继承与多态

Java Reference

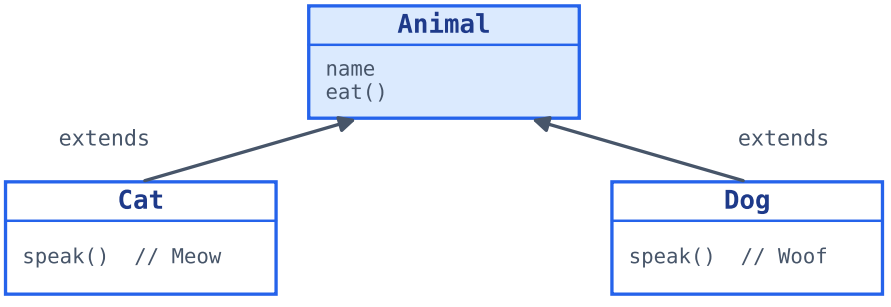
Inheritance

继承继承允许子类子类复用父类父类。编写 `class Cat extends Animal` 和 `Cat` 即可从 `Animal` 继承符合条件的字段和方法。私有成员不会被继承；构造函数也不会被继承。使用 `super(...)` 调用父类构造函数。

```
public class Main {
    public static void main(String[] args) {
        Cat c = new Cat("Milo");
        c.eat();      // Milo is eating  (inherited from Animal)
        c.speak();    // Meow            (Cat's own method)
    }
}

class Animal {
    protected String name;
    public Animal(String name) { this.name = name; }
    public void eat() { System.out.println(name + " is eating"); }
}

class Cat extends Animal {
    public Cat(String name) { super(name); }     // call Animal's
↪   constructor
    public void speak() { System.out.println("Meow"); }
}
```



Cat and Dog inherit name and eat(), and add their own speak()

09-inheritance

Cat 和 *Dog* 都 *extends Animal:* 继承它的成员，再加上自己的方法

Polymorphism & toString

子类可以重写 (override) 一个方法, 以替换父类的版本。多态 (polymorphism) 是指一个 Shape 变量可以持有任何子类型,Java 会在运行时选择正确的 toString。System.out.println(obj) 会自动调用 obj.toString()。

```
public class Main {
    public static void main(String[] args) {
        Shape[] shapes = { new Circle(2), new Square(3) };
        for (Shape s : shapes) {
            System.out.println(s);          // each calls its own
↪   toString
        }
    }
}

class Shape {
    public String toString() { return "a shape"; }
}

class Circle extends Shape {
    private int r;
    public Circle(int r) { this.r = r; }
    public String toString() { return "Circle r=" + r; }     // override
}

class Square extends Shape {
    private int side;
    public Square(int side) { this.side = side; }
    public String toString() { return "Square side=" + side; }   //
↪   override
}
```

常见错误

- 重写方法的签名必须完全一致; 加上 @Override, 让编译器帮你抓错。
- super(...) 必须是子类构造方法的第一行。
- 子类对象**是**一个父类对象, 但反过来不成立。