

Object-Oriented Approaches to Programming \sim Aggregation, Composition, Inheritance \checkmark

Didier Verna

didier@didierverna.net

@didierverna



didier.verna



Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

Build 2024-11-03 12:30:26+01:00



\varTheta 🖯 🖯 Outline

Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

OAP / Aggregation, Composition, Inheritance





Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

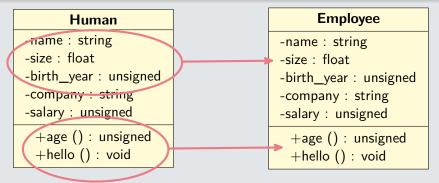
OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



⊖ ⊖ ⊖ Copy / Paste

Example



Very bad approach!

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

Build 2024-11-03 12:30:26+01:00





Relations between Classes

Aggregation

Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

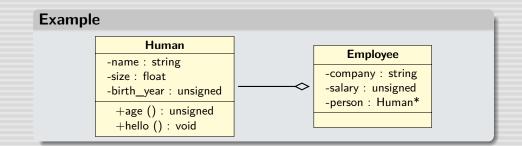
OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



e e e Aggregation

- A kind of inclusion
 - ▶ Aggregate: maintenance of references / pointers to aggregated objets



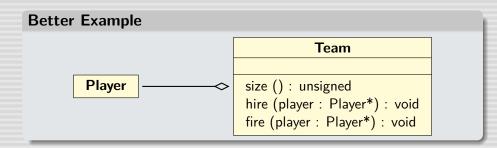
Not very fit for this example

- Relation "set / elements" (transitive)
- The aggregated survives its aggregate



⊖ ⊖ Aggregation

- A kind of inclusion
 - ► Aggregate: maintenance of references / pointers to aggregated objets





e e e Aggregation

```
A kind c C++
     ► Agg class Player {};
        Bet using player_set = std::unordered_set<const Player*>;
             class Team
             public:
              player_set::size_type size () const;
               void hire (const Player& player);
               void fire (const Player& player);
             private:
              player_set members;
           • }:
             player set::size type Team::size () const { return members.size (); }
             void Team::hire (const Player& player) { members.insert (&player); }
             void Team::fire (const Player& player) { members.erase (&player); }
```

OAP / Aggregation, Composition, Inheritance



⊖ ⊖ Aggregation

- A kind of inclusion
 - ▶ Aggregate: maintenance of references / pointers to aggregated objets

```
Java
Bet
public class Player {}

public class Team
{
    public Team () { members = new HashSet<Player> (); }
    public int size () { return members.size (); }

    public void hire (Player player) { members.add (player); }
    public void fire (Player player) { members.remove (player); }
    private HashSet<Player> members;
}
```

OAP / Aggregation, Composition, Inheritance





Relations between Classes

Aggregation

Composition

Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

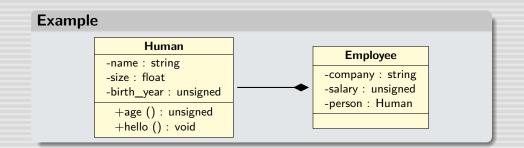
 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● ● Composition (Composite Aggregation)

- A stricter form of inclusion
 - The aggregated does not survive its (unique) aggregate



Still not very fit for this example

Access to the Human part is (still) indirect



● ● **Composition** (Composite Aggregation)

- ► A stricter form of inclusion
 - The aggregated does not survive its (unique) aggregate

Better Example		
	Head	Body

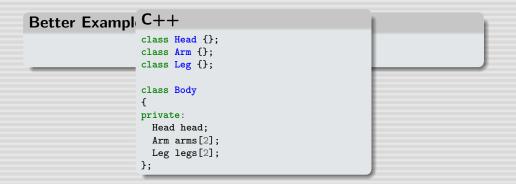
OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● **Composition** (Composite Aggregation)

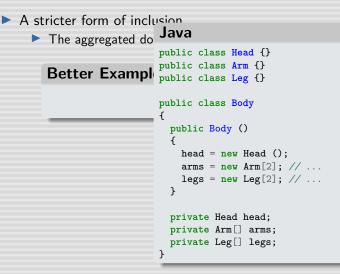
- A stricter form of inclusion
 - The aggregated does not survive its (unique) aggregate



OAP / Aggregation, Composition, Inheritance



● ● **Composition** (Composite Aggregation)



OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0





Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

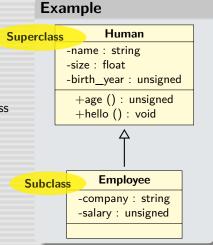


● ● Inheritance / Derivation

- An even stricter form of inclusion
 - Aggregated contents directly incorporated into the class
- Best solution here
 - No risk related to manual duplication
 - No intermediate object (aggregated)
 - The contents of class Human belongs *implicitly* to class Employee as well (except for constructors / destructors)

Class Employee "inherits" or "derives" from class Human

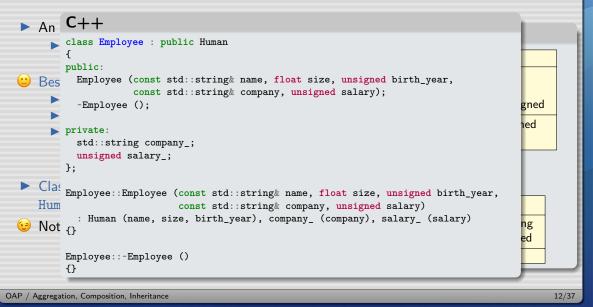
😉 Not far from (automatic) cut'n paste



OAP / Aggregation, Composition, Inheritance



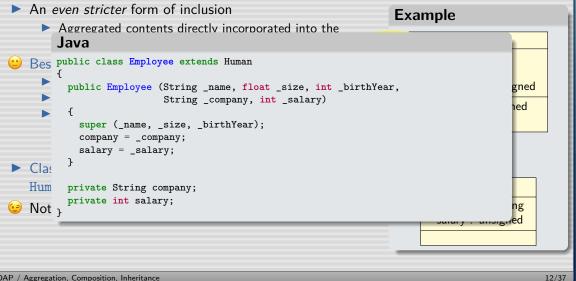
● ● Inheritance / **Derivation**



Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● ● Inheritance / Derivation



OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



\varTheta 🖯 🖯 Plan

Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

OAP / Aggregation, Composition, Inheritance

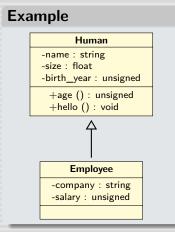
Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● **Characteristics of Inheritance**

- Aggregation: "has a" relationship
 - A team has a player, a body has a head, etc.
- Inheritance: "is a" relationship
 - An employee is a human
- Consequence
 - Inheritance looks like sub-typing
 - Every employee can be seen as a simple human
 - The actual class of an object does not need to be known at compile-time any longer
 - A But "looks like" only!

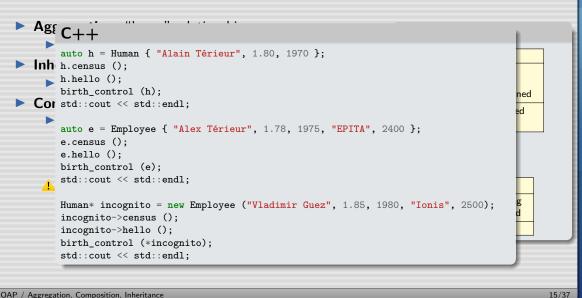
Cf. the Liskov substitution principle [Liskov, 1988]



OAP / Aggregation, Composition, Inheritance



⊖ ⊖ ⊖ Characteristics of Inheritance

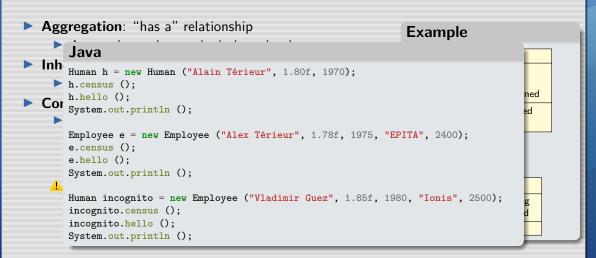


OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



O Characteristics of Inheritance



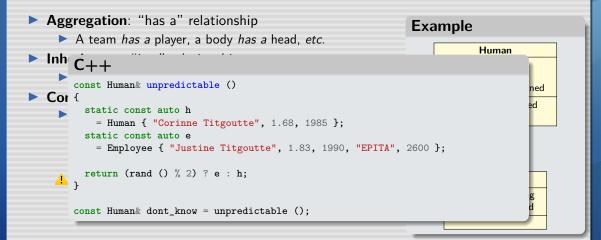
OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

Build 2024-11-03 12:30:26+01:00



O O Characteristics of Inheritance

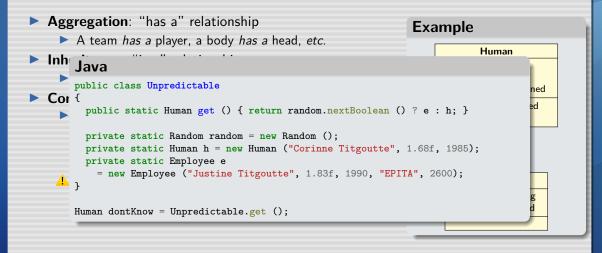


OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



000 Characteristics of Inheritance



OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



\varTheta 🖯 🖯 Plan

Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies

Instantiation Policies Accessibility

Inheritance Problems

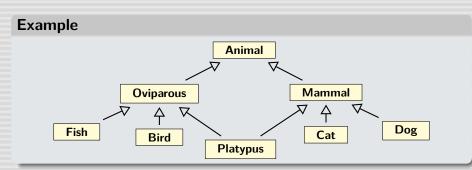
Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



O O Class Hierarchies



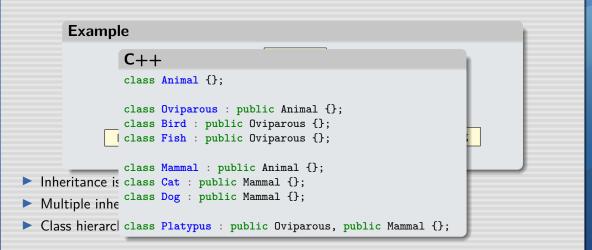
- Inheritance is a transitive relation
- Multiple inheritance (not always available): several super-classes
- Class hierarchy: oriented inheritance tree (or graph)

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● Class Hierarchies



OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

18/37

Build 2024-11-03 12:30:26+01:00



\varTheta 🖯 🖯 Plan

Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies

Instantiation Policies

Accessibility

Inheritance Problems

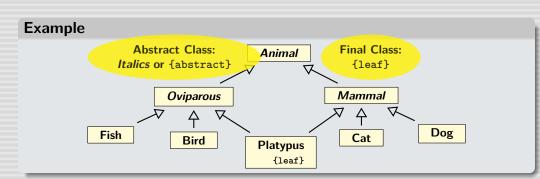
Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



••• Instantiation Policies



- Abstract Class: not instantiable
- Final Class: non derivable Additional technical benefits: safety, performance

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



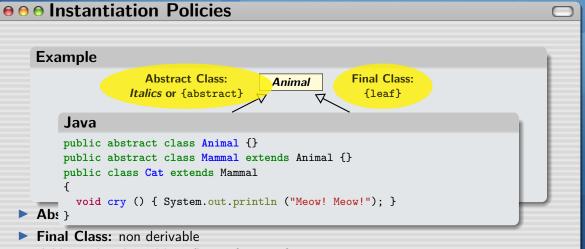
● ● ● Instantiation Policies

```
Exa C++
       class Animal
       public:
        // Make this class abstract.
        virtual void cry () const = 0;
       };
       // Also abstract.
       class Mammal : public Animal {};
Abs class Cat : public Mammal
{
Fin public:
  Adc // Not abstract anymore.
         void cry () const override { std::cout << "Meow! Meow!\n"; };</pre>
       };
```

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0





Additional technical benefits: safety, performance

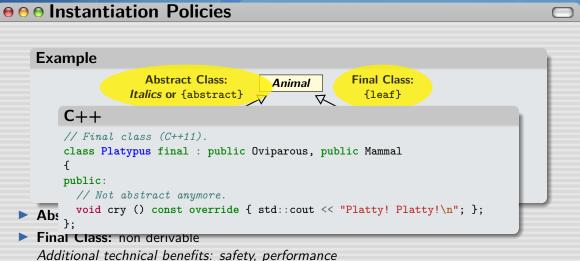
OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

21/37

Build 2024-11-03 12:30:26+01:00





OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

21/37

Build 2024-11-03 12:30:26+01:00



● ● ● Instantiation Policies Example Abstract Class: **Final Class:** Animal Italics or {abstract} {leaf} Ovinarous Mammal Java public final class Platypus extends Animal void cry () { System.out.println ("Platty! Platty!"); } Abstract Class: not instantiable Final Class: non derivable

Additional technical benefits: safety, performance

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



\varTheta 🖯 🖯 Plan

Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

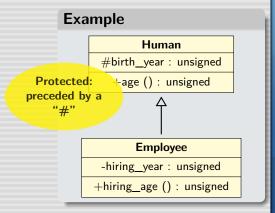
 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● ● Accessibility and Inheritance

- Protected Attribute / Method: acces restricted to the class sub-hierarchy
- Remark: Public interface + protected attributes = closest form to the principle of sub-typing

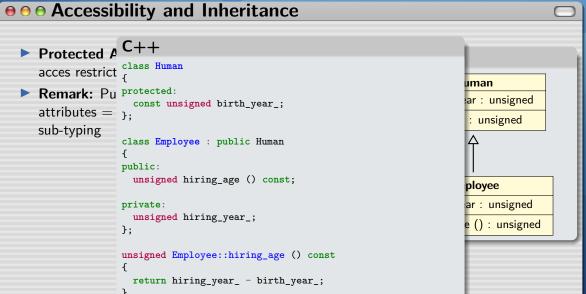


OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

Build 2024-11-03 12:30:26+01:00



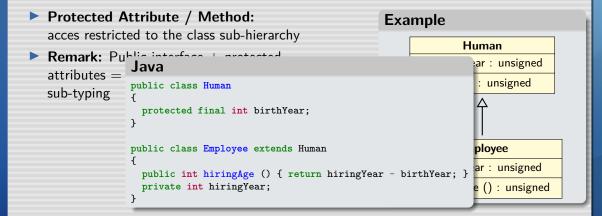


OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



● ● ● Accessibility and Inheritance



OAP / Aggregation, Composition, Inheritance

23/37

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

OAP / Aggregation, Composition, Inheritance



Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation

Ambivalence of Inheritance Multiple Inheritance

 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



O O Inheritance and Instantiation

🙏 Manipulate the relation "is a" with caution

- An employee is a (kind of) human
- Alex is an employee (in particular)



OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

26/37

Build 2024-11-03 12:30:26+01:00



Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

Inheritance and Instantiation

Ambivalence of Inheritance

Multiple Inheritance

 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

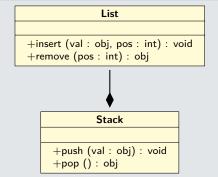


O O Ambivalence of Inheritance

Problems:

- 1. Exposition of the implementation
- 2. Inheritance of the list's interface
- Two effects of sub-classing:
 - 1. Inheritance of implementation *code reusability*
 - 2. Inheritance of interface *semantics, sub-typing*
- Implementation inheritance entails interface inheritance
- Favor composition over inheritance A stack is not a list

Example



OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



• • • Remark: "Private" Inheritance

```
C++
class List
{
public:
    void insert (obj val, int pos);
    obj remove (int pos);
};
class Stack : public private List
{
public:
    void push (obj val) { insert (val, 0); };
    obj pop () { return remove (0); }
};
```

"Is implemented in terms of" relation

Favor composition, still

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0





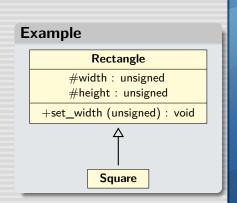
● ● Inheritance by Restriction

The square rectangle (elliptic circle) problem

- A square is a rectangle...
- …although with static constraints…
- …and dynamic ones

Differential Programming:

- Inherit in an additive (not restrictive) manner
- Problem mostly related to mutation
- Cf. Liskov substitution principle [Liskov, 1988]



Build 2024-11-03 12:30:26+01:00



Relations between Classes

Aggregation Composition Inheritance

Characteristics of Inheritance

Class Hierarchies Instantiation Policies Accessibility

Inheritance Problems

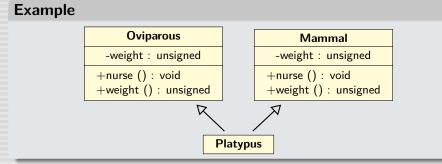
Inheritance and Instantiation Ambivalence of Inheritance Multiple Inheritance

 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0



000 Multiple Inheritance Ambiguities



- Which method(s) to choose (nurse) ?
- Why would there be several (weight) ?
- Those remarks apply to attributes as well

OAP / Aggregation, Composition, Inheritance



000 Multiple Inheritance Ambiguities

```
Exa C++
            class Oviparous
            public:
              void nurse () const { std::cout << "I brood my eggs.\n"; }</pre>
            };
            class Mammal
            public:
               void nurse () const { std::cout << "I suckle my offsprings.\n"; }</pre>
            }:
Which n
            class Platypus final : public Oviparous, public Mammal {};
Why wo
            Platypus platty;
Those re platty.Oviparous::nurse ();
            platty.Mammal::nurse ();
```

OAP / Aggregation, Composition, Inheritance



● ● ● Diamond Inheritance Exemple Transport Animal -weight : unsigned -speed : unsigned +weight () : unsigned +speed () : unsigned Oviparous Mammal Boat Plane R Platypus Seaplane

How many copies of the base class do we want?

- Why reason at the class level?
- 🔔 Each language has its own position...



O O Diamond Inheritance

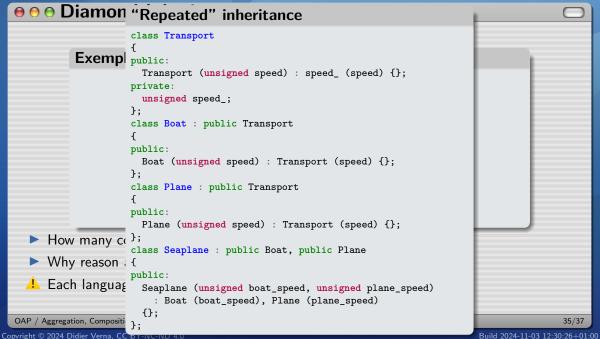
Exemp	"Shared" inheritance
	<pre>class Animal { public: Animal (unsigned weight) : weight_ (weight) {};</pre>
	<pre>private: unsigned weight_; };</pre>
	<pre>class Oviparous : public virtual Animal {}; class Mammal : public virtual Animal {};</pre>
► How many co	<pre>class Platypus final : public Oviparous, public Mammal {</pre>
► Why reason	<pre>public: Platypus (unsigned weight) : Animal (weight) {};</pre>
👃 Each languag	};

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

Build 2024-11-03 12:30:26+01:00











 OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0

36/37

Build 2024-11-03 12:30:26+01:00



\varTheta 🖯 🖯 Bibliography

Barbara Liskov. Data Abstraction and Hierarchy. *OOPSLA'87 Keynote*, 1988.

OAP / Aggregation, Composition, Inheritance

Copyright © 2024 Didier Verna, CC BY-NC-ND 4.0