Week-9: Conceptual Modelling

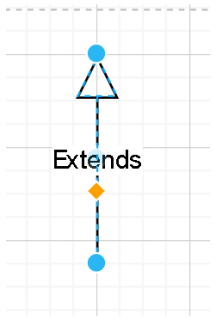
Inheritance

Interface

Liskov

Association class

Inheritance



Definition

- Representation of "**is-a**" relationships
- To create a taxonomy of categories
- To structure and organize a set of classifiers (classes)
- Parent-Child / Superclass-subclass

Meaning

- In a taxonomy, a class has a parent->a class is a parent
- Tree structure

Symbol

- The white triangle in the parent and a line connecting to the subclass

Application

- To create taxonomies of classifies /set of categories
- **Reuse data (attributes) and behavior (methods/functionality)**

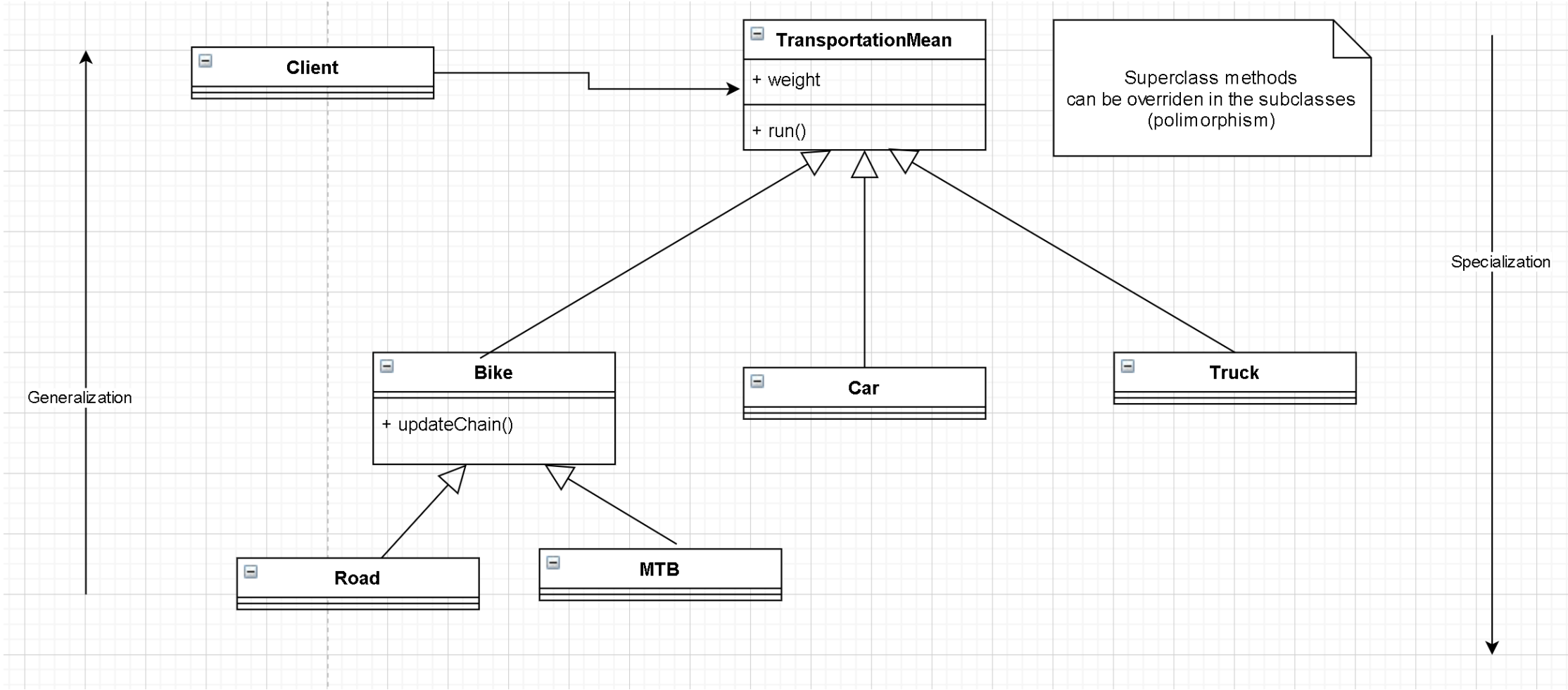
Example

- Animal
  - Mammal
    - Lion
    - Dog
    - Horse
    - …
  - Not mammal
    - Bird
      - Eagle
    - Bug
  - ..

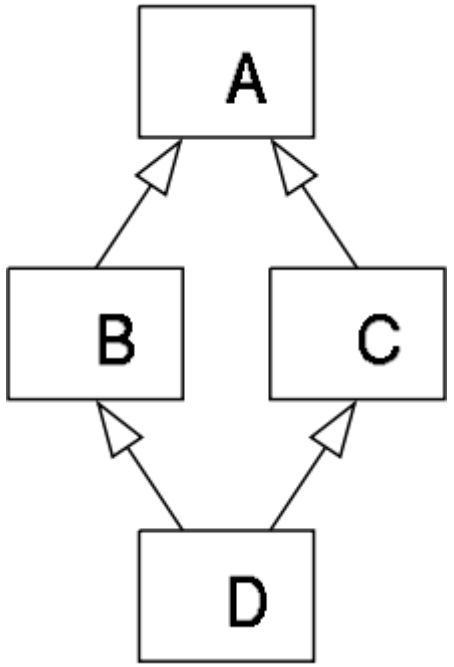
Characteristics

- Transitive
- Asymmetric

Inheritance example

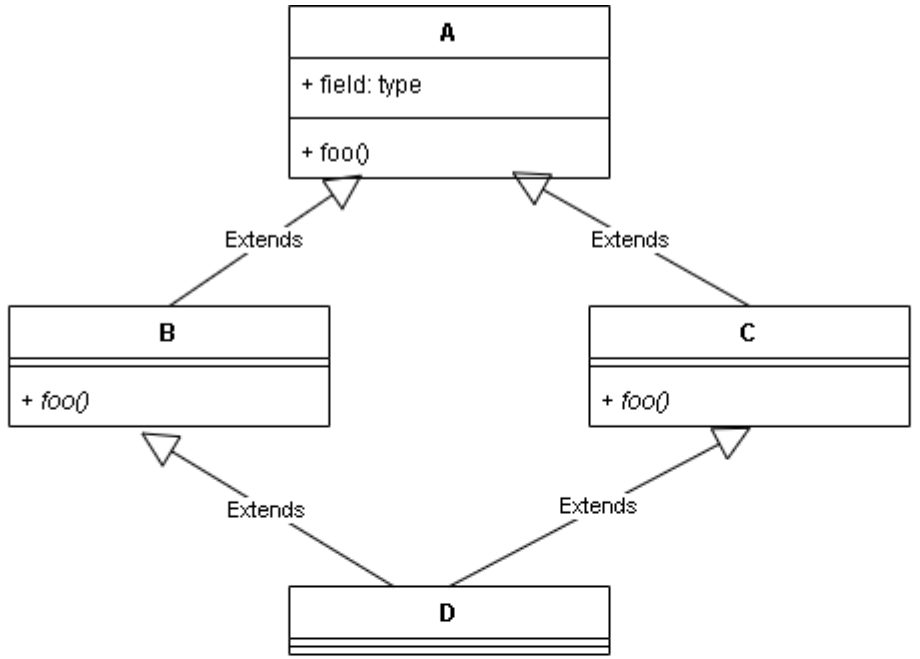


Inheritance: the diamond problem



- Multiple inheritance
  - Ambiguity
- Wrong application:
  - Reuse of data →OK
  - …behavior→Careful! → misleading behaviors
  - →Composition vs Inheritance

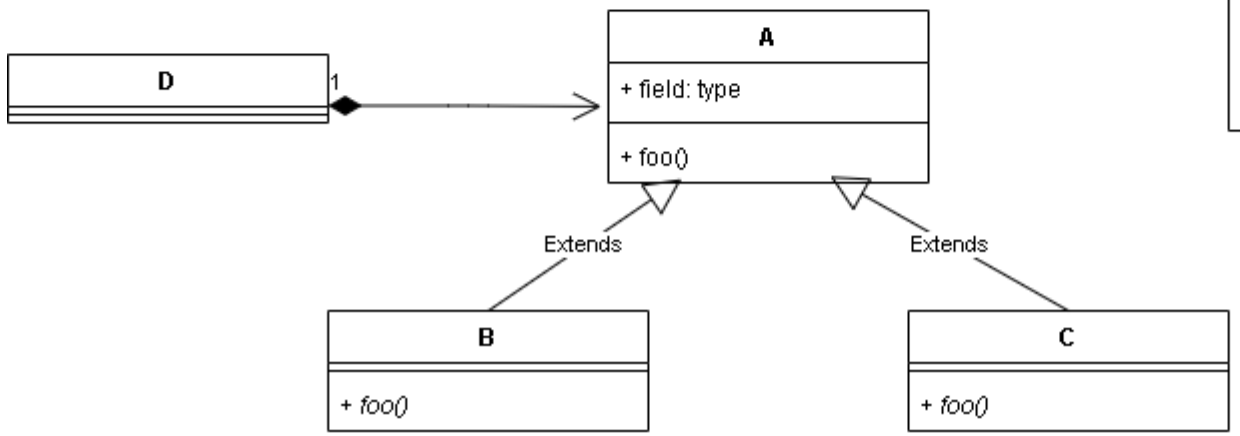
Example



d = new D()
d.foo() ???
d.foo()-->b.foo()
d.foo()-->c.foo()

- Multiple inheritance-->OK for modelling
- Multiple inheritance-->NOK for implementation
- Performance: an instance of D implies 3 other instances (A,B and C)
- Inheritance-->Delegation

d.foo()-->return a.foo()



Discussion: composition vs inheritance

- https://www.thoughtworks.com/insights/blog/composition-vs-inheritance-how-choose

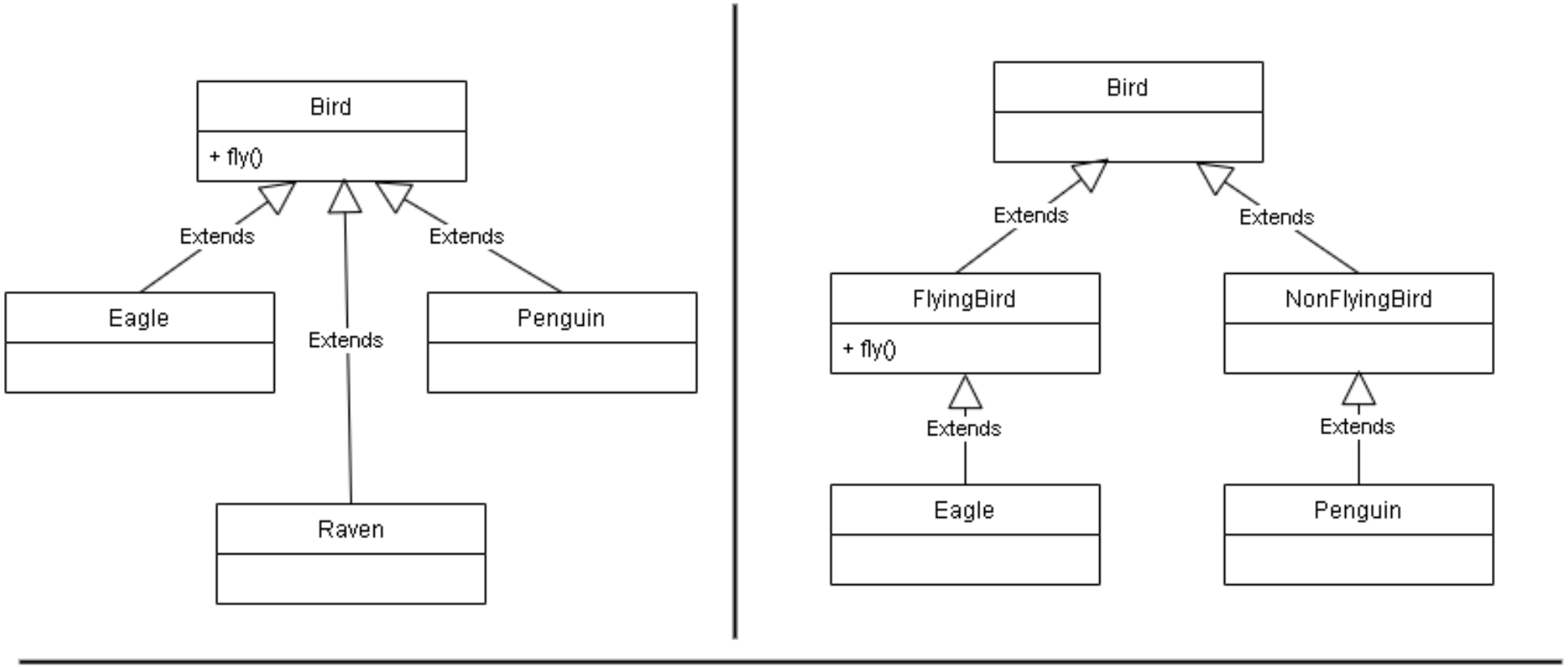
Interface

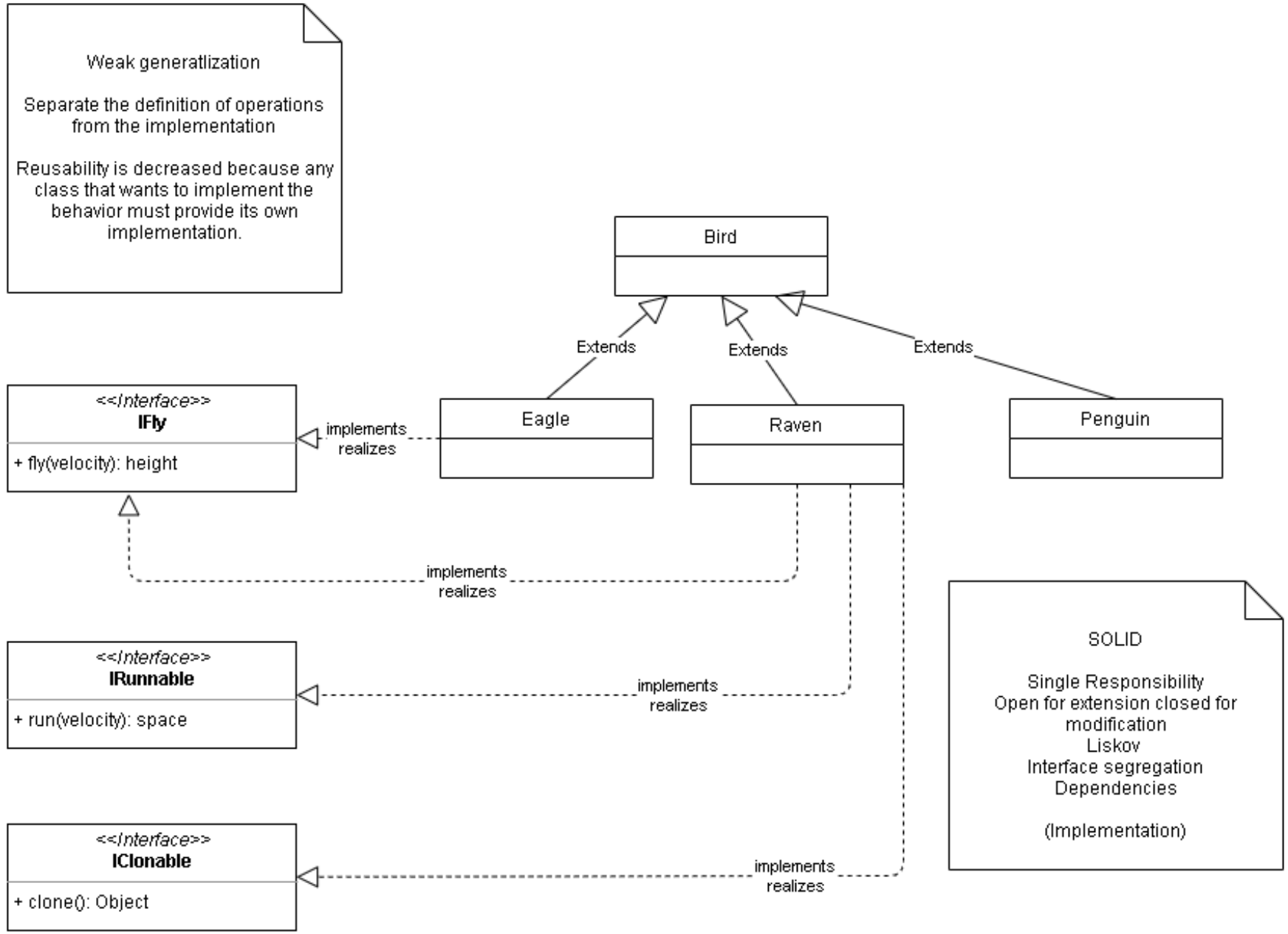
- **Separate behavior from implementation**
-
- For example, say we have a car class and a scooter class and a truck class. Each of these three classes should have a start_engine() action. How the "engine is started" for each vehicle is left to each particular class, but the fact that **they must** have a start_engine action is the domain of the interface.

SOLID principles: Liskov substitution principle (only)

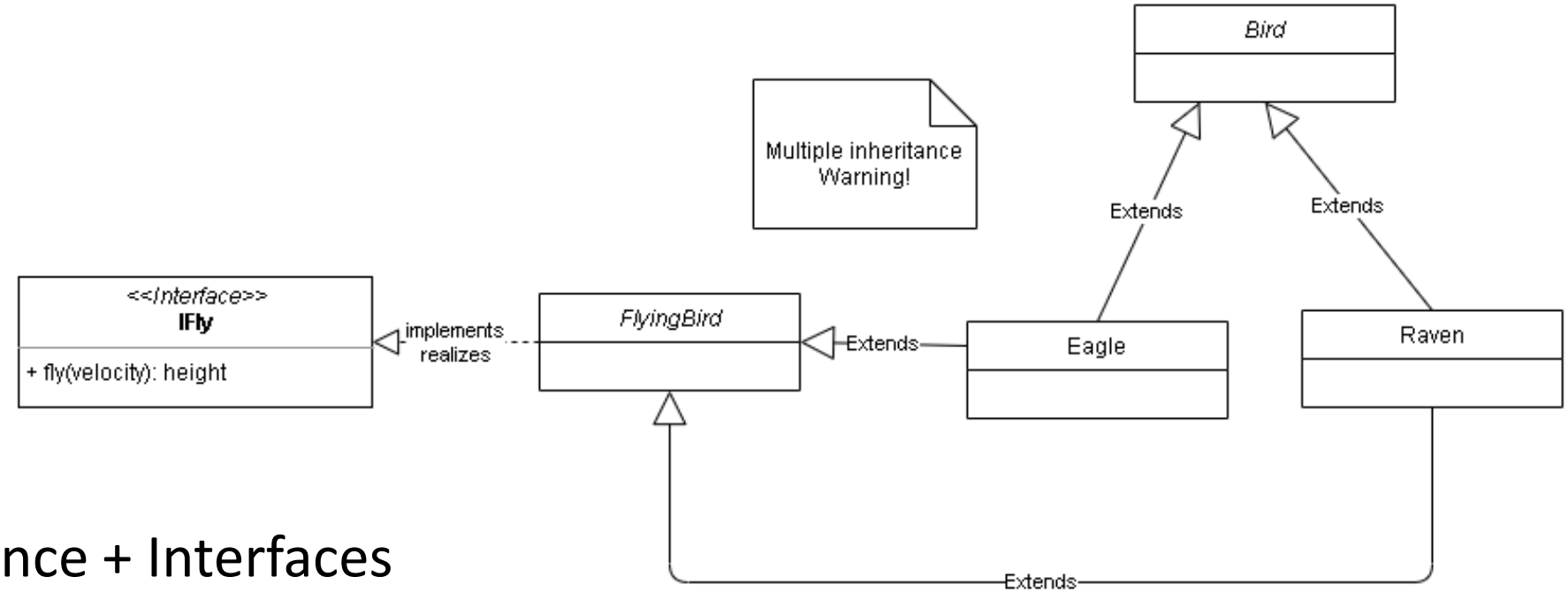
SRP	The Single Responsibility Principle	*A class should have one, and only one, reason to change.*
OCP	The Open Closed Principle	*You should be able to extend a classes behavior, without modifying it.*
LSP	**The Liskov Substitution Principle**	***Derived classes must be substitutable for their base classes.***
ISP	The Interface Segregation Principle	*Make fine grained interfaces that are client specific.*
DIP	The Dependency Inversion Principle	*Depend on abstractions, not on concretions.*

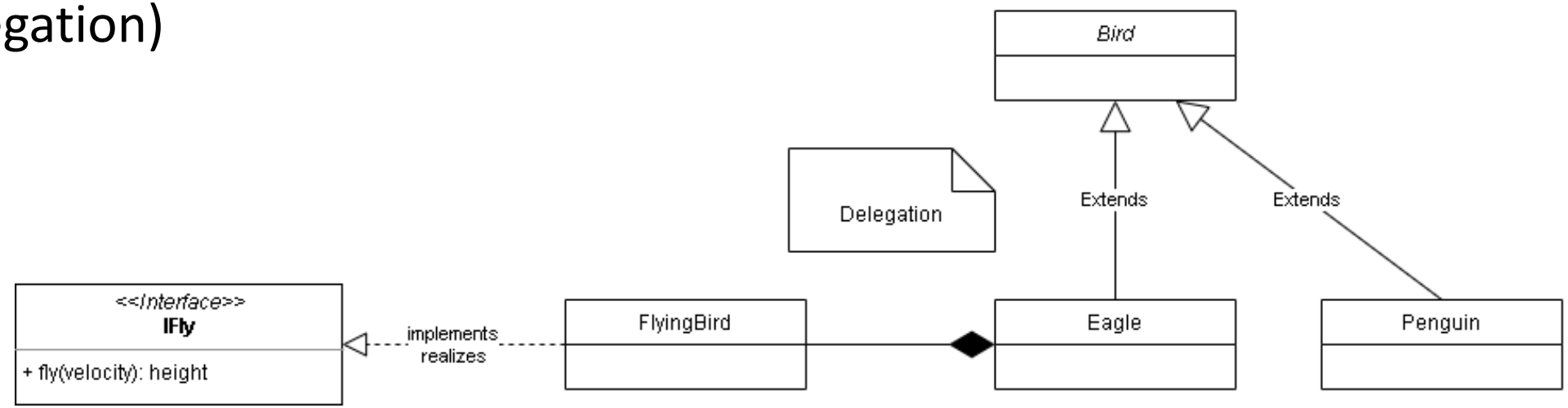
Example: from inheritance to interfaces + composition




Weak generatlization
Separate the definition of operations from the implementation
Reusability is decreased because any class that wants to implement the behavior must provide its own implementation.
Bird
Extends
Extends
Extends
<<Interface>>
IFly
+ fly(velocity): height
implements
realizes
Eagle
Raven
Penguin
implements
realizes
<<Interface>>
IRunnable
+ run(velocity): space
implements
realizes
<<Interface>>
IClonable
+ clone(): Object
implements
realizes
SOLID
Single Responsibility
Open for extension closed for modification
Liskov
Interface segregation
Dependencies
(Implementation)

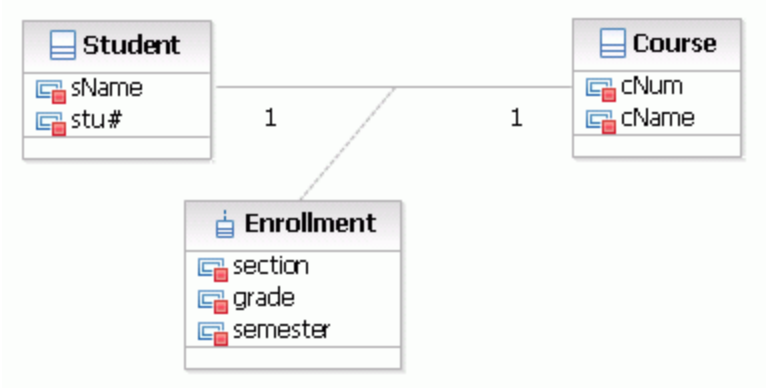
Inheritance + Interfaces
+ Composition
(delegation)

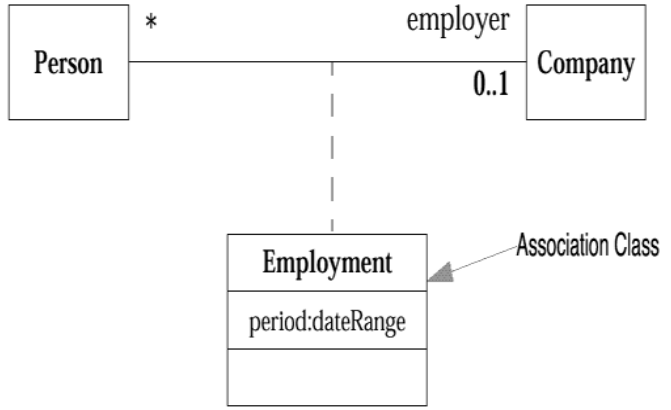




Association class

- In UML diagrams, an association class is a class that is part of an association relationship between two other classes.
- You can attach an association class to an association relationship to provide additional information about the relationship. An association class is identical to other classes and can contain operations, attributes, as well as other associations.
- **Association classes** allow you to add attributes, operations, and other features to associations.

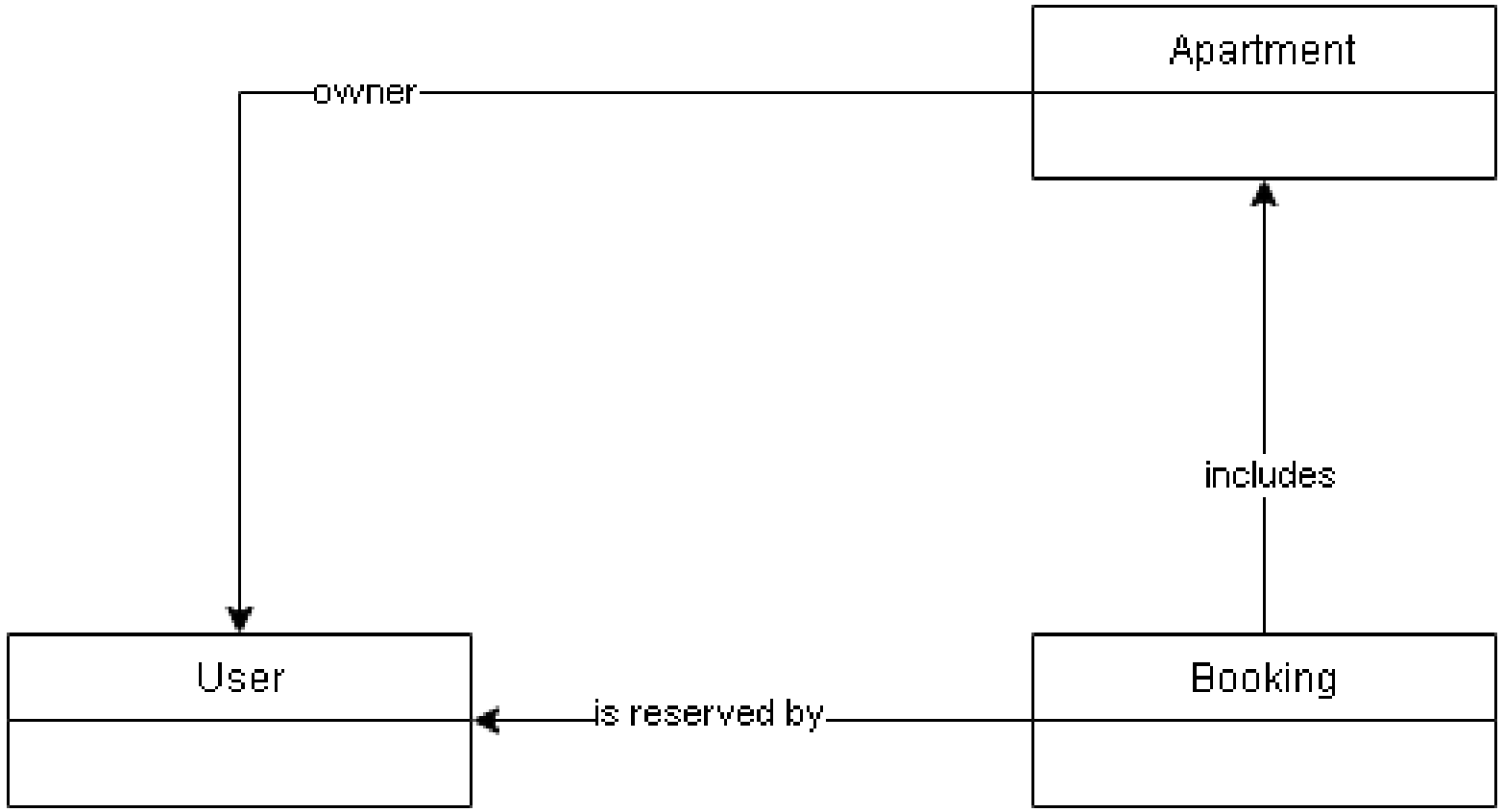




Class vs datatype (**conceptual** modelling)

	Class **E.g. Car**	**Datatype** **E.g. integer**
Scope	**Domain-specific**	**General-purpose**
Population	Finite and variable E.g. Car.color	Finite/Infinite but constant E.g. Integer numbers→Infinite, 5→constant
Compare	By reference (Are they same?)	By value (Are they equal?)
Associated	Unidirectional/Bi-directional	Unidirectional
Create	Copy or clone the instance	Copy the value
Structure	A compilation of different attributes and methods→Complex	Stores a value→simple/complex E.g. Date (day, month, year)←→timestamp (long)

Class diagram (v1): static and logical view of the AirBnB booking system



Examples

- 1, 4, 5, 6, 7