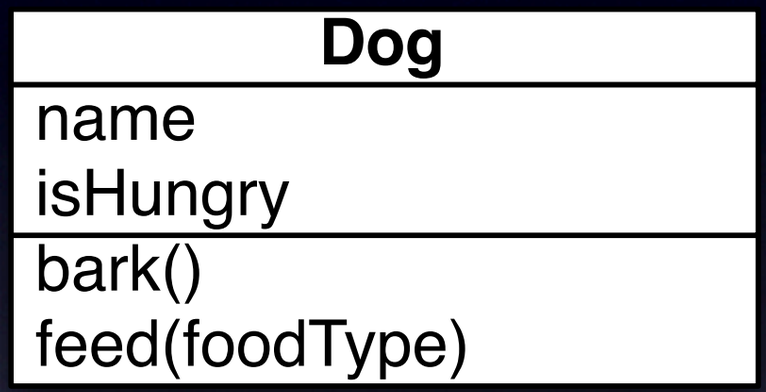

Dog
name
isHungry
bark()
feed(foodType)
HuntingDog
hunt(animalType)
SmallBreedDog
bark()

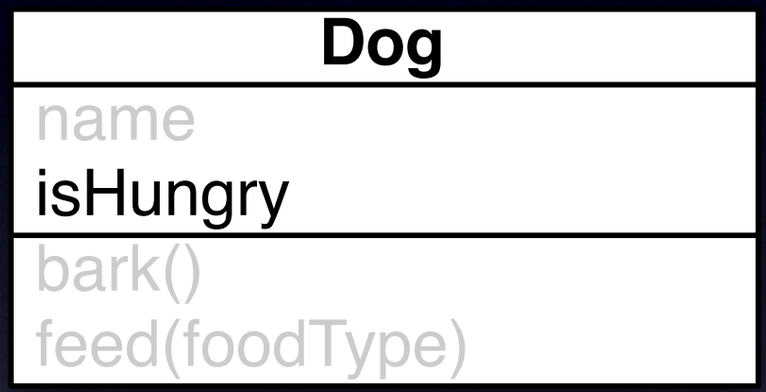
method overriding

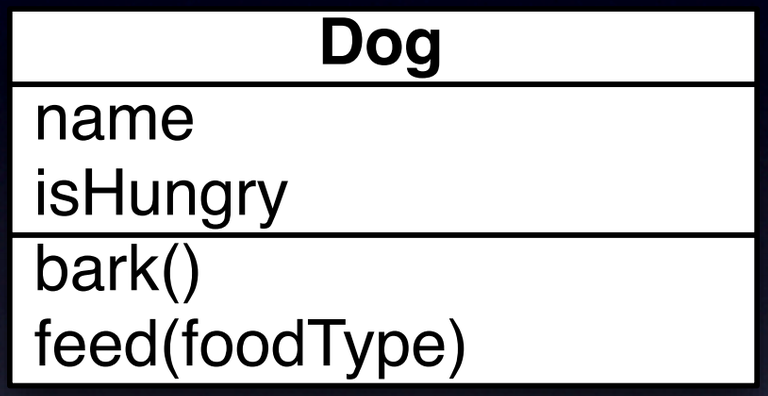
Changing the behavior of an inherited method by rewriting it in the subclass.

Demo

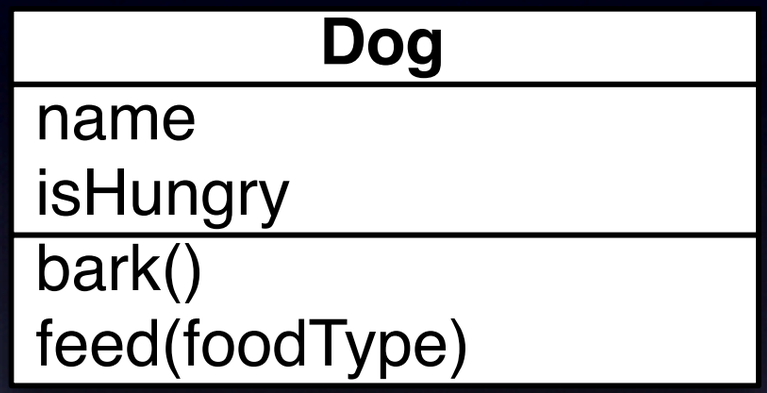
Encapsulation


Dog
name
isHungry
bark()
feed(foodType)


Dog
name
isHungry
bark()
feed(foodType)



What exists



What other objects see

Dog
name lastFedTime
bark() feed(foodType) isHungry()

What exists

Dog
name lastFedTime
bark() feed(foodType) isHungry()

What other objects see

access modifiers

Special keywords that allow you to hide information in one object from the prying eyes of another.

AS 3 Access Modifiers

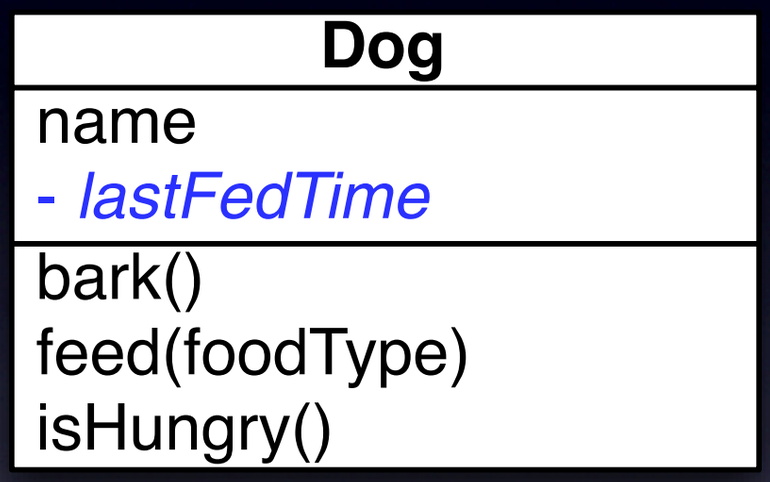
modifier	grants access to...
`public`	any class
`protected`	this class and its subclasses
`internal`	any class in the same package
`private`	no classes but this one

Dog
name lastFedTime
bark() feed(foodType) isHungry()

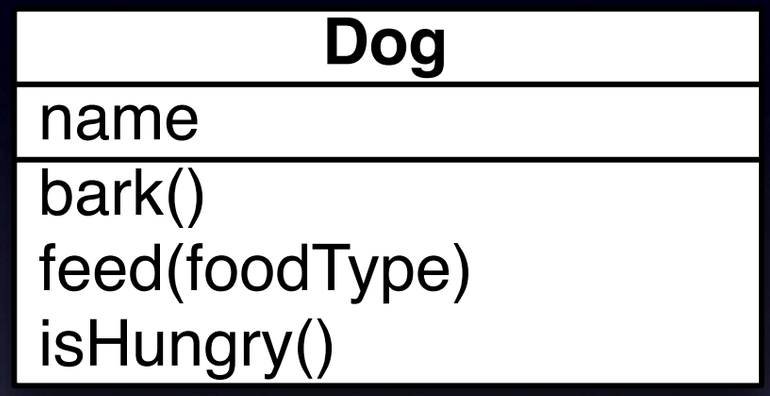
What exists

Dog
name lastFedTime
bark() feed(foodType) isHungry()

What other objects see



What exists



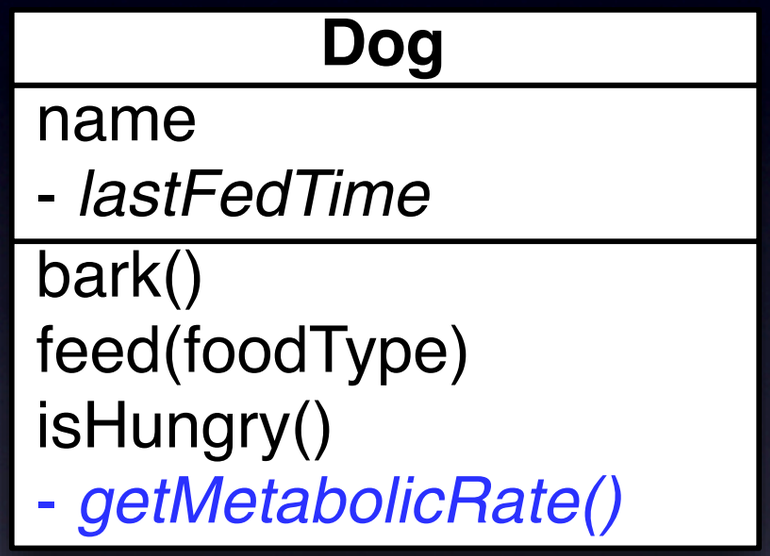
What other objects see

Dog
name - *lastFedTime*
bark() feed(foodType) isHungry() getMetabolicRate()

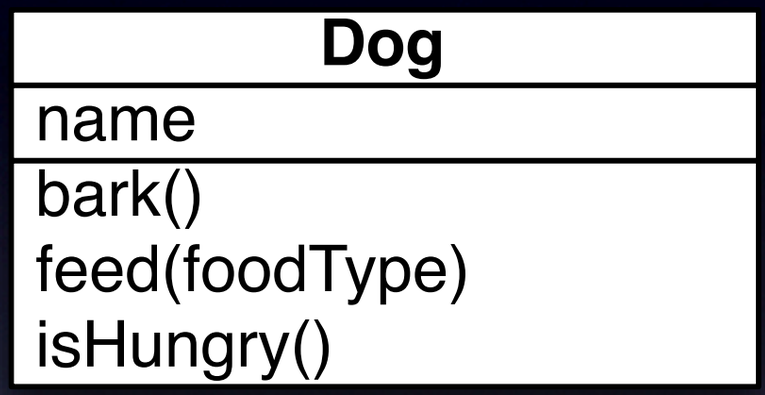
What exists

Dog
name
bark() feed(foodType) isHungry() getMetabolicRate()

What other objects see



What exists



What other objects see

Dog
name - *lastFedTime*
bark() feed(foodType) isHungry() - *getMetabolicRate()*

What exists

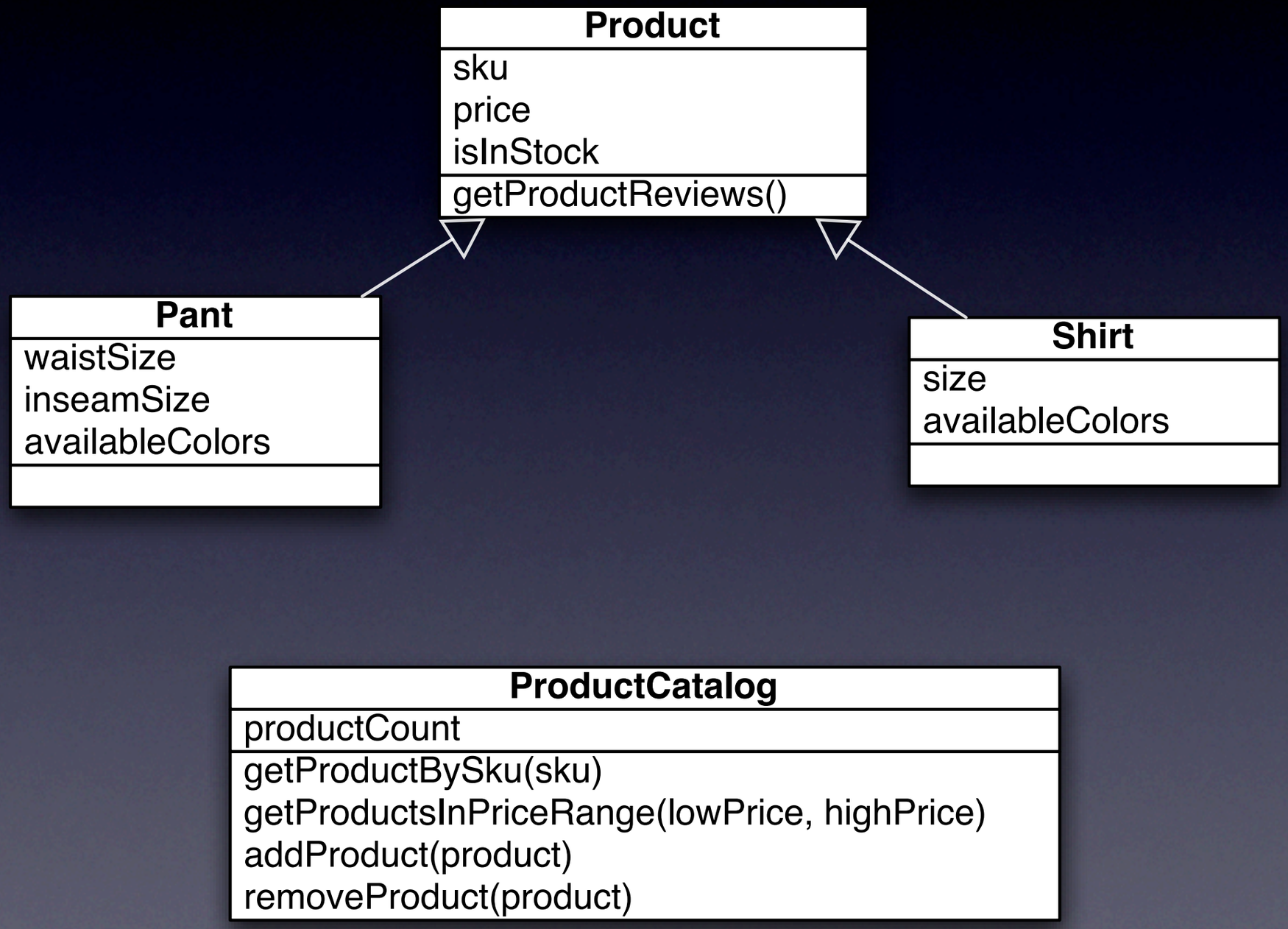
Dog
name
bark() feed(foodType) isHungry()

What other objects see

```
package animals
{
    public class Dog
    {
        public var name:String = "";

        private var lastFedTime:Date;

        public function bark() :void
        {...}

        public function feed(foodType:String) :void
        {...}

        public function isHungry() :Boolean
        {...}

        private function getMetabolicRate() :Number
        {...}
    }
}
```

OOP in “real” applications

Data classes



UI classes

ButtonBar
buttons selectedButton
addButton(button) removeButton(button)

Button
label icon
onPress() onRelease()

Slider
currentValue minValue maxValue
setSize(width, height) addEventListener(event, listener) removeEventListener(event, listener)

ProductView
width height
displayProduct(product) activateZoom() deactivateZoom()

What We Covered

- The *what* and *why* of OOP
- Four features of OO languages
  - Modularity - *class*
  - Inheritance - *subclass*
  - Polymorphism - *method overriding*
  - Encapsulation - *access modifiers*
- ActionScript 3 syntax for OOP

Contact me:

kschultz@resource.com

Replay this presentation:

www.createandgrow.com/presentations/

Join a user group:

www.commug.org (Columbus)

www.seocats.org (Athens)

Read:

"Object Technology: A Manager's Guide"

by David A. Taylor