Object Oriented Concepts

Introduction to the Java Programming Language

Produced by

Eamonn de Leastar
edeleastar@wit.ie

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

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

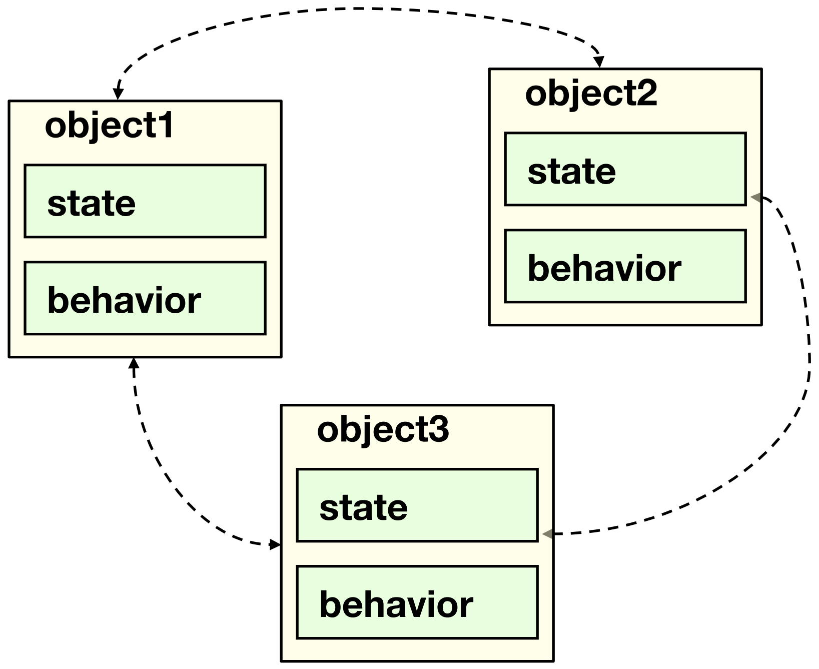
Object-Oriented Terms

- Objects
- Classes, instances, fields and methods
- Encapsulation
- Polymorphism
- Inheritance
- Dynamic binding

Objects

- Every object has:
  - State
  - Behavior
- State represents data - what an object knows, or what an object contains
- Behavior represents what an object can do

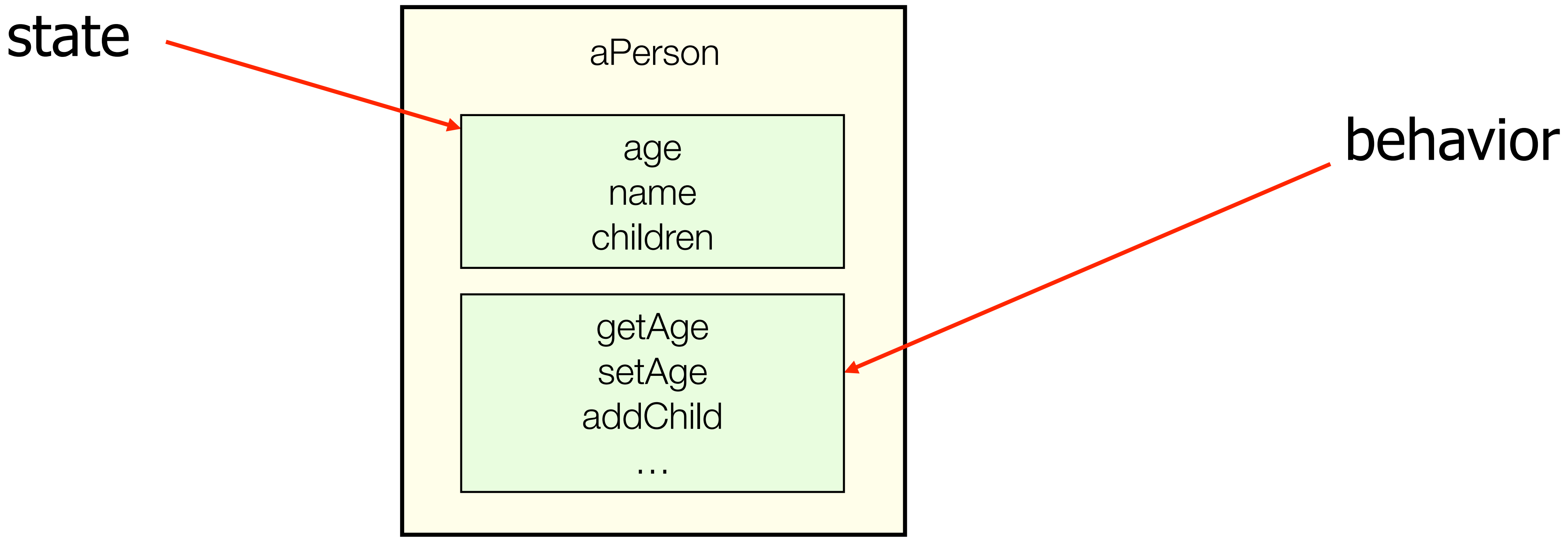
Objects and Loose Coupling



- 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

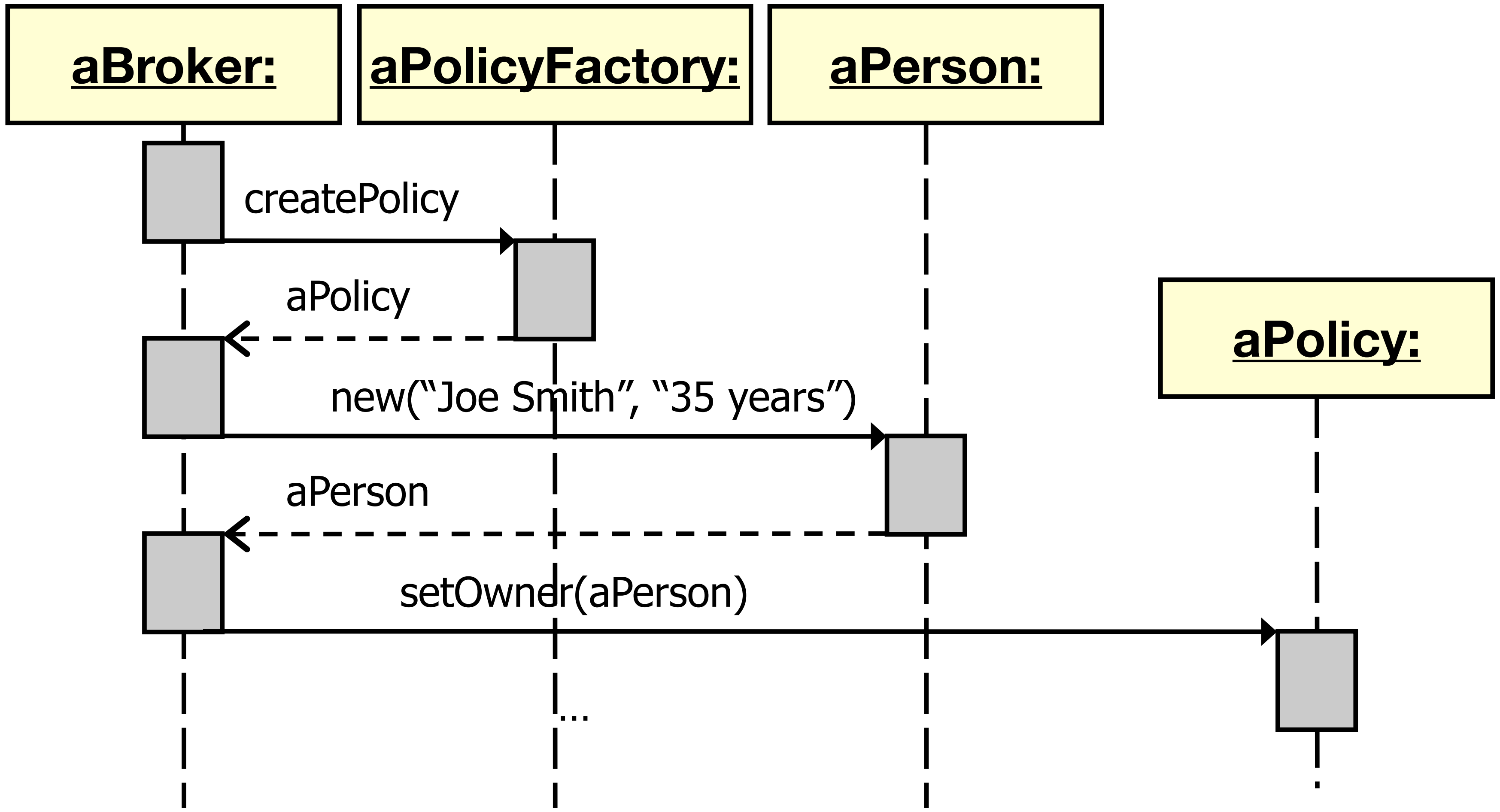
Object State and Behavior

- Person object
  - State: age, name, children
  - Behavior: addChild, getAge, setAge



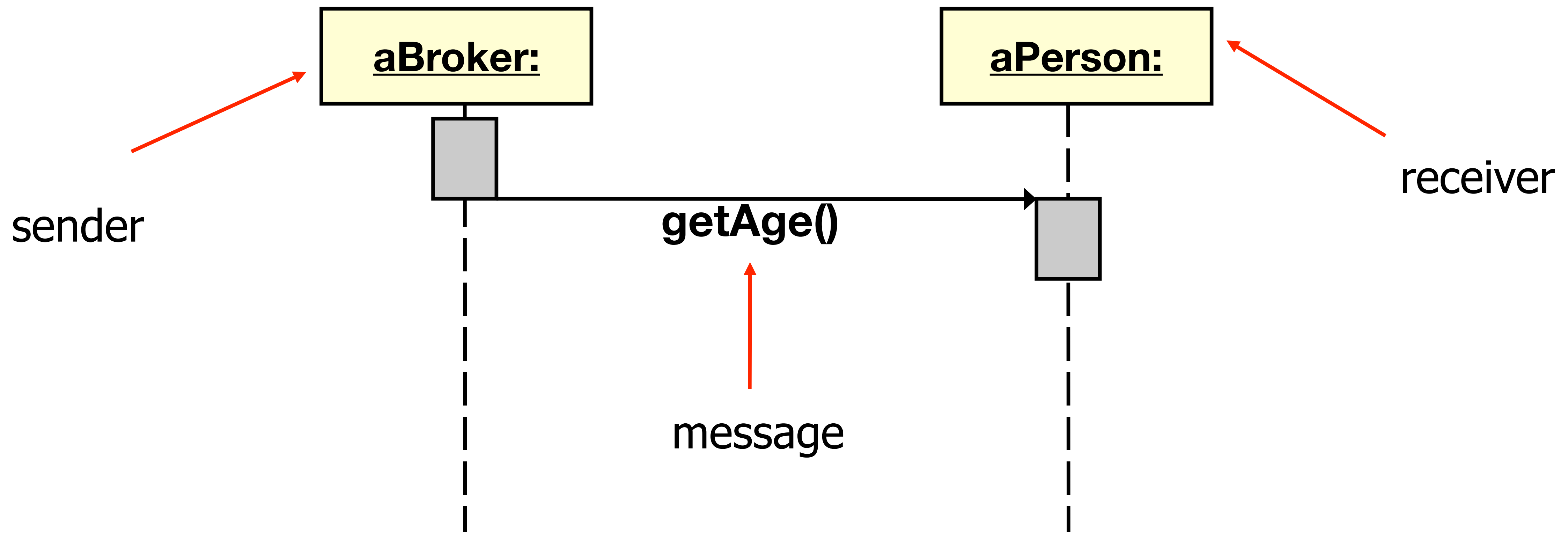
Interactions between Objects

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



Messages

- There are two major terms in messaging:
    - Message sender
    - Message receiver
- Messages may have arguments



Methods

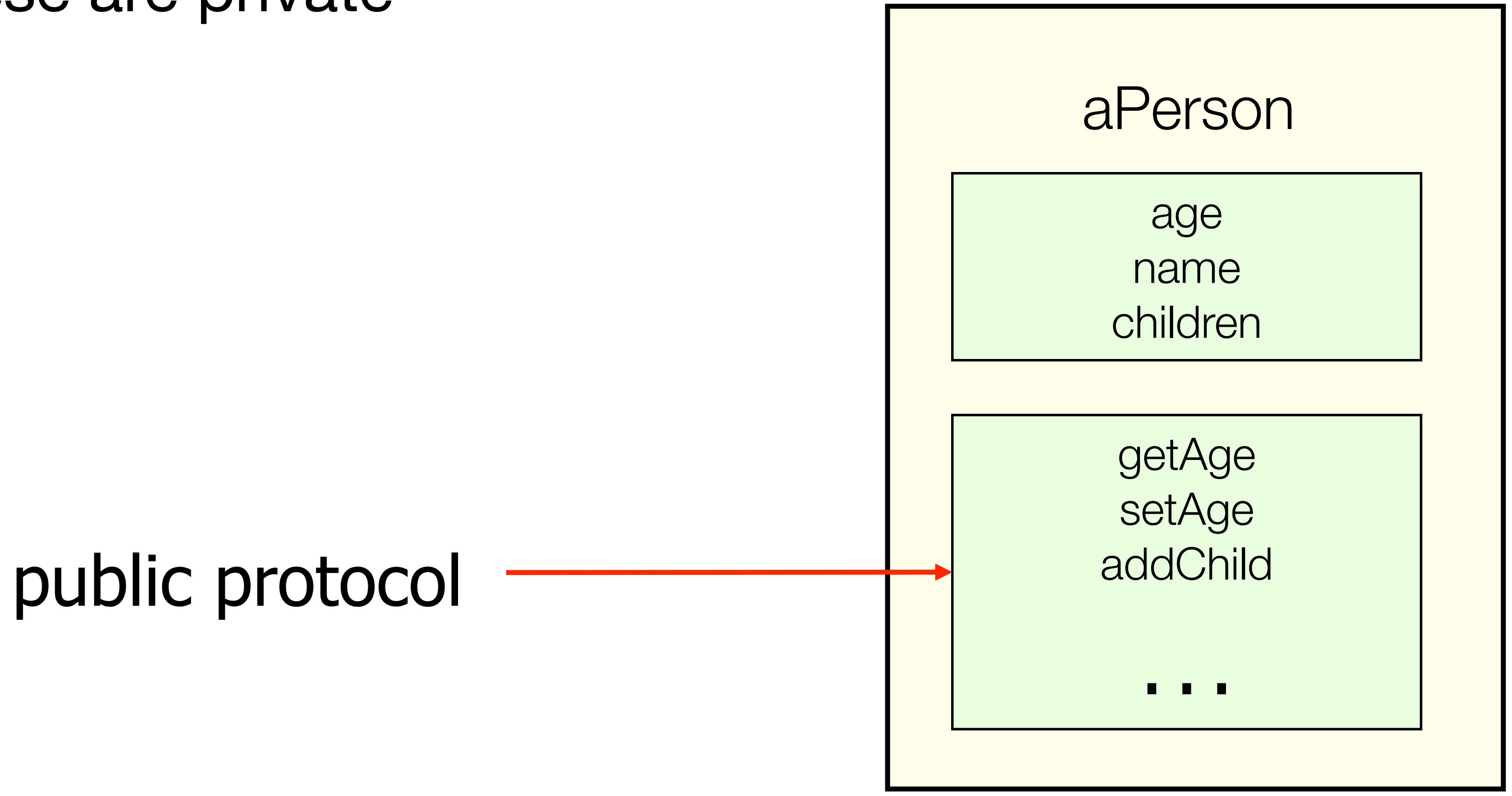
- Method is concrete implementation of a message
- When message is sent to a receiver:
  - Method is found by type of the receiver object and method signature
  - Method code is executed
- Method represents an object’s response to a message

Method Signature

- Method signature is unique identifier of the method
- It is used to distinguish methods with same name and same number of parameters
- It consists of:
    - Method name
    - Parameter name
    - Parameter type

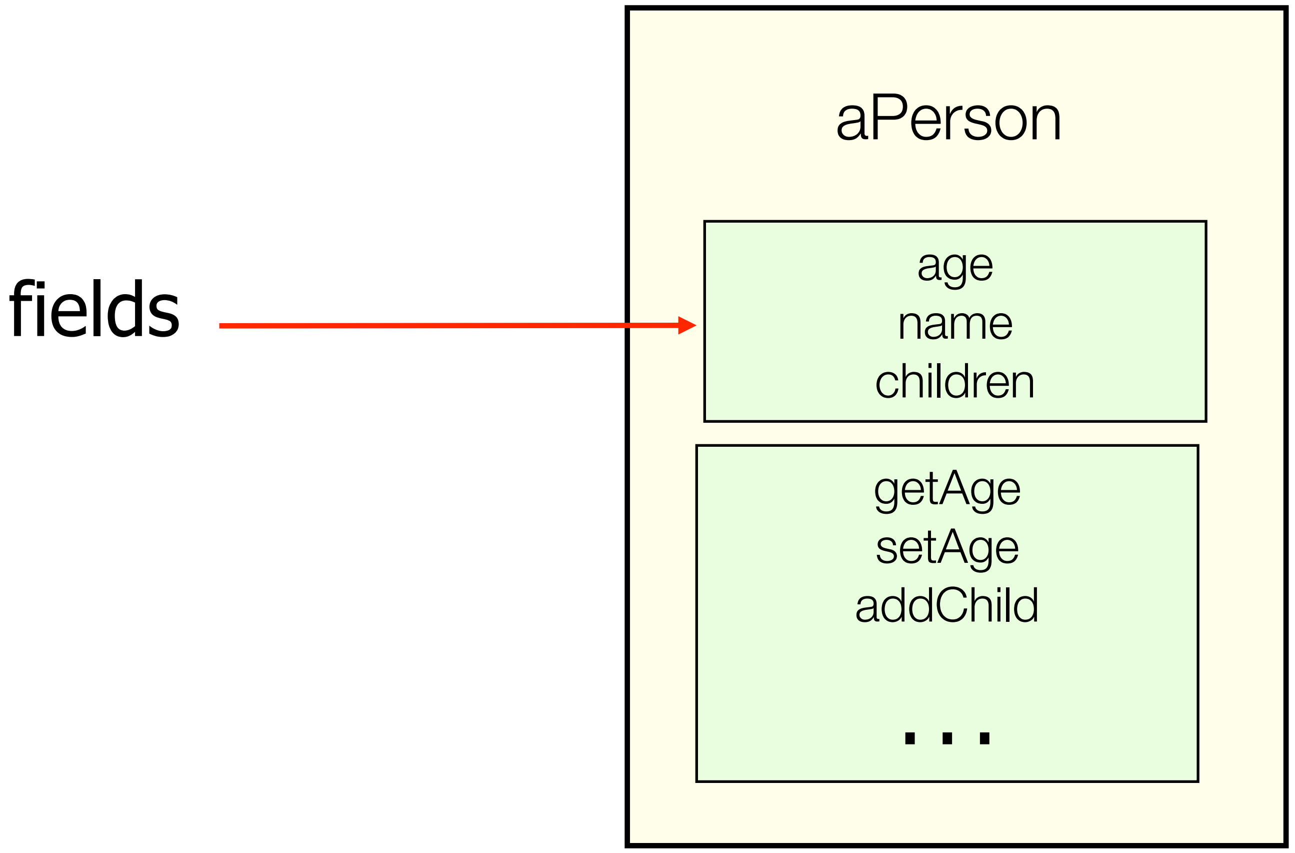
Object’s Public Protocol

- Public protocol is set of messages that can be sent to an object
- It does not include messages that an object can send to itself
  - These are private



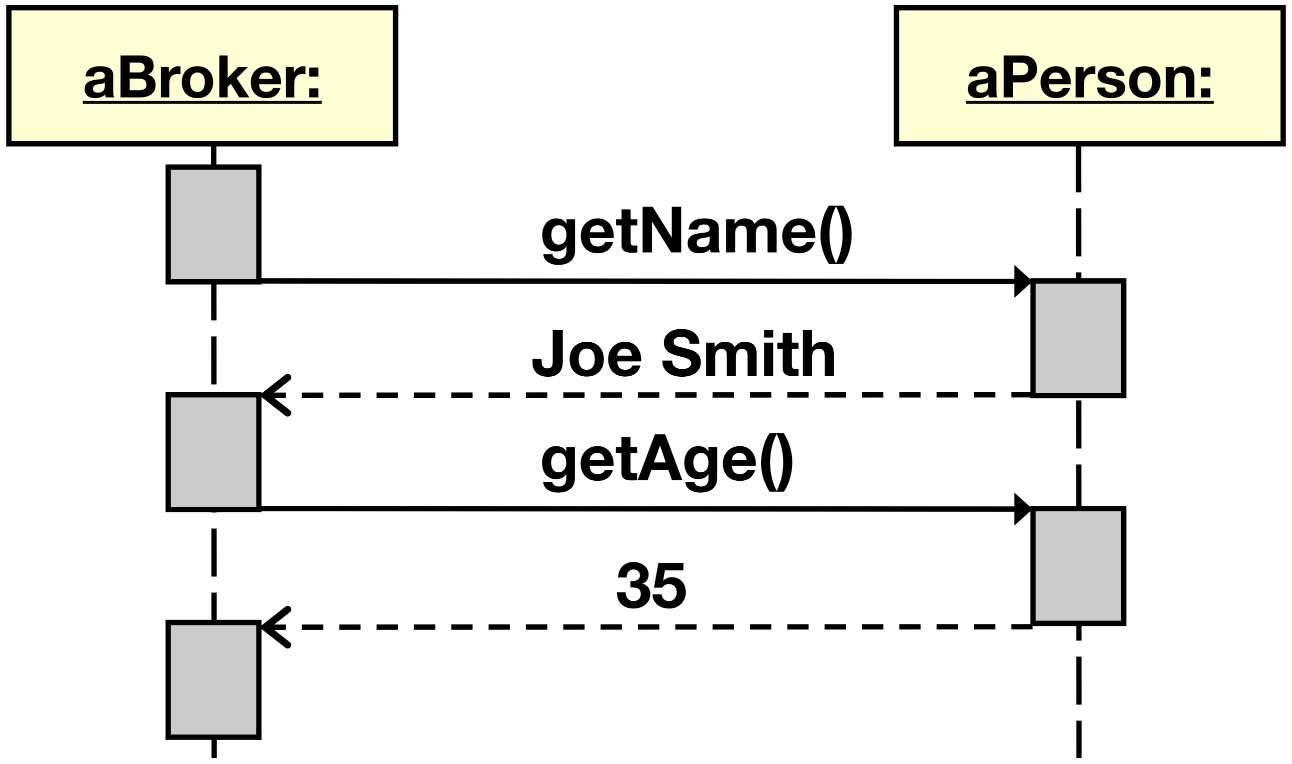
Fields

- Fields represent characteristics of an object
- Fields are also known as attributes, or instance variables



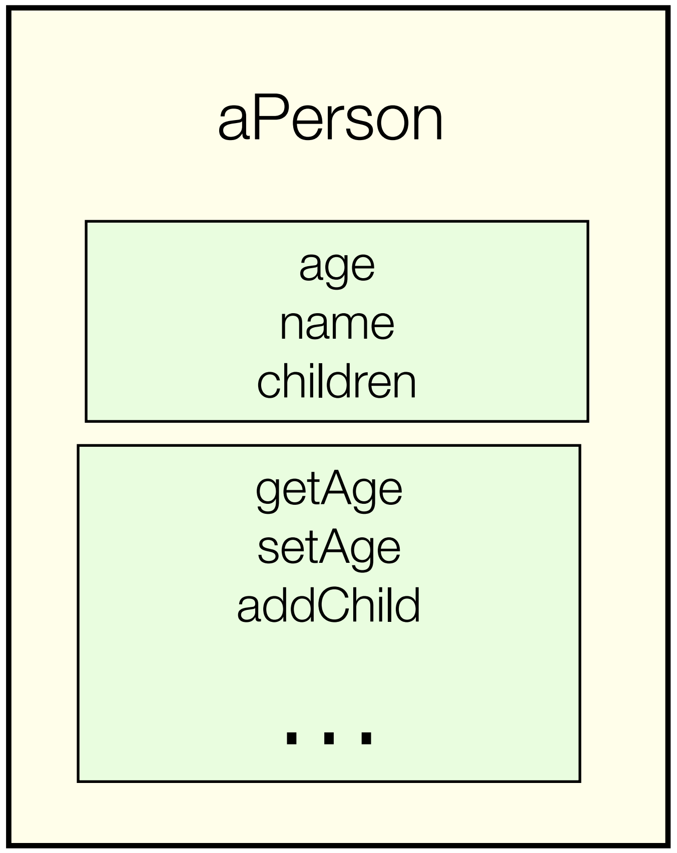
Object-Oriented Principle: Encapsulation

- Objects hide implementation of the messages behind their public protocols
  - Object's internal implementation is accessed by that object only
- Encapsulation is also known as implementation hiding



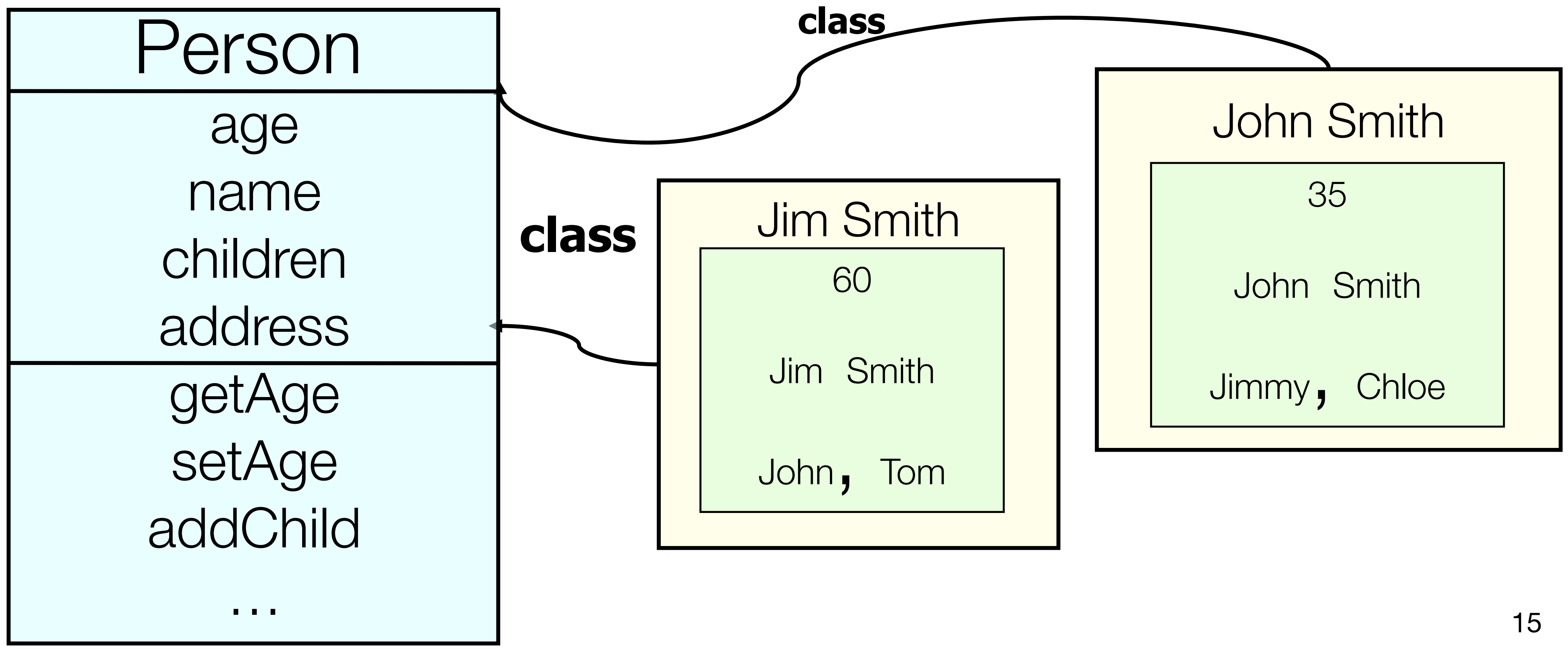
Classes

- Classes are:
  - Factories for creating objects
  - Template for the same kind of objects that describes their state and behavior
  - Code repository for objects
- Classes define objects (by defining their state and behavior) and their type



Instances

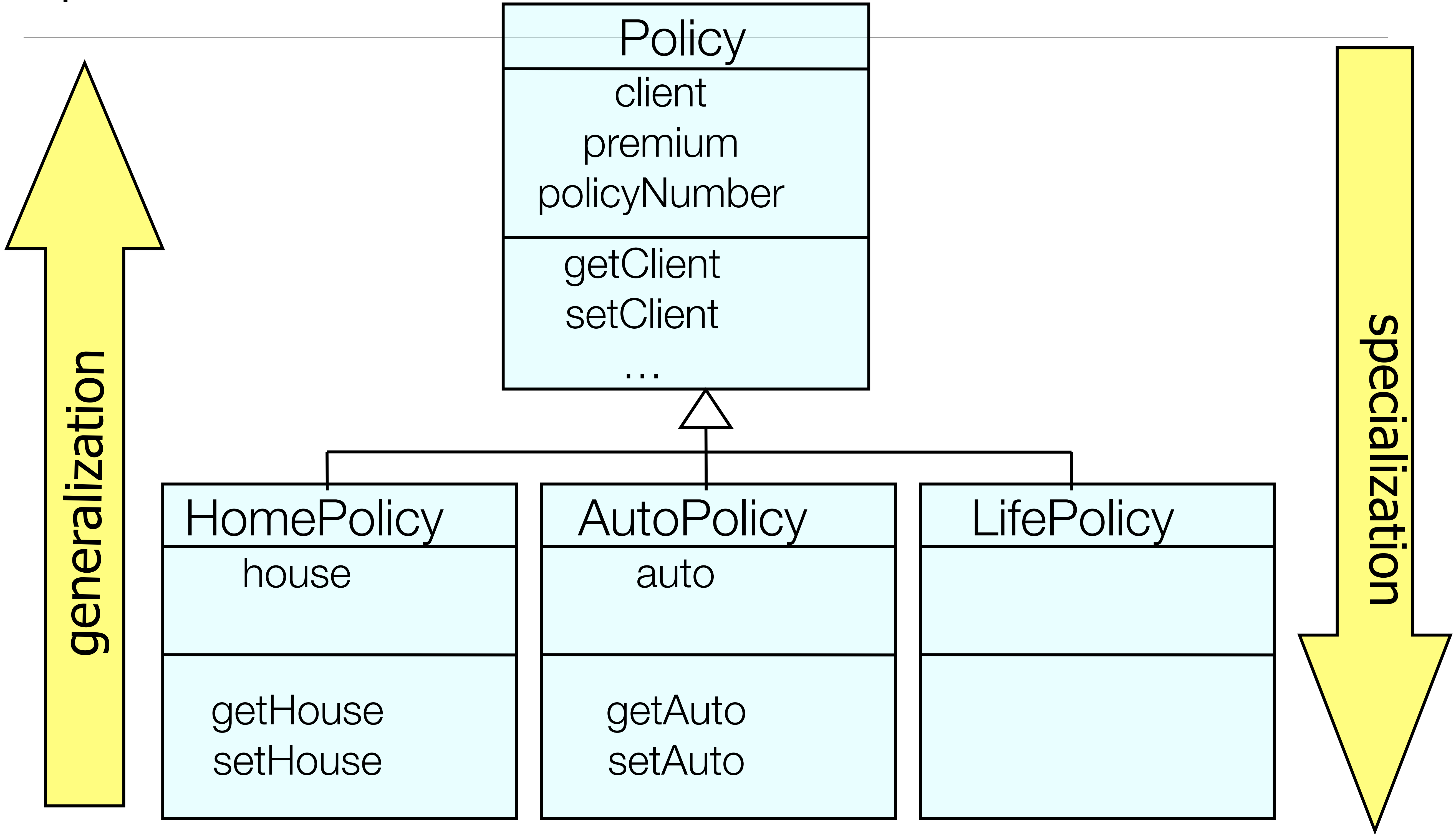
- Every object is an instance of some class
- All instances of same class have the same protocol
    - They have same fields and same methods that are defined by the class



Object-Oriented Principle: Inheritance

- Some classes may share commonalities
  - For example **`HomePolicy`**, **`AutoPolicy`**, **`LifePolicy`** classes may all have same state and behavior
- Instead of repeating commonalities in each class, we can abstract them in a common place
  - These commonalities can be stored in a super class
  - Each subclass inherits state and behavior from its superclass

Specialization and Generalization



Why Inheritance?

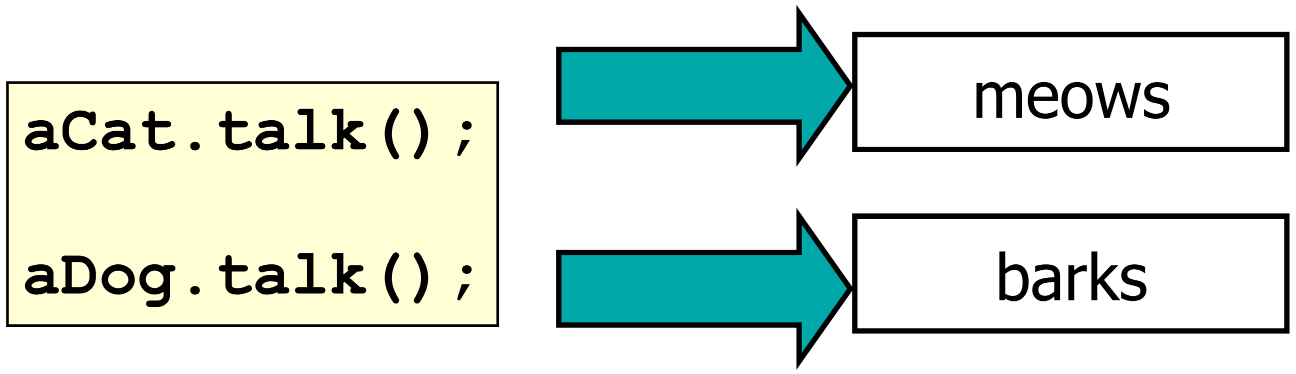
- Inheritance represents real-world modeling
    - Some objects are special cases of other objects
- Inheritance promotes reuse and extensibility
    - Same data and behavior is shared among objects of different types (different class)
    - New data and new behavior that is common for those objects is easier to add

Object-Oriented Principle: Polymorphism

- Polymorphism
  - different objects respond to the same message in different ways
  - For example when asked to talk a dog barks, and a cat meows
- It is often supported by method overriding
  - Overriding means that subclass may implement the same method as superclass, but with different code
  - **`toString()`** method in the **`Object`** class is an example of often overridden method

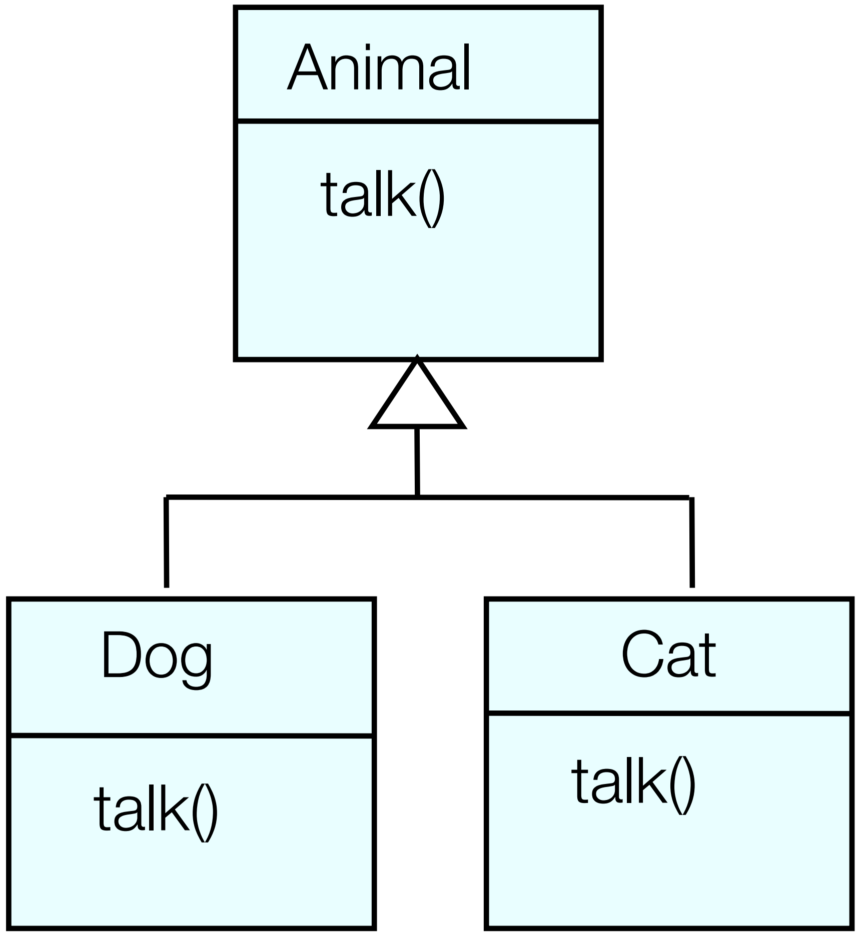
Overriding Example

- Consider **`Animal`** class:
  - **`Dog`** and **`Cat`** as subclasses
- All **`Animal`** objects should know how to talk



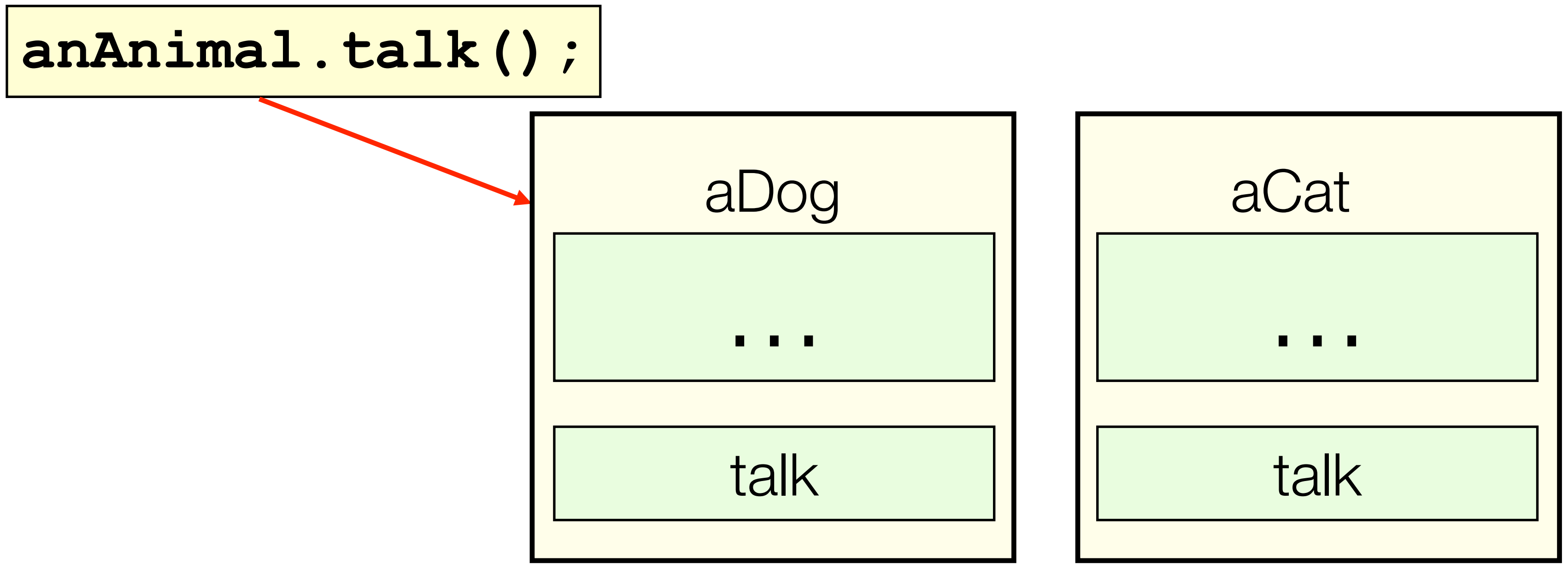
```
aCat.talk();

aDog.talk();
```



Dynamic Binding

- Dynamic binding represents runtime method binding
  - It is runtime binding of method invoked in the message to the method that implements that message
  - For example:::



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit