Inheritance & polymorphism

Java Reference

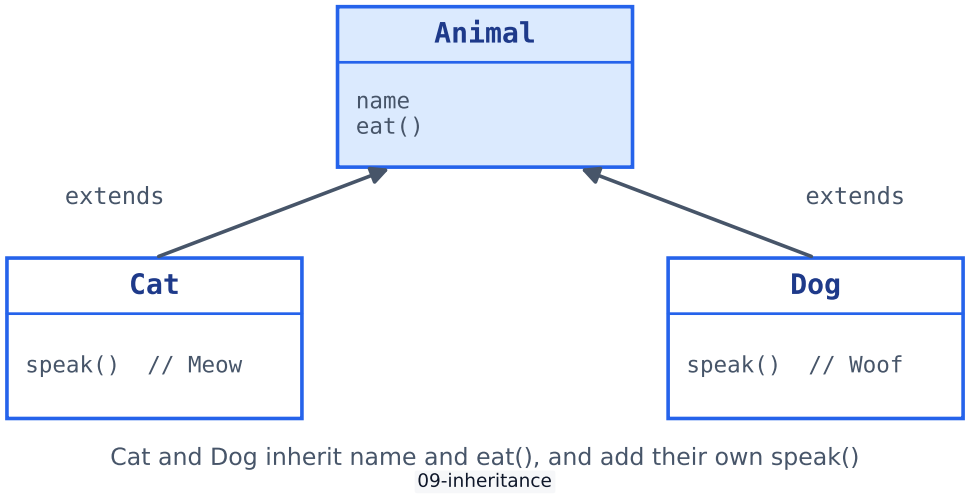
Inheritance

Inheritance 继承 lets a subclass 子类 reuse a superclass 父类. Write `class Cat extends Animal` and `Cat` gets `Animal`'s fields and methods for free. Call the parent constructor with `super(...)`.

```
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 extend Animal: they inherit its members and add their own

Polymorphism & toString

A subclass can override 重写 a method to replace the parent's version. Polymorphism 多态 means a `Shape` variable can hold any subtype, and Java picks the right `toString` at run time. `System.out.println(obj)` automatically calls `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
}
```

Common mistakes

- An overriding method must match the signature exactly; add `@Override` so the compiler catches slips.
- `super(...)` must be the first line of a subclass constructor.
- A subclass object IS-A superclass object, but not the other way round.