

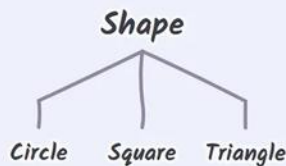
Inheritance, Interfaces and Polymorphism

reusing a class without copying it, and letting one call mean different things



Inheritance is an is-a relationship

- a subclass extends a superclass, taking its fields and methods without copying any code, and adding or changing what it needs
- the test is the sentence: a Circle is a Shape, so Circle extends Shape; a Car has an Engine, which is composition and not inheritance
- super reaches the parent: super() calls the parent constructor and must be the first statement, and super.method() calls the parent's version
- a subclass constructor always runs its parent's constructor first, whether or not super() is written, so the chain builds from the top down



hierarchical inheritance: one superclass, several subclasses, each an is-a of the parent

The forms Java allows, and the one it does not *

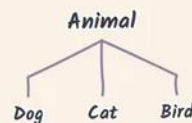
Single

one subclass extends one superclass, the ordinary case

Multilevel

a subclass is itself extended, so the chain is three deep or more

Hierarchical



several subclasses extend the same superclass

Multiple, NOT allowed

a Java class cannot extend two classes, because a clash between two inherited versions would have no resolution; interfaces are the way round it

Multilevel inheritance, and how far a member travels

