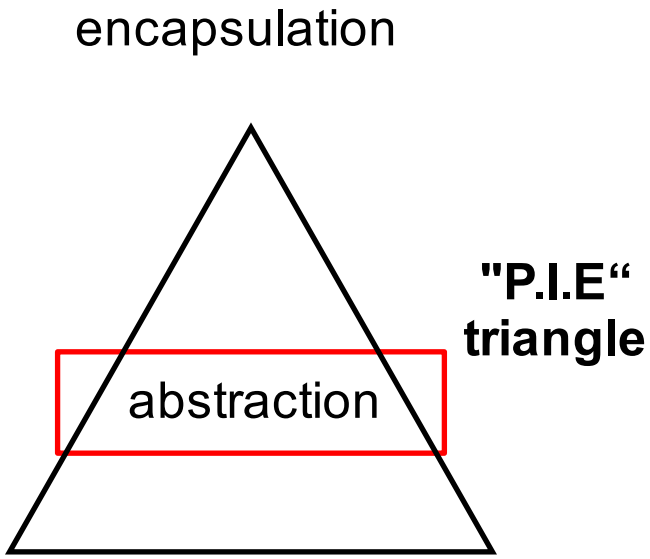
Object-Oriented Programming

Abstraction

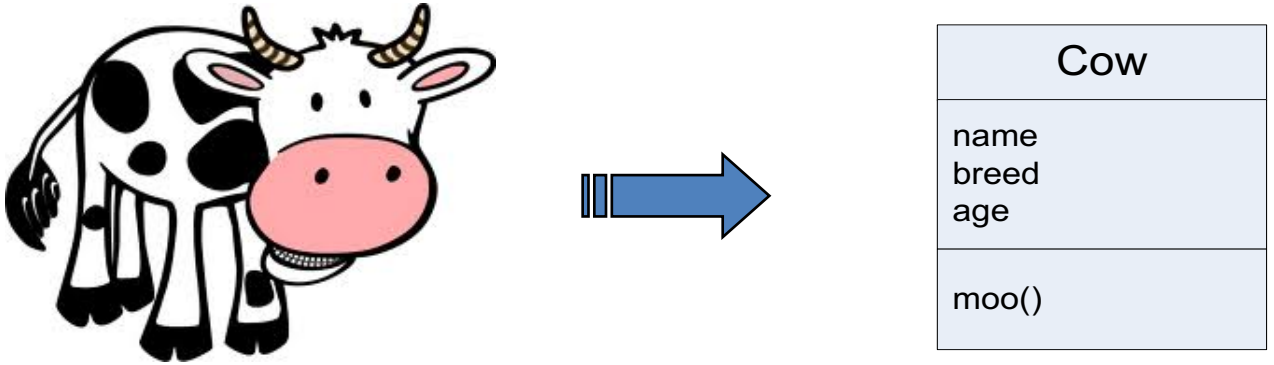
Contents

- Concept of abstraction
- Abstract classes
- Abstract methods

Important OO Concepts



What is Abstraction?



- Abstraction: is the process to simplify a complicated system down to its most fundamental parts and describe these parts in a simple, precise language:
  - naming the parts
  - explaining their functionality

What is Abstraction?

Sue's car:
Fuel: 20 liter
Speed: 0 km/h
License plate: "143 WJT"

Martin's car:
Fuel: 49.2 liter
Speed: 76 km/h
License plate: "947 JST"

Tom's car:
Fuel: 12 liter
Speed: 40 km/h
License plate: "241 NGO"

Automobile:

- fuel
- speed
- license plate

- speed up
- slow down
- stop

Abstraction vs. Inheritance Design

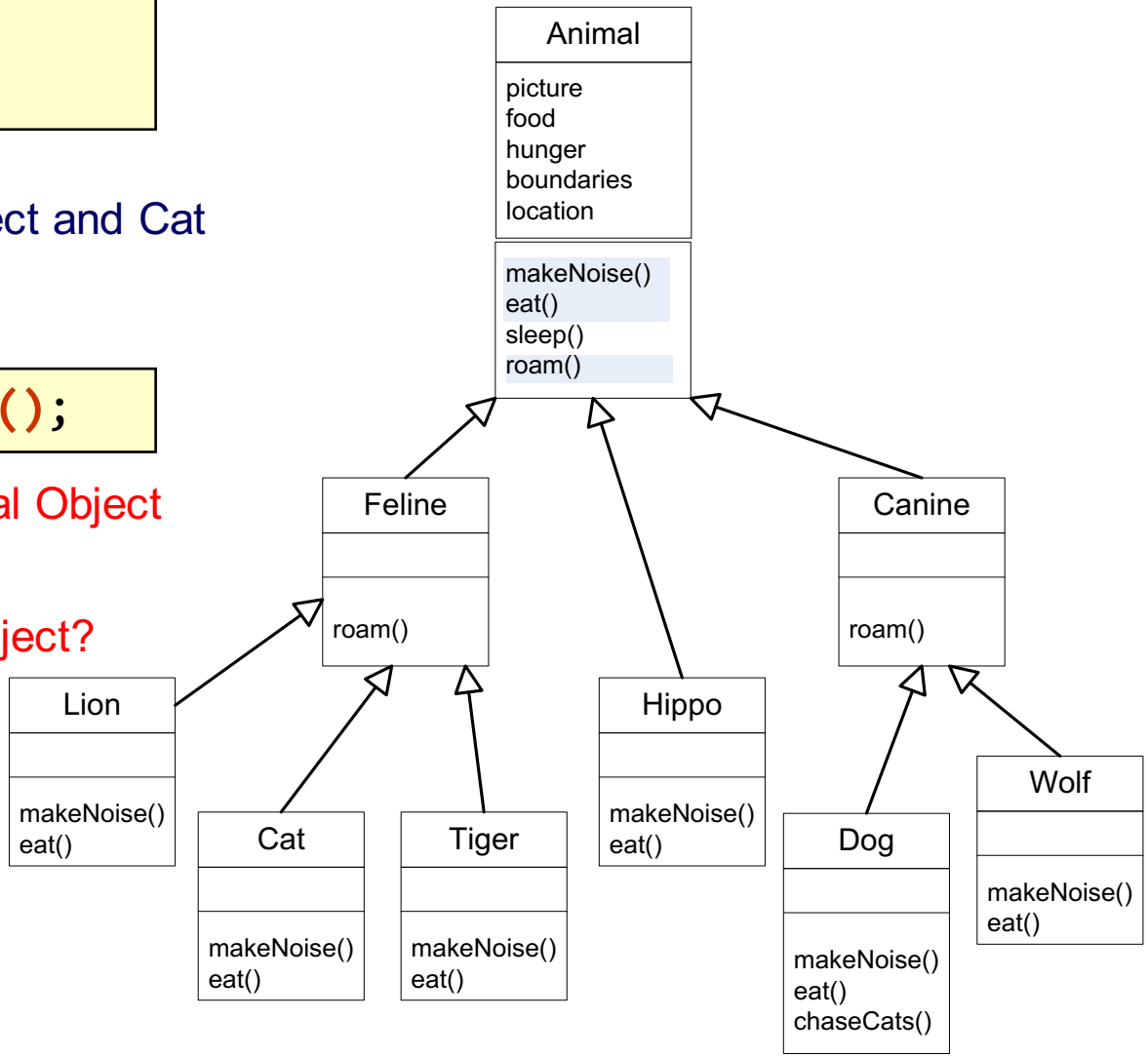
```
Dog d = new Dog();
Cat c = new Cat();
```

→ We can imagine how Dog Object and Cat Object look like

```
Animal anim = new Animal();
```

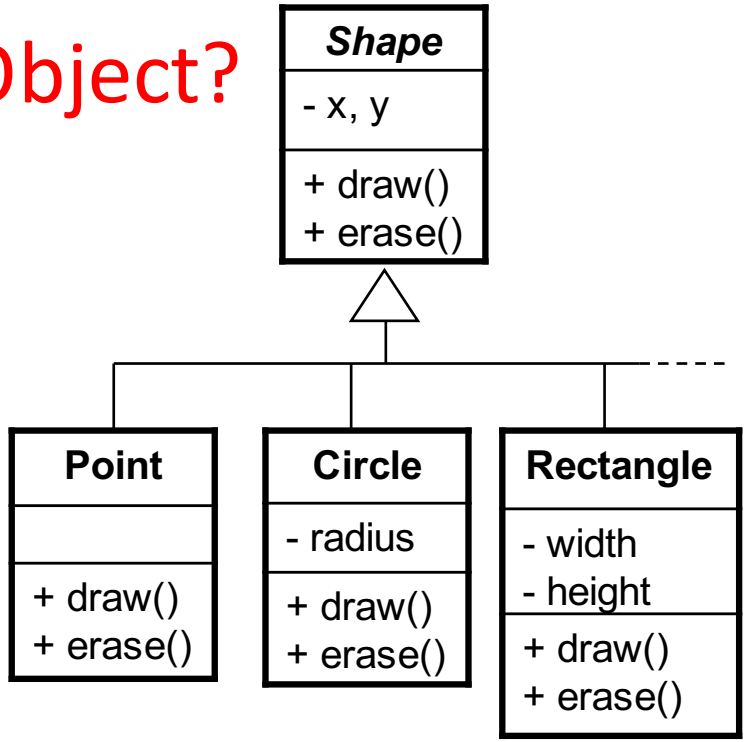
→ But, what does a generic Animal Object look like?

→ Do we ever need an Animal Object?



Abstraction vs. Inheritance Design

- What does a generic Shape Object look like?
- How to *draw()* it?
- Do we ever need a Shape Object?



Abstract Classes

- Abstract classes present generic classes. Abstract classes are **not** instantiated
- Why care about abstract classes?
  - We want Circle and Triangle objects, but no Shape objects
  - We want Dogs and Cats, but no Animal objects
- Declare abstract classes with the keyword "abstract"

```
public abstract class Animal {
  public void eat() {}
  ...
}
```

Abstract Classes

- In an abstract class:
  - The compiler will guarantee that no instances are created
  - But object references of abstract class types are allowed

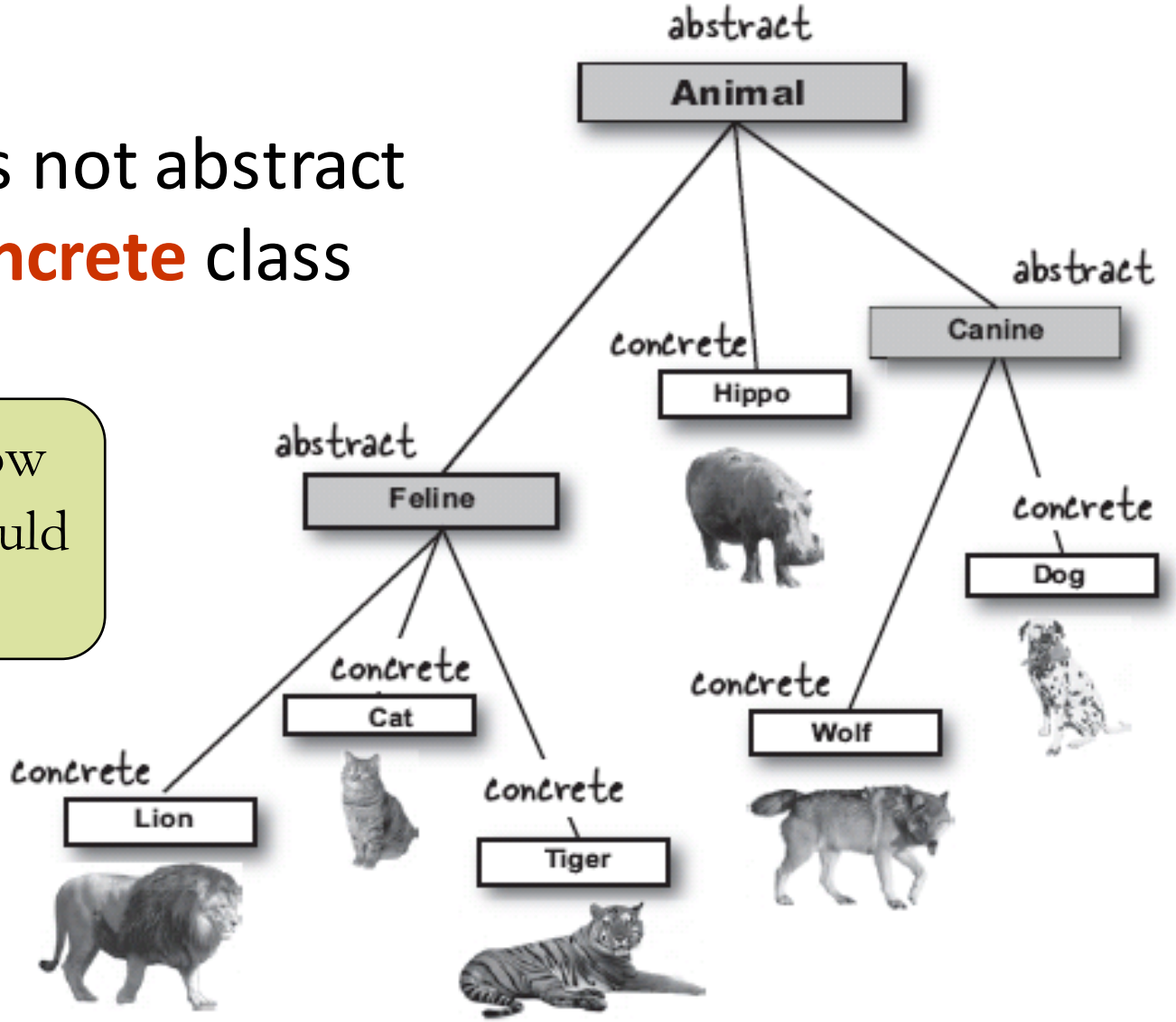
```
public abstract class Animal {
}
public class Dog extends Animal {
}
```

```
Animal a = new Animal(); // Error!!!
Animal d = new Dog(); // no error.
```

Abstract vs. Concrete

- A class that is not abstract is called a **concrete** class

How do we know when a class should be abstract?



Abstract vs. Concrete

- mobile phone
- smart phone
- iPhone
- iPhone 4

Abstract Methods

- If Animal is an abstract class, how do we implement?
  - Animal.makeNoise() or Animal.eat()?

```
public void makeNoise() {
  System.out.print("Hmm");
}
```

  - Is there any generic implementation that is *useful?*
- For this, we mark those generic methods as "**abstract methods**" with no body

```
public abstract class Animal {
  public abstract void makeNoise();
  ...
```

No method body!
End it with a semicolon.

Abstract Methods

Abstraction rules:

- An abstract method must belong to an abstract class. A concrete class cannot contain an abstract method
- An abstract class means that it must be extended
- An abstract method means that it must be overridden
- A concrete subclass must have all the inherited abstract methods implemented

```
public abstract class Shape {
    protected int x, y;
    Shape(int _x, int _y) {
        x = _x;
        y = _y;
    }
    public abstract void draw();
    public abstract void erase();
    public void moveTo(int _x, int _y) {
        erase();
        x = _x;
        y = _y;
        draw();
    }
}
```

```
public class Circle extends Shape {
    private int radius;
    public Circle(int _x, int _y, int _r) {
        super(_x, _y);
        radius = _r;
    }
    public void draw() {
        System.out.println("Draw circle at "+x+","+y);
    }
    public void erase() {
        System.out.println("Erase circle at "+x+","+y);
    }
}
```