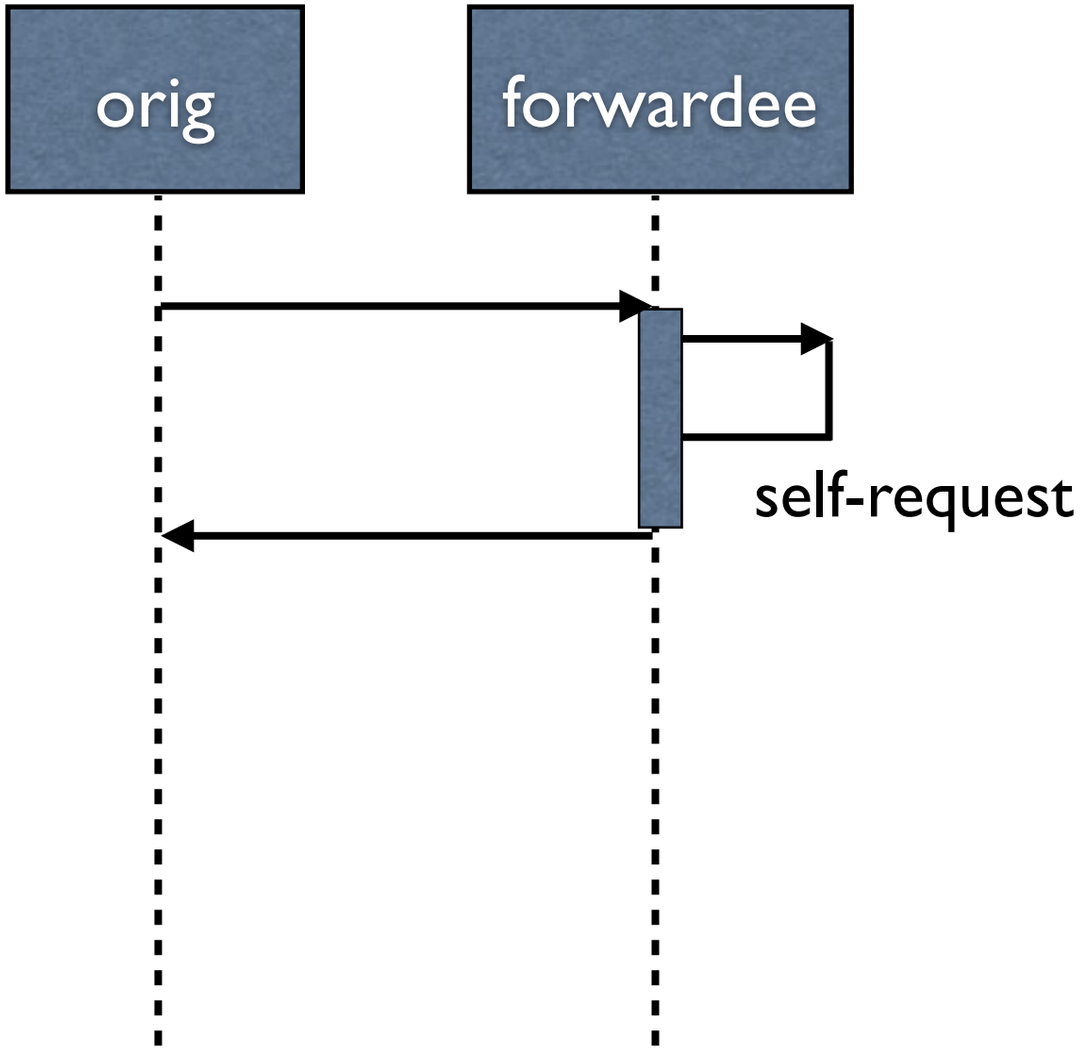
Acquiring behaviour through Inheritance

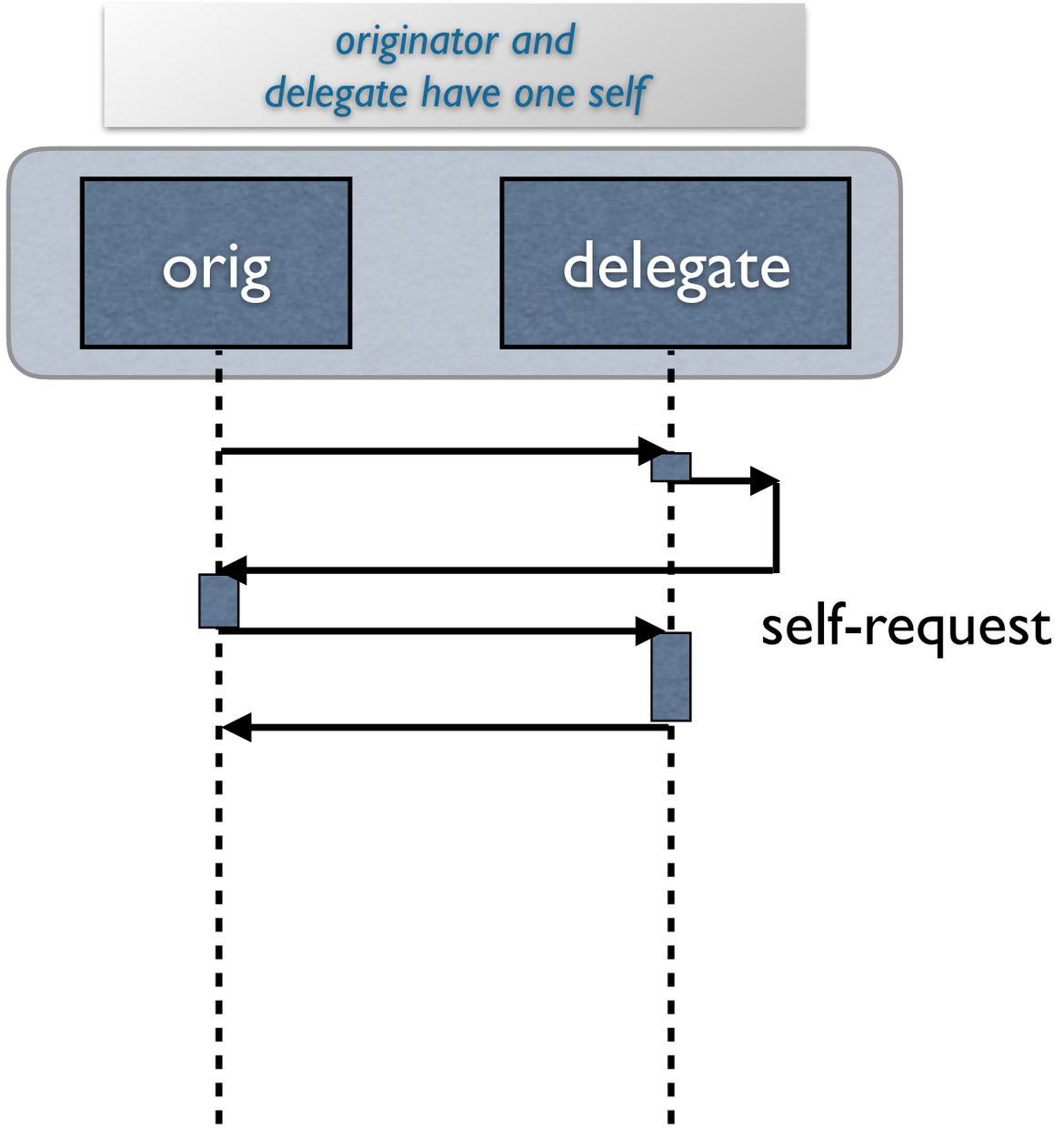
Andrew P. Black

What is Inheritance?

- Objects respond to requests
- How?
  - they have their own methods
  - they “pass the buck” to another object: forwarding
  - they acquire behavior from another object: delegation

Forwarding vs Delegation





The right class structure

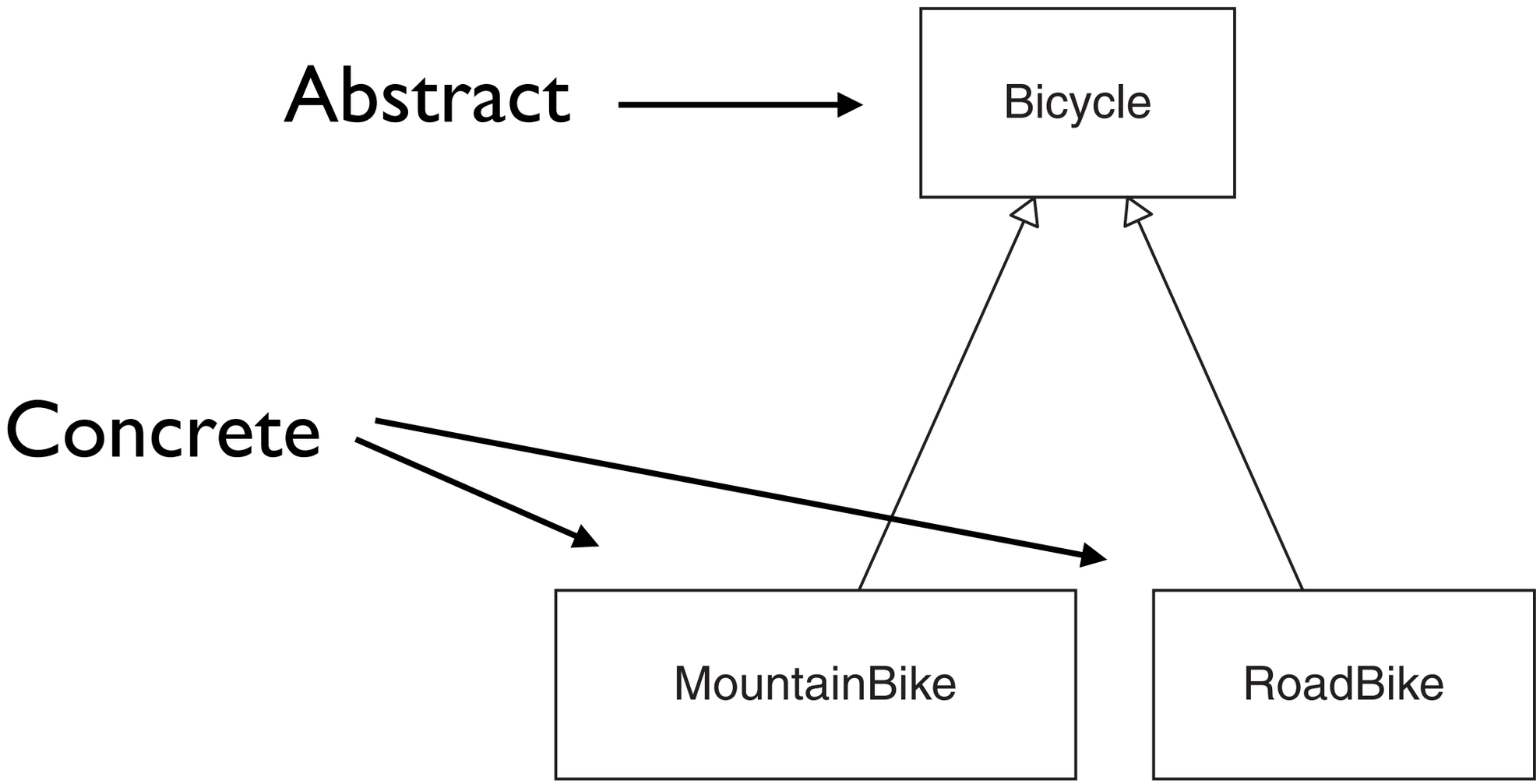


Figure 6.6 *Bicycle* as the superclass of *MountainBike* and *RoadBike*.

Abstract & Concrete

- Abstract Classes **exist** to be subclassed. This is their sole purpose.
  - it rarely makes sense to have an abstract superclass with only one subclass.
  - don't try to anticipate the need to subclass
- Put off creating a superclass until you need *three* subclasses?

Learn to spot missing classes

- Where have you seen an object with a field that encodes a *style*, *category*, *rôle* or *kind* ?
  - bicycle.style → **road** or **mountain**
  - twoThreeTreeNode → **twoNode** or **threeNode**
  - dancingBox → **leader** or **follower**
- If the kind or role *changes*, consider a state or strategy component

How to add inheritance

- Add an empty abstract superclass
- It is better to promote code **up** to a superclass than to push it down to a subclass.
  - *When* you make a mistake, you will get a *noSuchMethod* error, rather than the wrong behaviour.

Template method

- requiring a sub-object to send a *super* message creates another opportunity for error
- Metz suggests sending a template message in the super-object
  - sub-object overrides the template method, but never requests a method of the super-object
- What's the snag?

- Implement all your template methods
  - with an empty method, or
  - with a *required*, *abstract*, or *error* method

Default values

- Where do you put default values?
  - Make each default a separate attribute

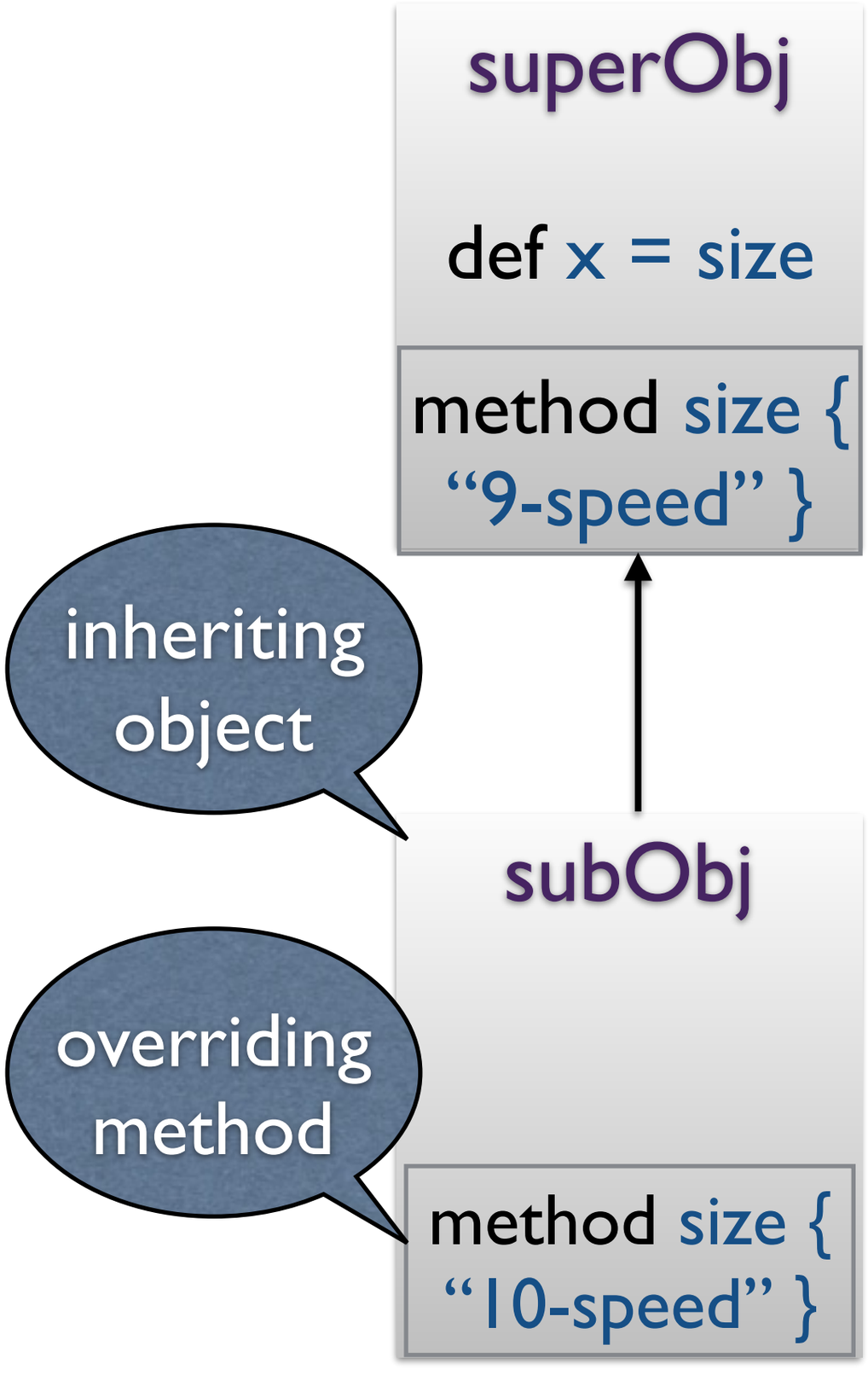
```
method defaultTireSize is confidential {
        "25 x 622"
}

def defaultChain = "11-speed"
```

- Why?

Initialization

- Inheritance makes initialization complicated
- When does initialization of variables happen?
  - How are initial values sent from the sub-object to the super-object?
  - If the initialization makes a self-request, *which version* of the requested method is executed?



- When superObj is initialized, it requests `size` on self.
- *which* `size` method is executed?
- C++ says "9-speed", Java and Grace say "10-speed"