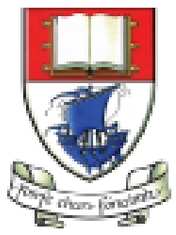
Object Oriented Concepts

An introduction to the Java Programming Language

Produced by:

Eamonn de Leastar (edeleastar@wit.ie)

Dr. Siobhan Drohan (sdrohan@wit.ie)



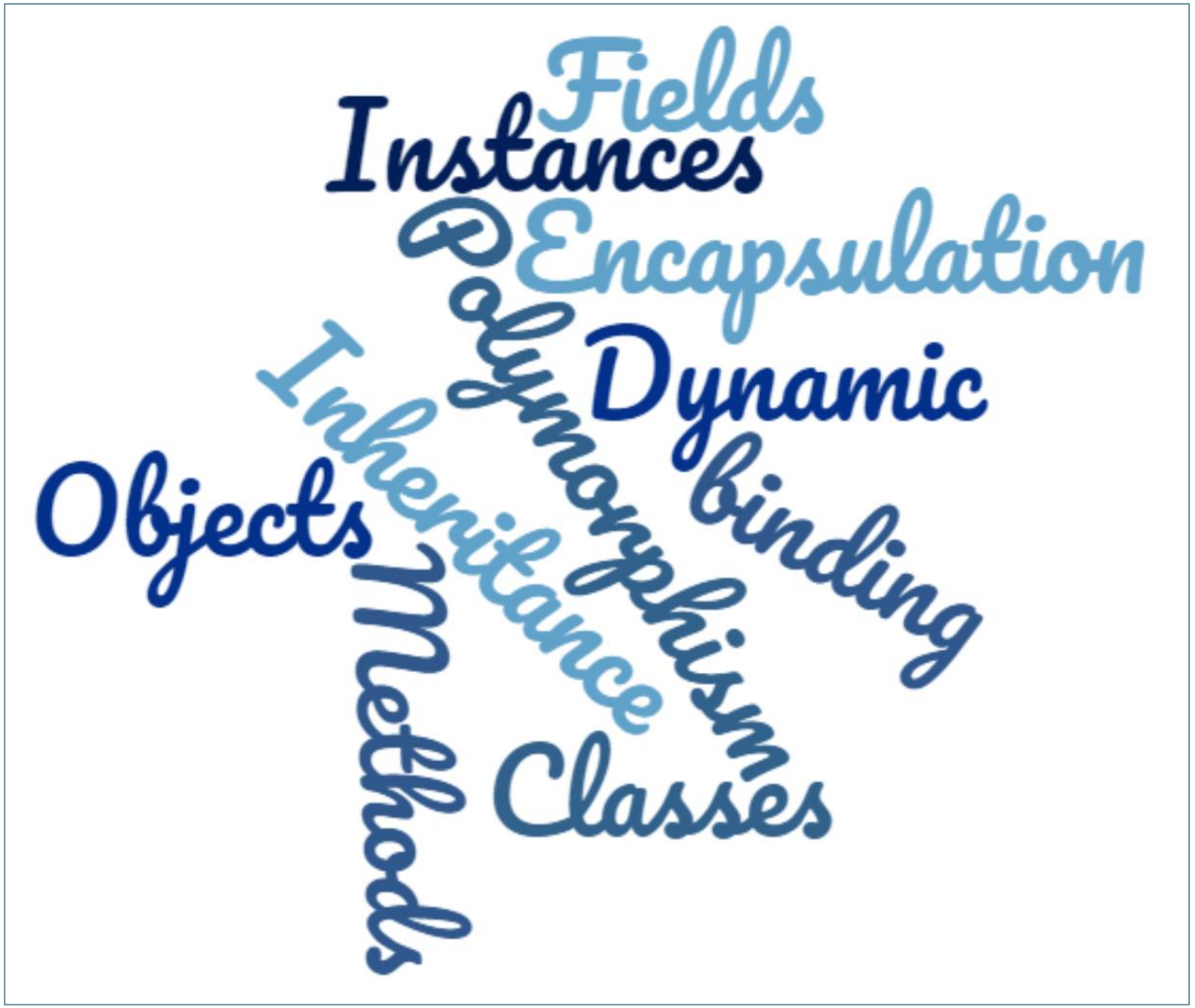
Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Centre for
Technology-Enhanced Learning

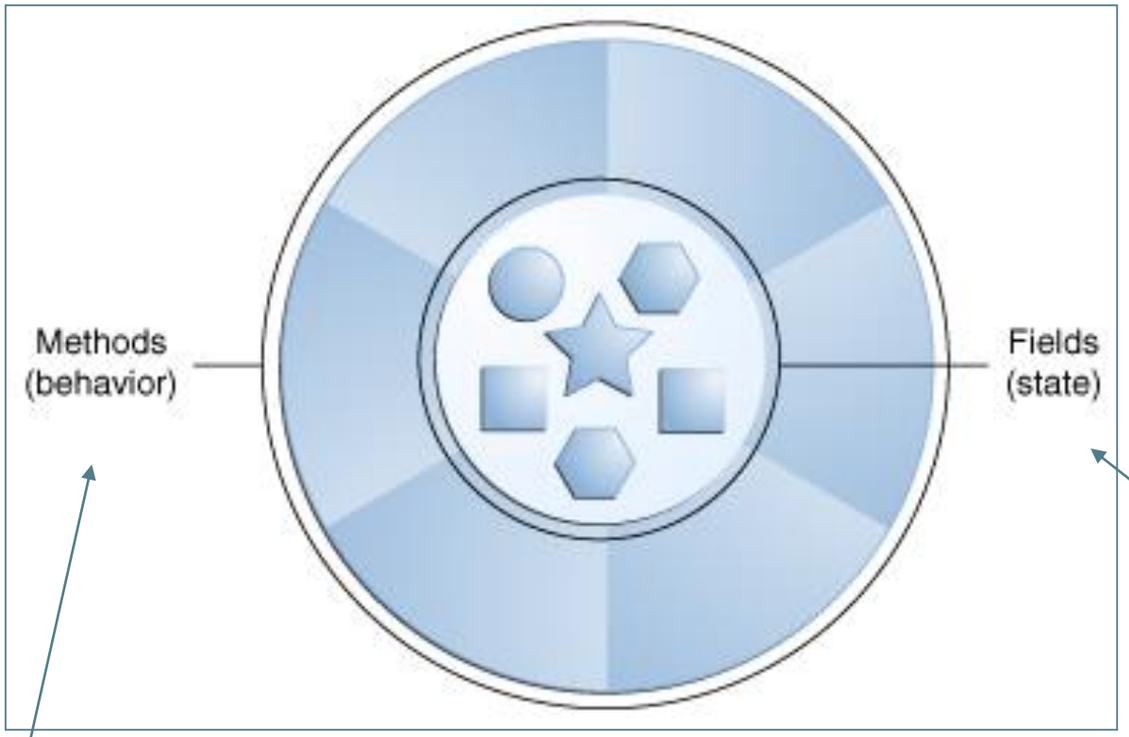
Object-Oriented Software

- Developing object-oriented software is identifying:
  - Objects
  - Characteristics of individual objects
  - Relationships between objects
- Objects interact by sending messages to each other
  - Interacting objects make an object-oriented system

Object-Oriented Terms



Objects

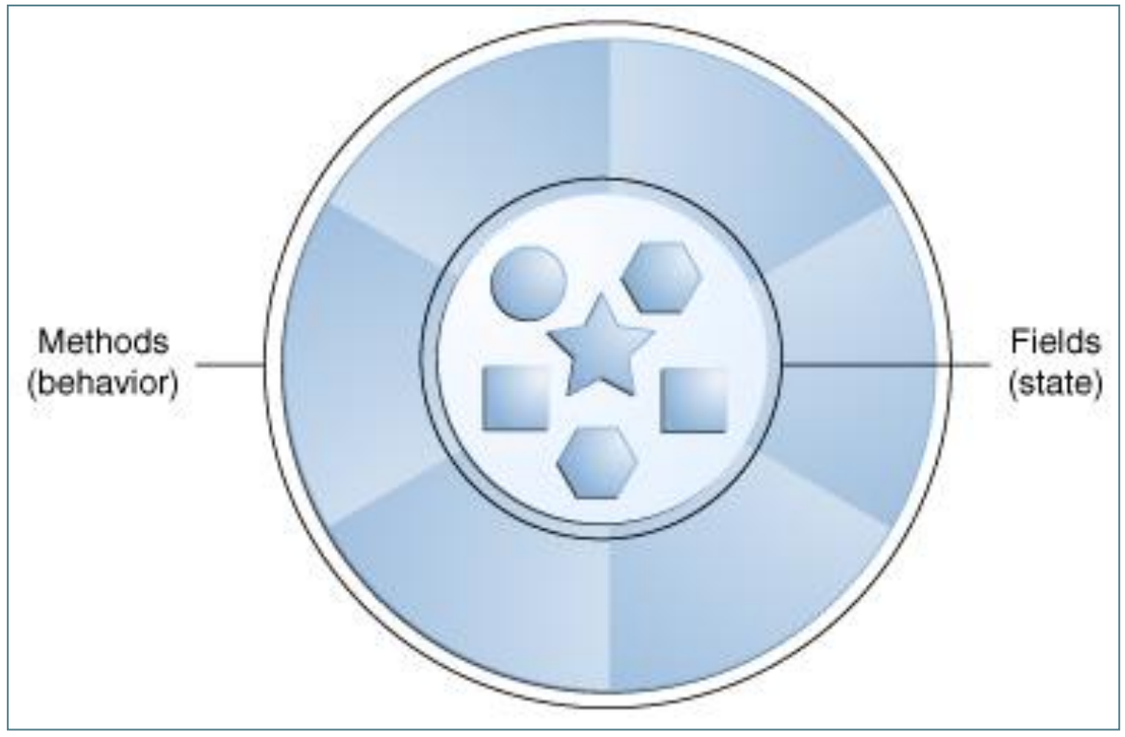


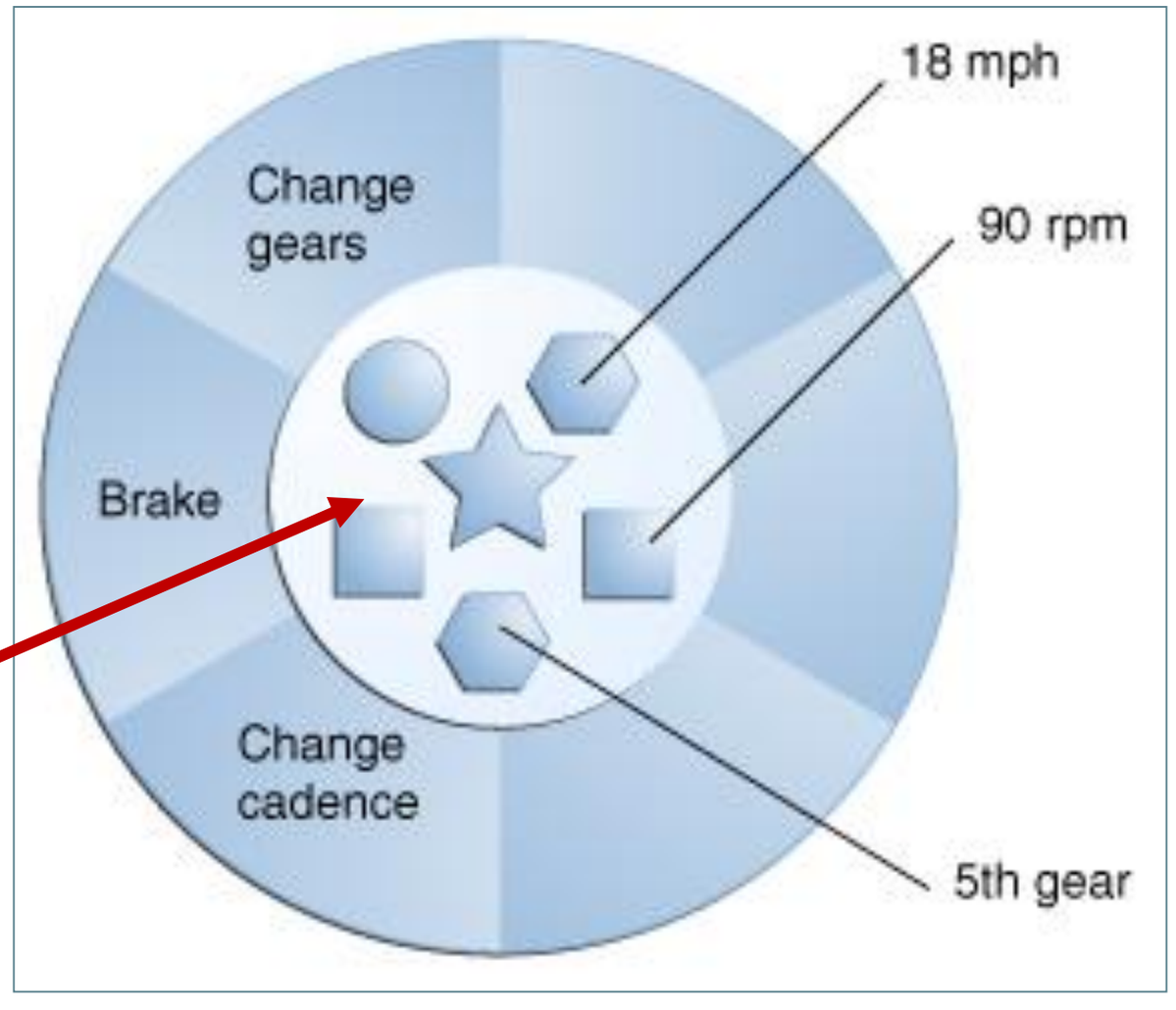
Behaviour…what an object can do.

State represents data:

What an object knows, or what an object contains.

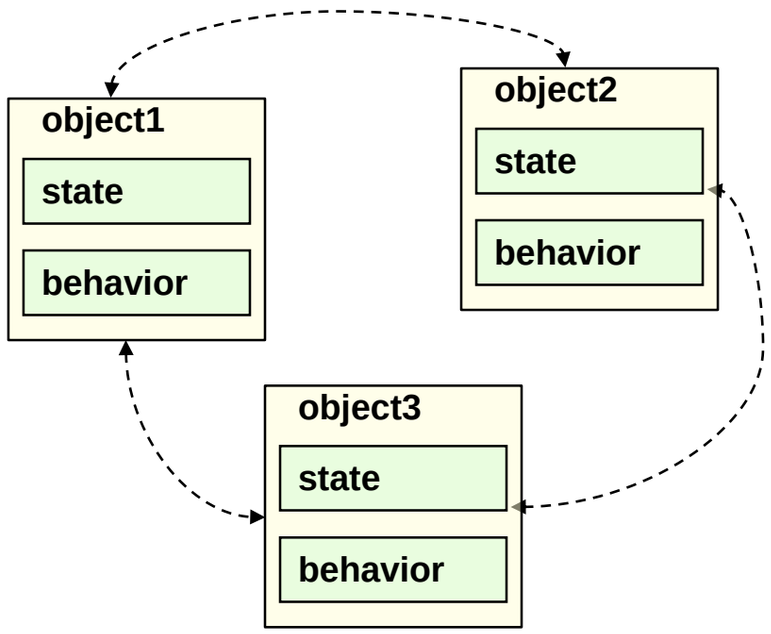
Object state





Each instance of the Bicycle class will have it's own copy of the variables.

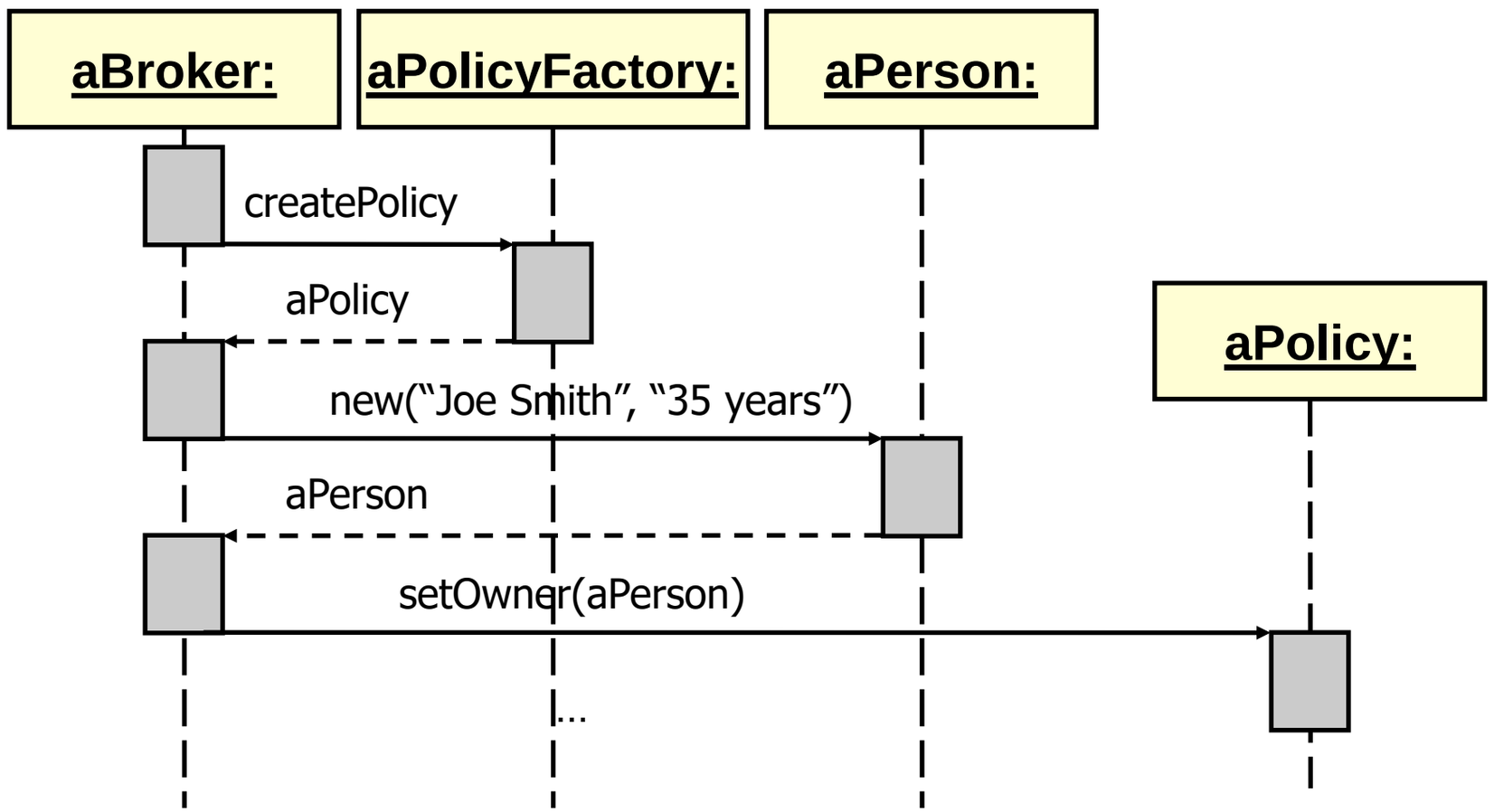
Objects and Loose Coupling



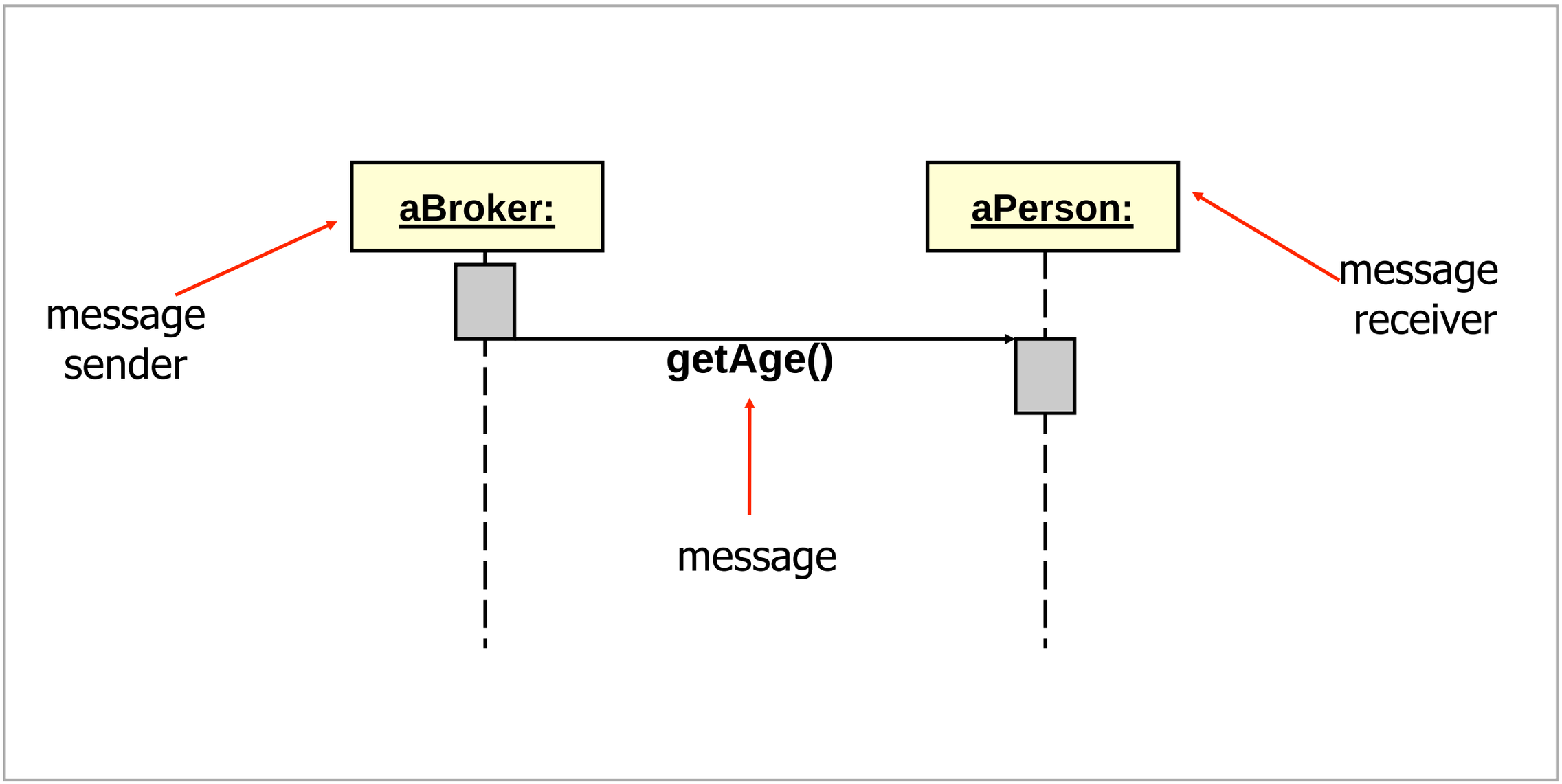
- Changing an object's data does not lead to changes in an object's external behavior
- An object's external interface stays the same
- Promotes loose coupling between objects

Interactions between Objects

- Object interact by sending messages to each other
- Objects and interactions between them make up an object-oriented system

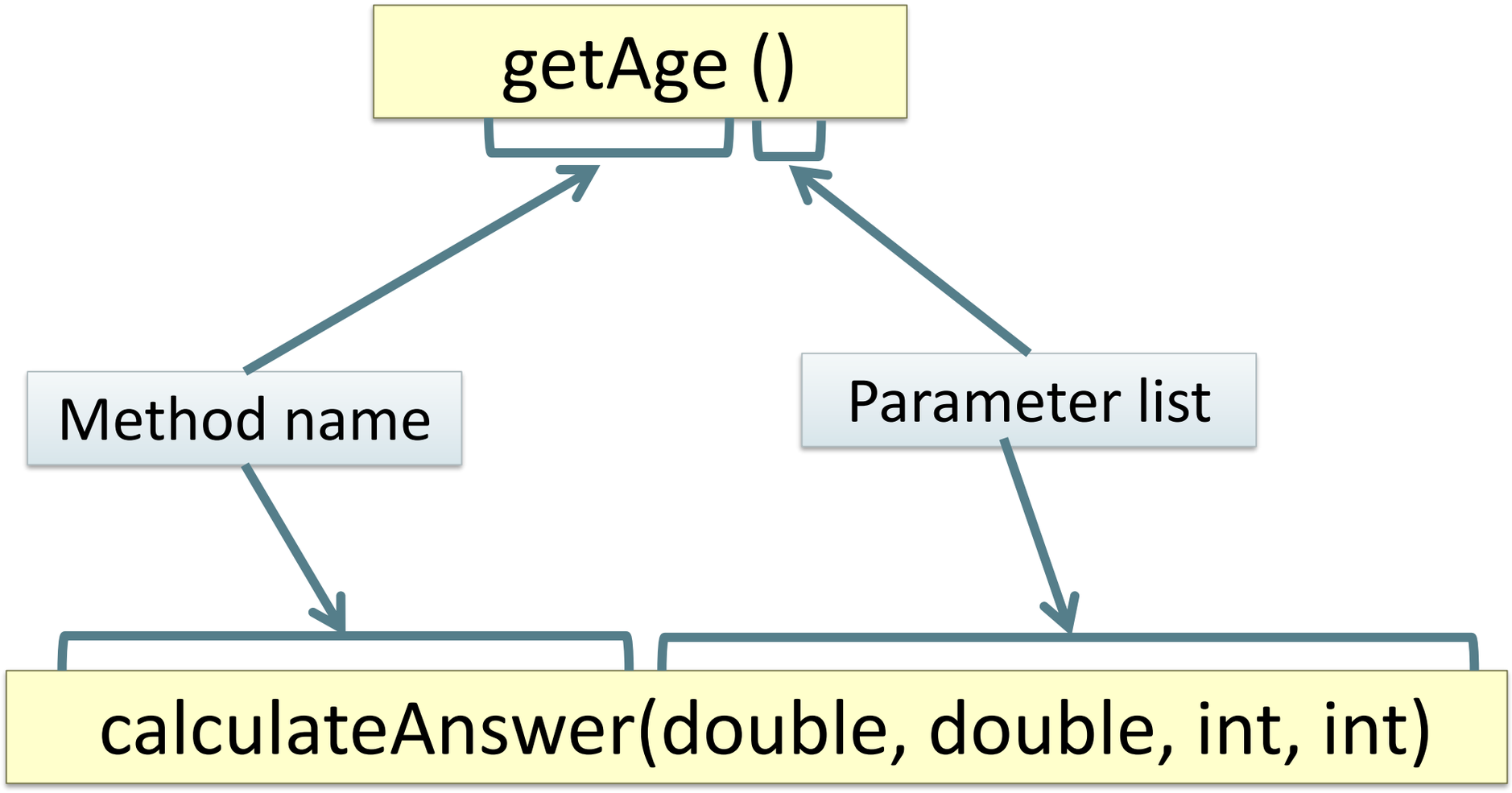


Methods → concrete implementation of a message



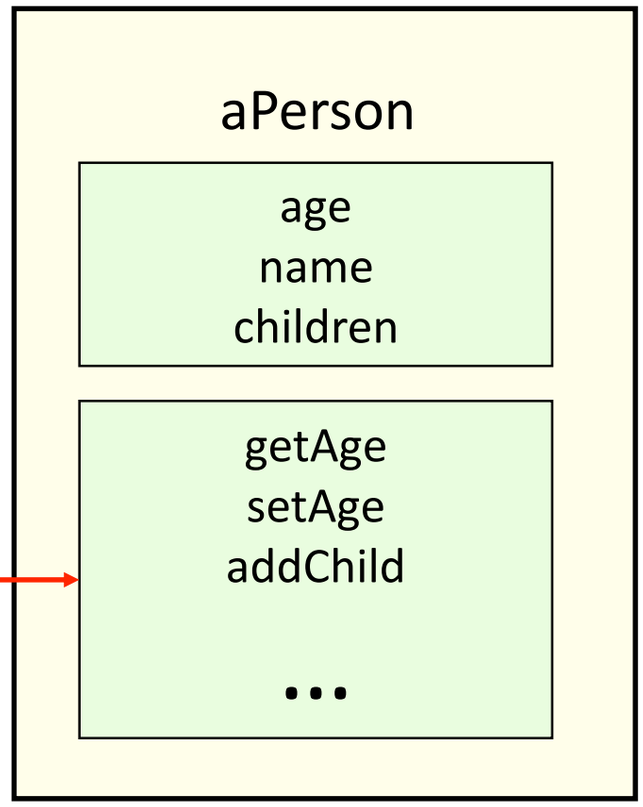
Methods may have arguments

Method Signature → Unique Identifier



Object's Public Protocol

public protocol i.e. the set of messages that can be sent to an object.

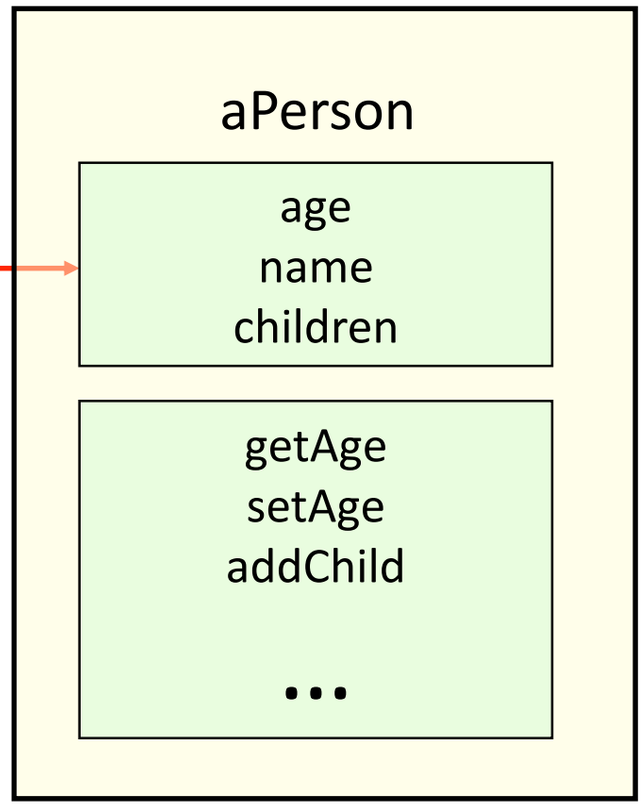


An object can also call it's own methods (e.g. private methods)

Fields

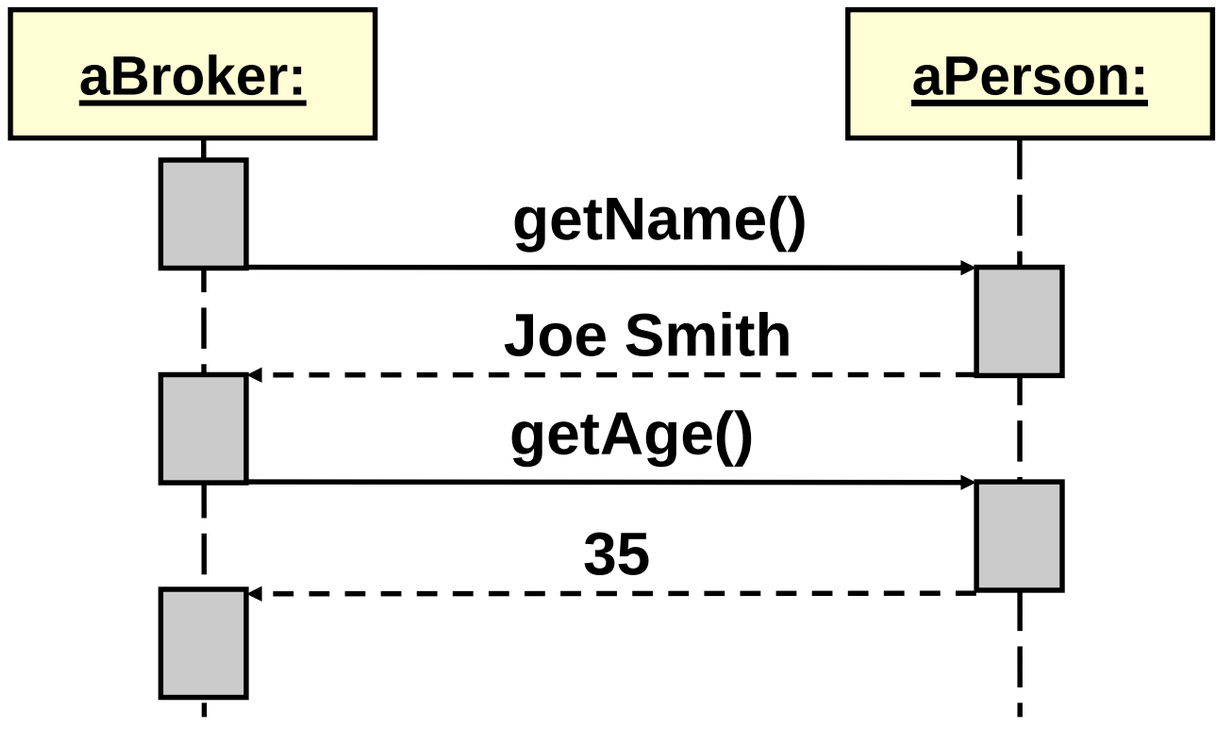
Fields are the attributes of an object.

Also known as instance variables.

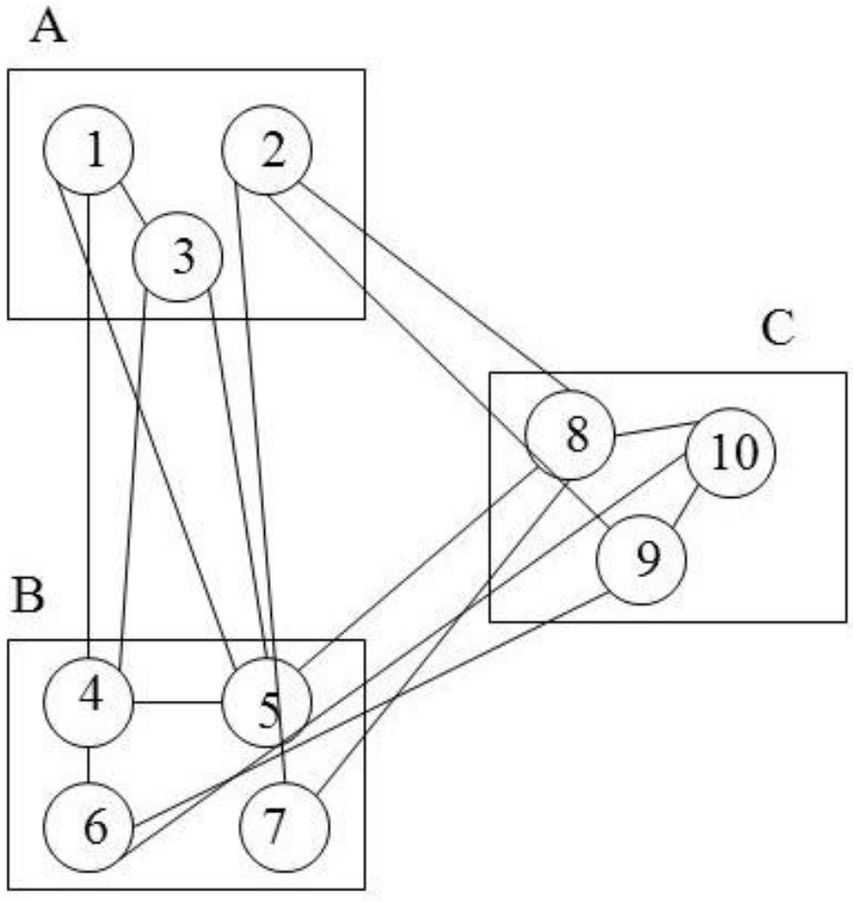


Object-Oriented Principle: Encapsulation

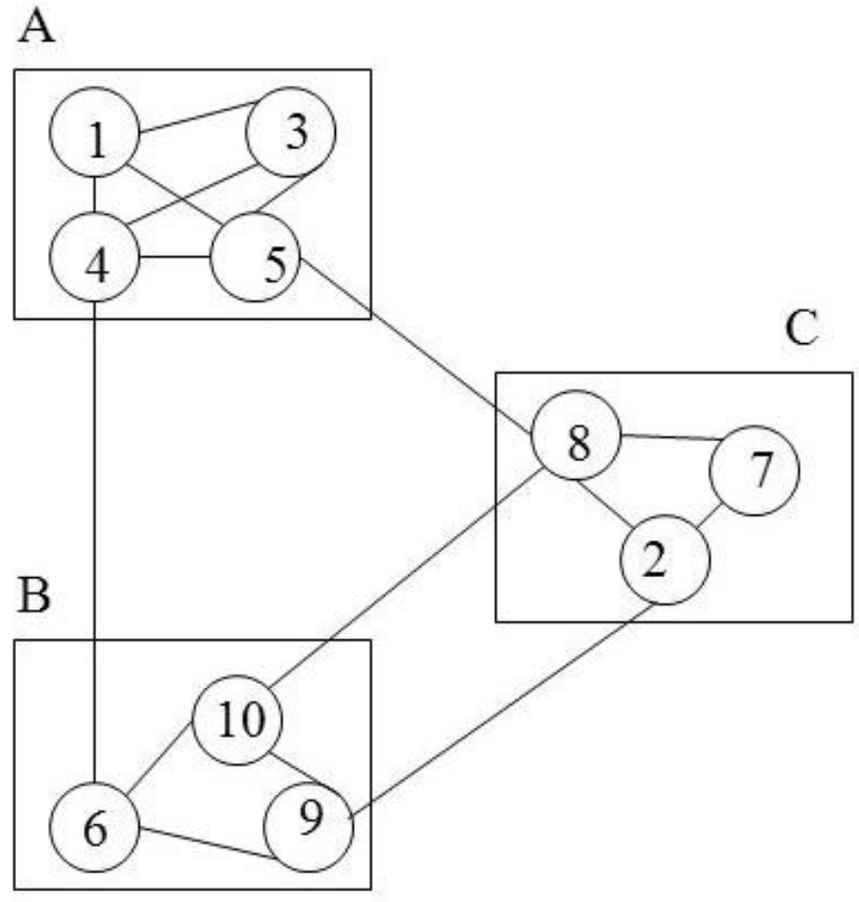
Objects hide implementation of the messages behind their public protocols (aka implementation hiding).



Modularization: Cohesion and Coupling



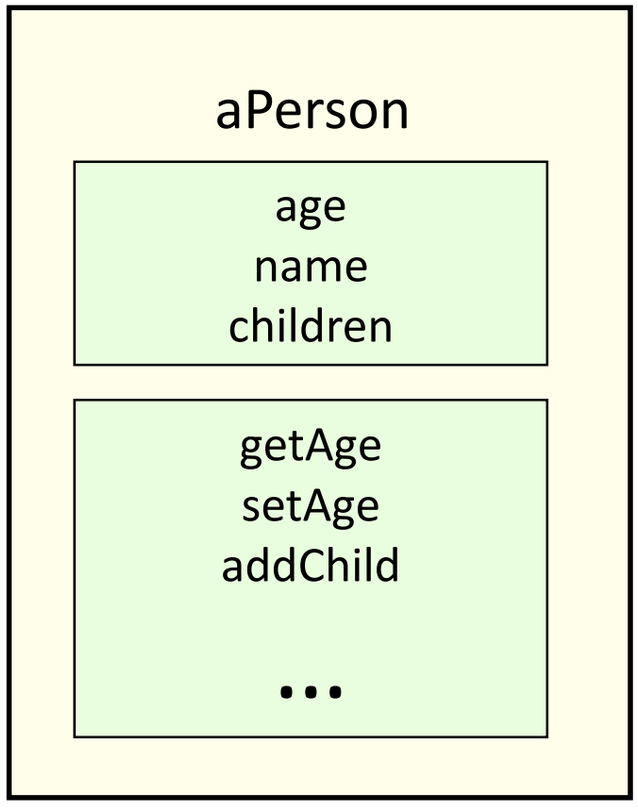
Bad modularization:
low cohesion, high coupling



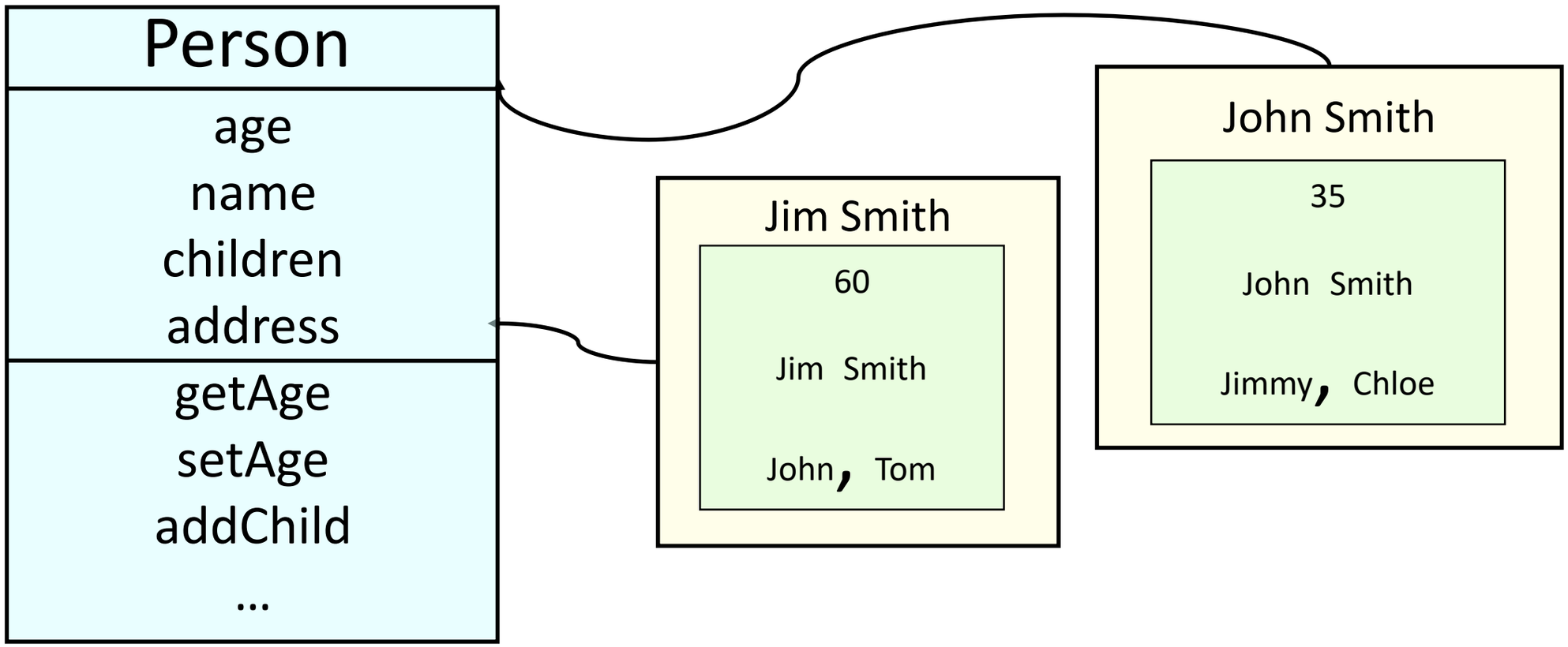
Good modularization:
high cohesion, low coupling

Classes

- Factories for creating objects
- Template for the same kind of objects that describes their state and behavior

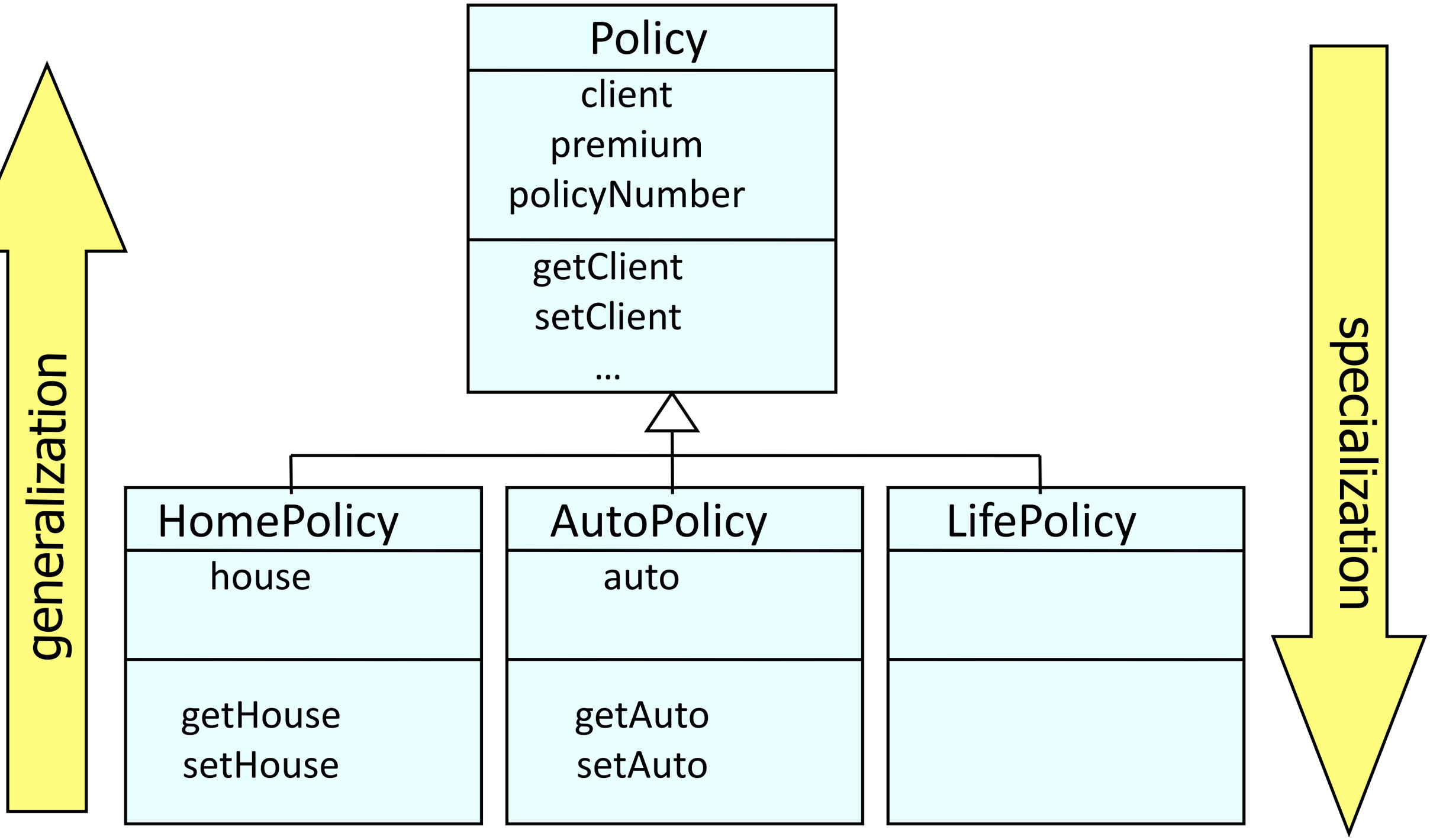


Instances → every object is an instance of some class

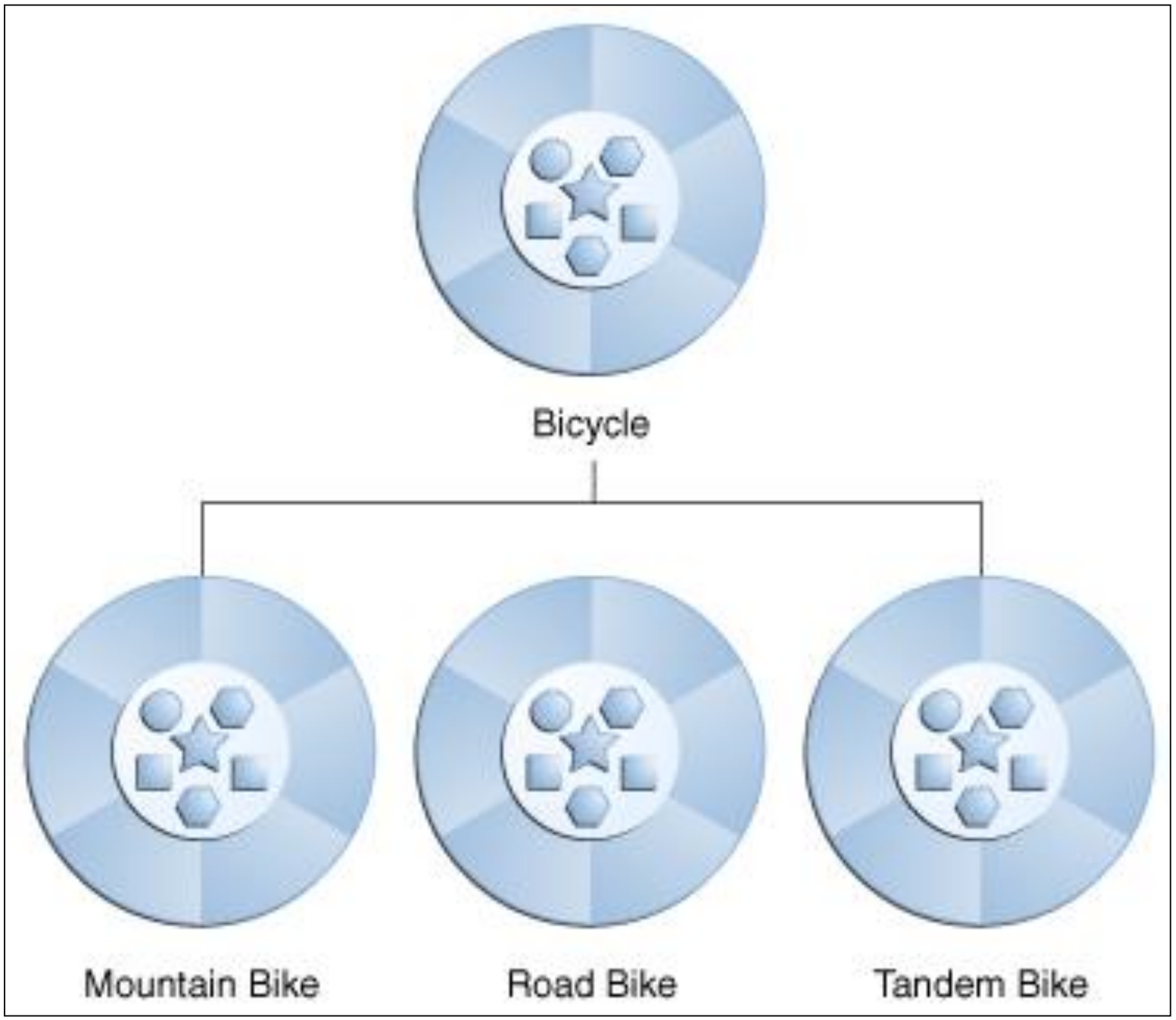


Two objects of type Person. Both have the same protocol.

Object-Oriented Principle: Inheritance



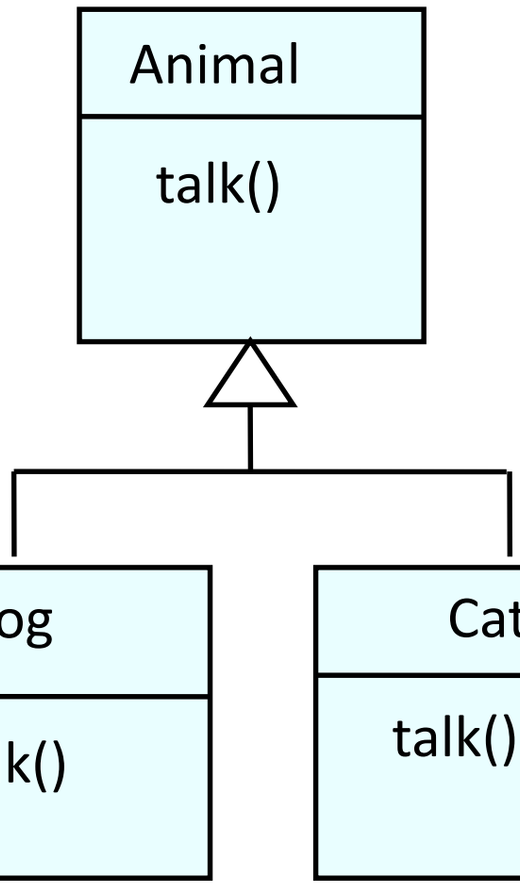
Why Inheritance?



Inheritance represents real-world modeling

Inheritance promotes reuse and extensibility

Object-Oriented Principle: Polymorphism



Polymorphism - different objects respond to the same message in different ways.

Supported by **overriding**.

```
aCat.talk();
```

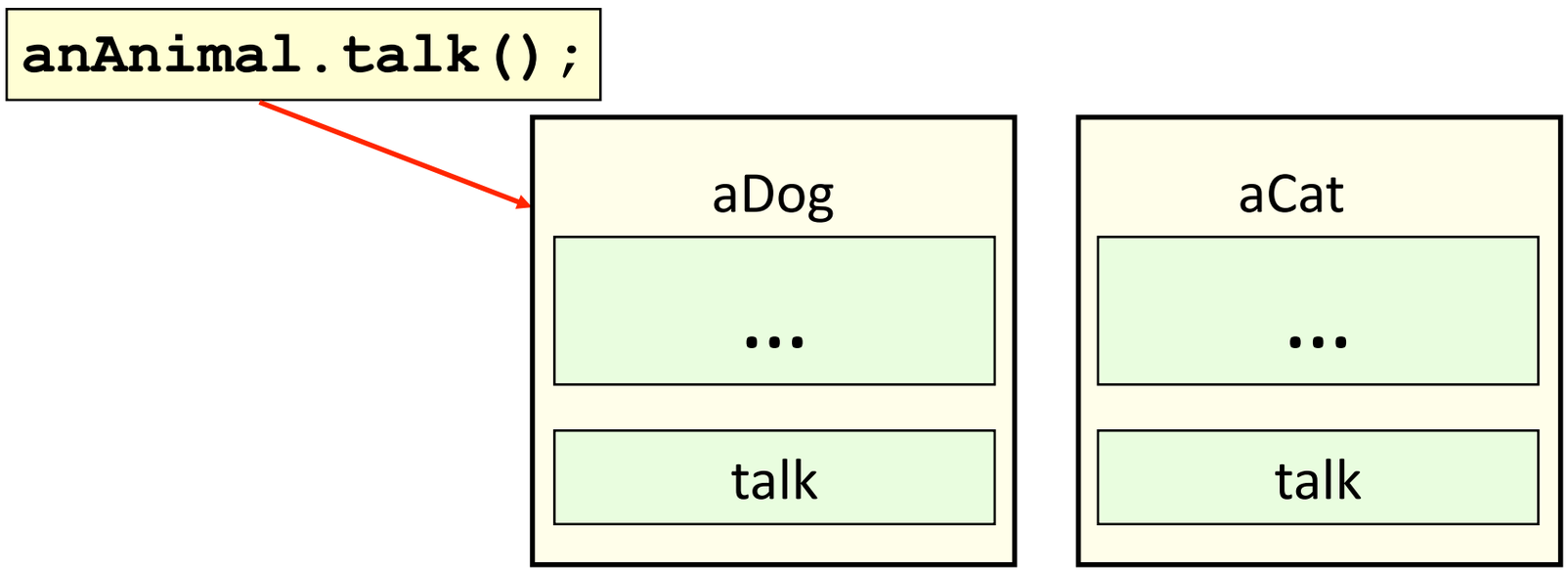
→ meows

```
aDog.talk();
```

→ barks

Overriding means that a subclass may implement the same method as the superclass, but with different code.

Dynamic Binding (runtime method binding)



It is runtime binding of a method invoked in the message to the method that implements that message.