Object-Oriented Analysis

Lecture 2 : Class Diagram

Nga Nguyen, EISTI

Recall : Use Case Diagram

- How do you model AREL ?

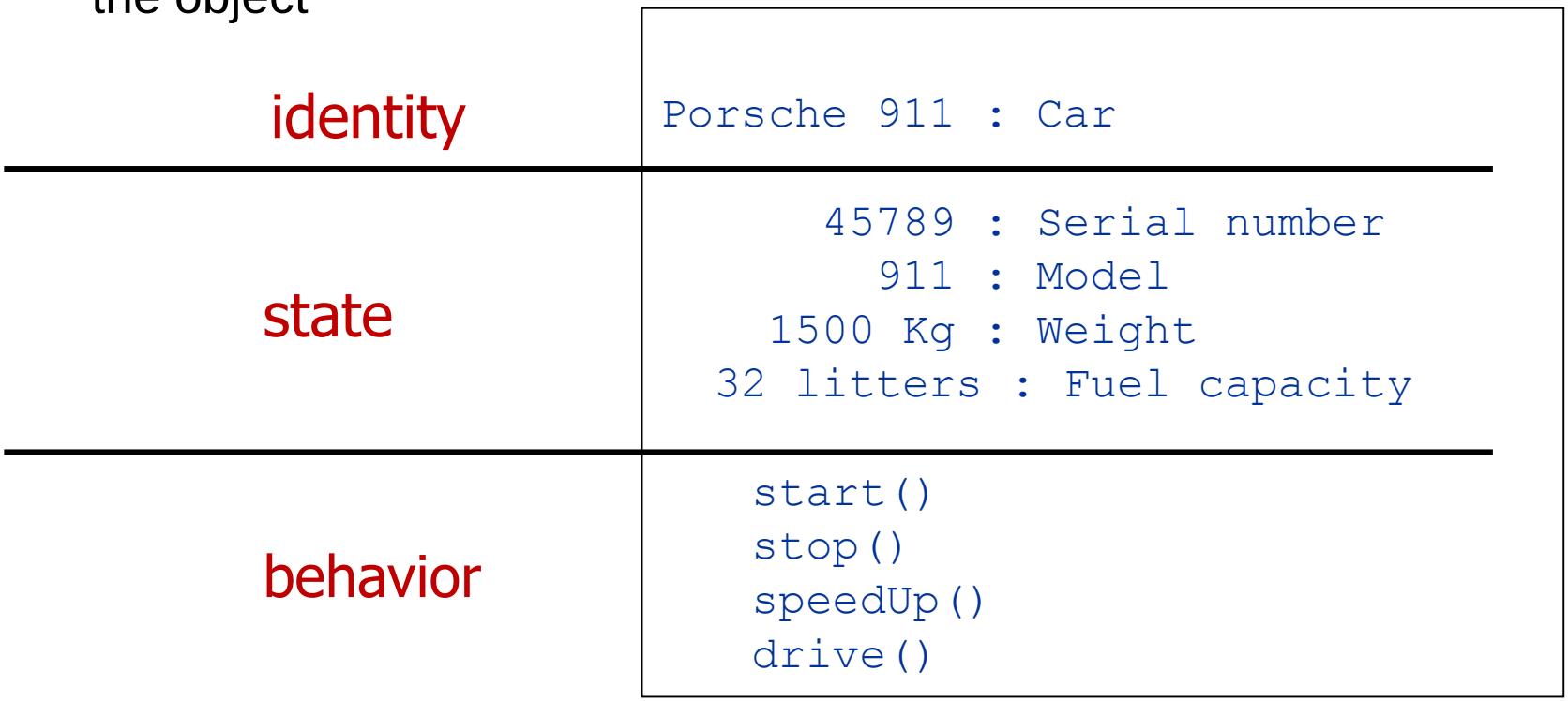
Object-Oriented Analysis

- Model real world entities as objects which are encapsulated along with their attributes and behavior.
- An object can be :
  - a living reality like a human being, an animal, ..
  - a material reality like a car, a bottle, ...
  - an immaterial or abstract reality such as an idea, a debt, …

Object

An object is composed of 3 parts :

- identity : to distinguish it from another object
- state : set of attributes' values
- behavior : set of methods that describe what we can do with the object



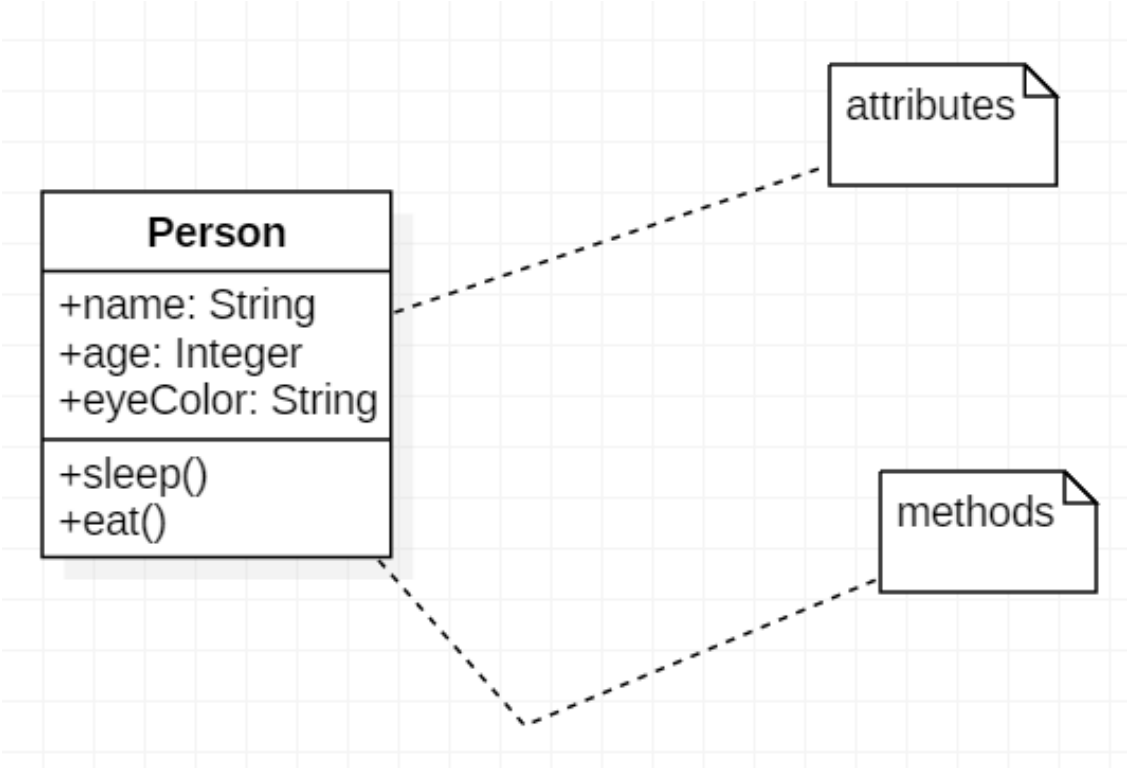
State and Behavior

- The values of the different attributes determine the state of the object.
- An object must be in a consistent state (*weight > 0*)
- The behavior of an object depends on its actual state
- The state of an object can only be changed by its methods =>

encapsulation

Class

- A class is a common description of objects
- An object is an instance of a class

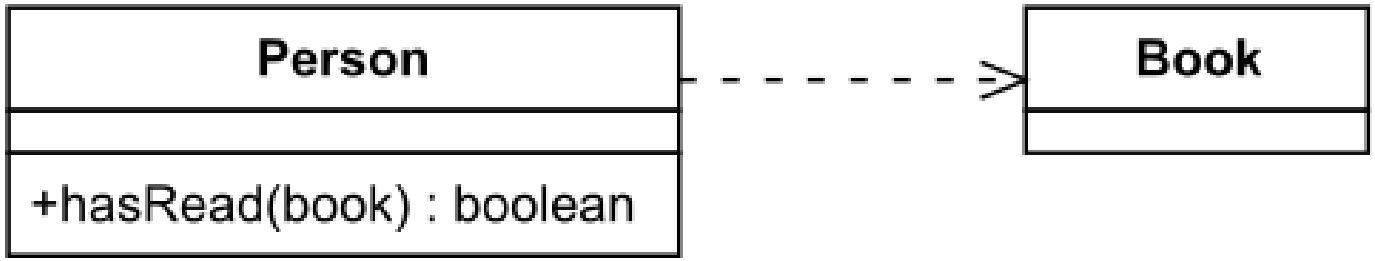


Class Diagram

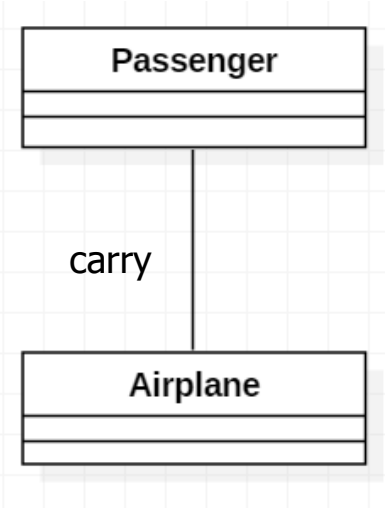
- A collection of elements that shows the static structure of the system
- Classes : name, attributes, operations
- Relationships between classes :
  - Dependency
  - Association
  - Aggregation
  - Composition
  - Inheritance

Dependency

- A dependency is a weak relationship between classes
- Class A depends on class B if A uses B in one of its operations

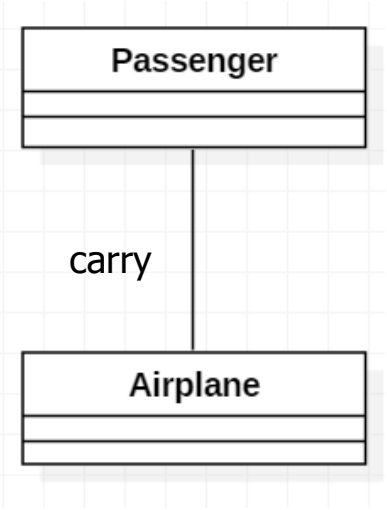


Association

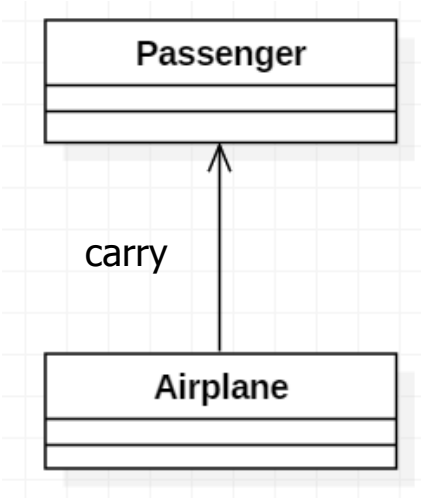


Association

Association

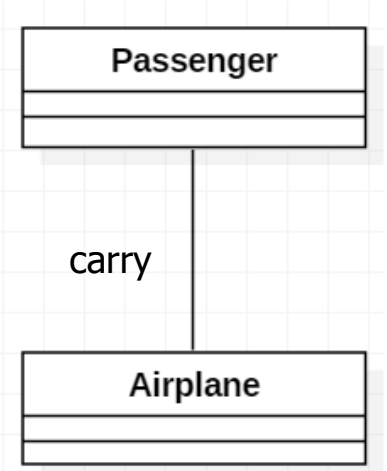


Association

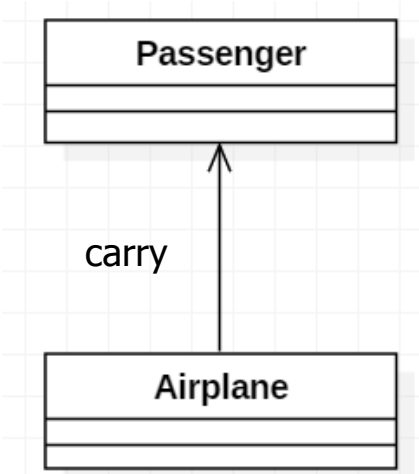


Directed association

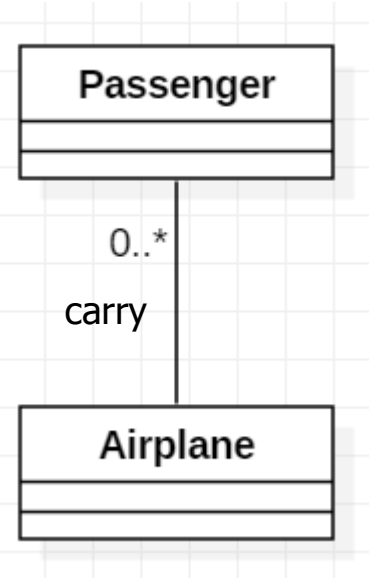
Association



Association



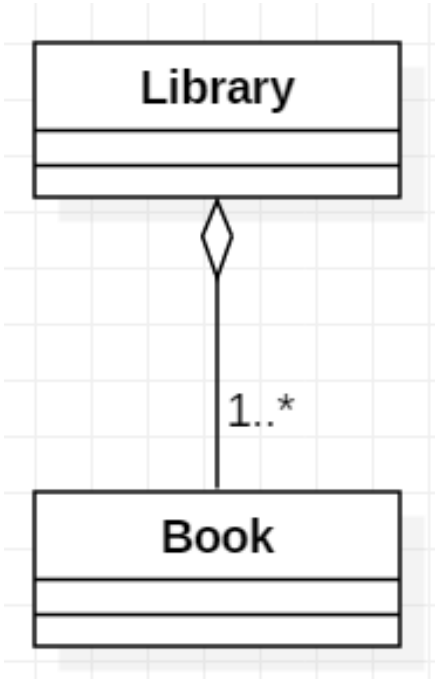
Directed association



Multiplicity

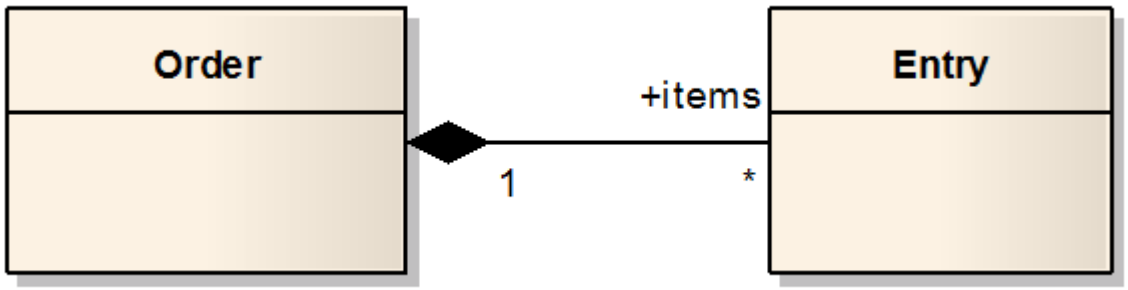
Aggregation

- « part of » relationship
- Stronger than a normal association
- The contained classes are not strongly dependent on the lifecycle of the container



Composition

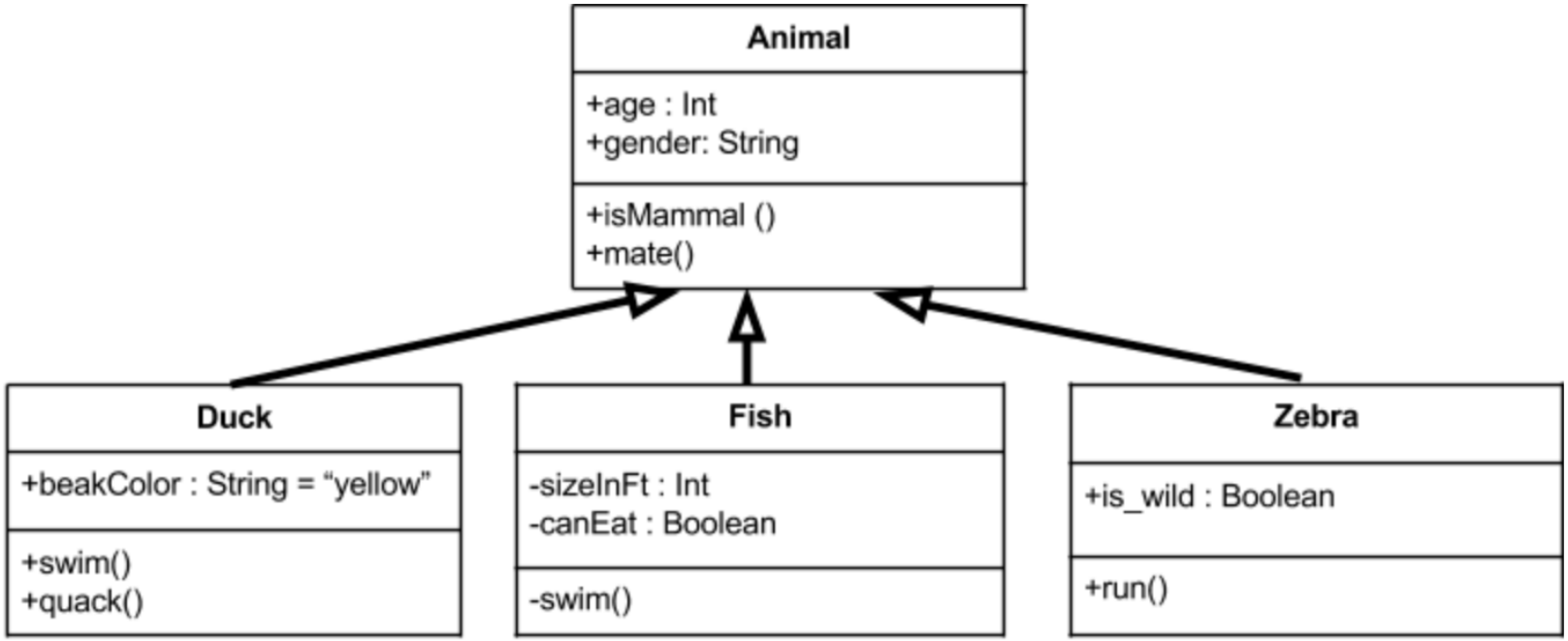
- « part of » relationship
- Stronger than an aggregation
- The contained class will be obliterated when the container class is destroyed.



Inheritance

- « is a » relationship
- The child/sub class is a specific type of the parent/super class. The child class :
  1) has the same descriptions (attributes, operations, associations) as its parent class
  2) adds descriptions (attributes, operations, associations) that are specific to it.

Inheritance



https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd8

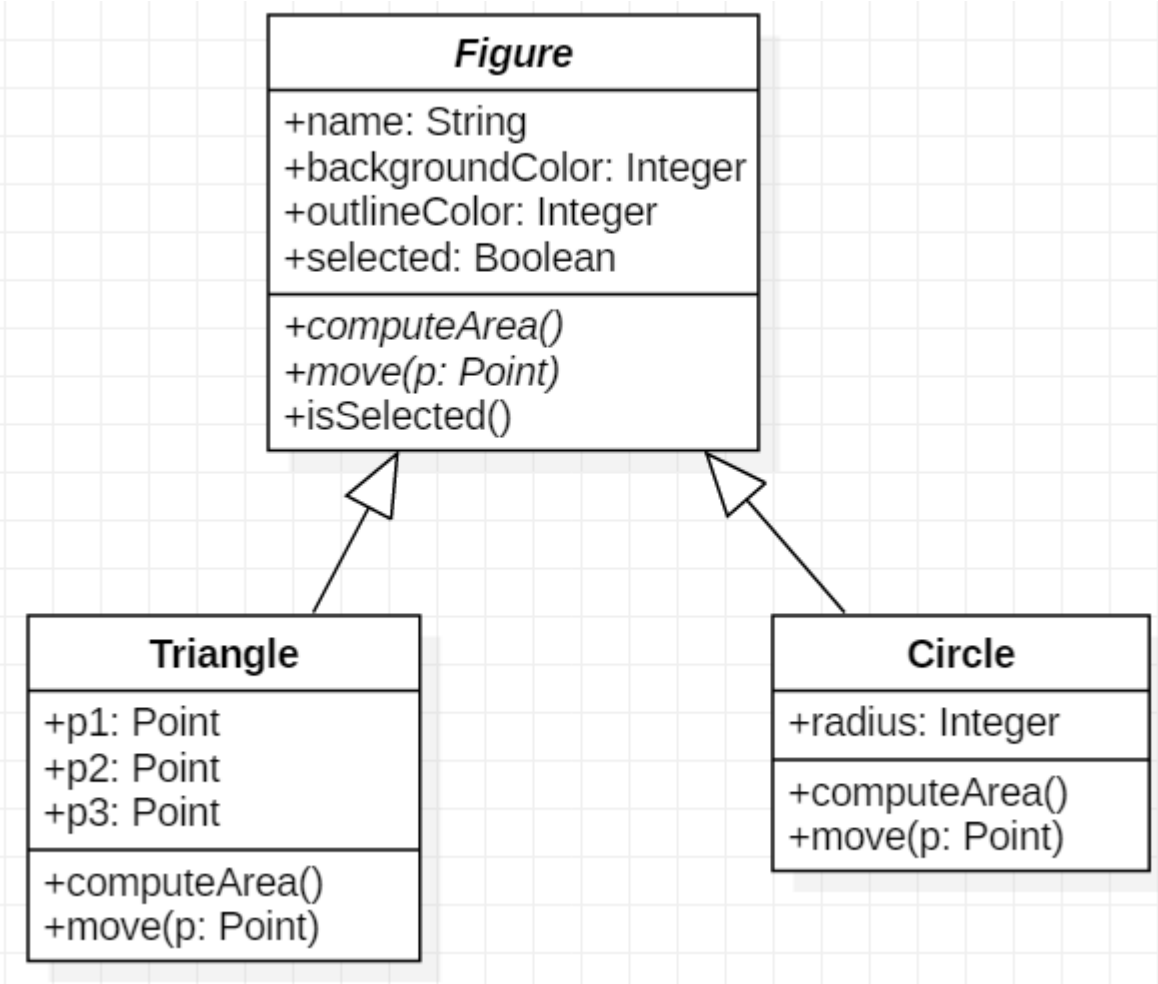
Inheritance : remarks

- In a child class, some operations of the parent class can be redefined
- A semantic trap :
  - Toto is a canary
  - A canary is a bird

Inheritance and Abstract class

- Some classes may not be fully defined to give rise to instances. They are *abstract classes*.
- They will only be used as parent classes in an inheritance.

Inheritance and Abstract class



Inheritance and Abstract class

- The class *Figure* is abstract. The instances are either triangles or circles but not figures.
- *computeArea()* and *move()* are *abstract methods* of Figure class. An **abstract method** has no implementation.

- The two classes Triangle and Circle have some common behaviors but the implementation can be different =>

  polymorphism

Four Object-Oriented Concepts

- State + behavior
- Encapsulation
- Inheritance
- Polymorphism