

Contents(1)

- **Goal and Scope**
- **Basic Concepts on OOT**
 - Basic Concepts to represent the world
 - Basic Concepts for Reuse
 - Information Hiding Principle and Java Program
 - Superiority of OOT
- **Modeling Techniques**
 - **Static Model: Class and Association**
 - Dynamic Model: State Machine
 - Dynamic Model: Interaction Diagram
 - Concurrency Description: Active Object and Multi-thread Programming
 - Outline of UML2.0

Static Modeling

“Class and Association”

Koichiro OCHIMIZU

School of Information Science

JAIST

JAIST Koichiro Ochimizu

A Simple model of an insurance business



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu

Visibility

Invoice

+ amount : Real
+ date : Date
+ customer : String
+ specification : String
- administration : String

- **+ public** referenced from classes other than the one in which they are defined
 - It violates Information hiding principle
- **- private** cannot access it from other classes
- **# protected** can access it from sub classes

A class-scope attribute or a class variable

Invoice
+ amount : Real + date : Date = Current date + customer : String + specification : String - administration : String = “Unspecified” - <u>number of invoices : Integer</u>

- A class variable (underlined) is shared by all objects of the class

Class-scope operation

Figure
size : Size pos : Position <u>figcounter : Integer</u>
draw() <u>getCounter(): Integer</u>

- Access class-scope attributes
- Creation of objects

Visibility of Operation

Figure
size : Size pos : Position
+ draw() + scaleFigure(percent: Integer = 25) + returnPos(): Position

Signature

Car
+ registration number : String - data : CarData + speed : Integer + direction : Direction - administration : String
+ drive(speed: Integer, direction: Direction) + getData(): CarData

- Signature: a return-type, a name, zero or more parameters

UML notations

Representation of Relationships between Classes

 **Association**

 **Navigable Association**

 **Dependency**

 **Generalization**

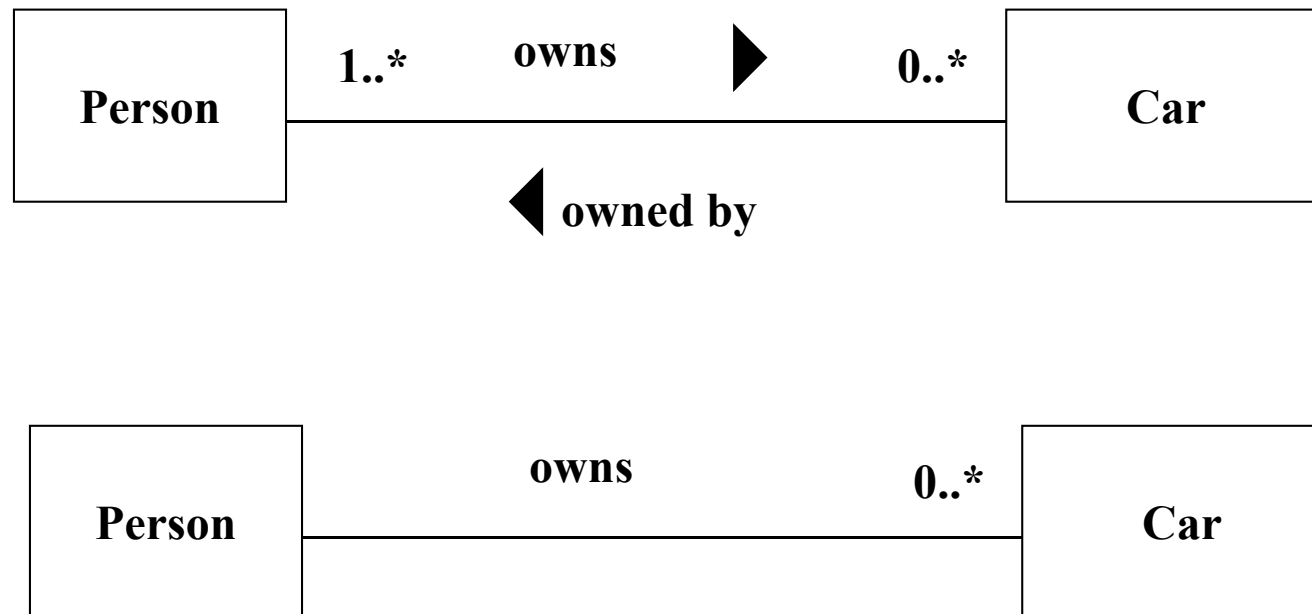
 **Aggregation: whole-part association**

 **Composition: the same life span**

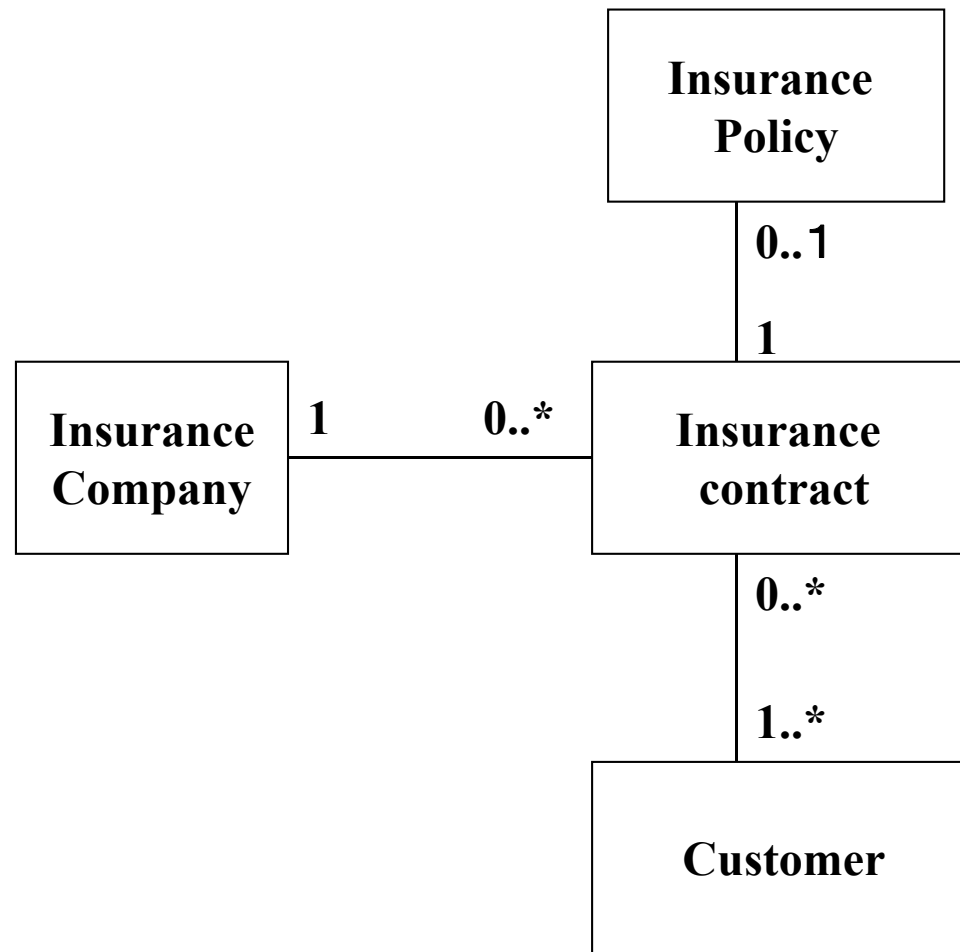
G.Booch, J.Rumbaugh, I. Jacobson , "The Unified Modeling Language User Guide", Addison Wesley, 1999.

JAIST Koichiro Ochimizu

Multiplicity



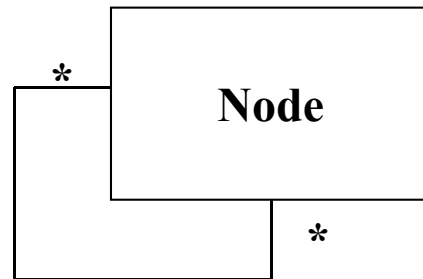
A class diagram describing an Insurance business



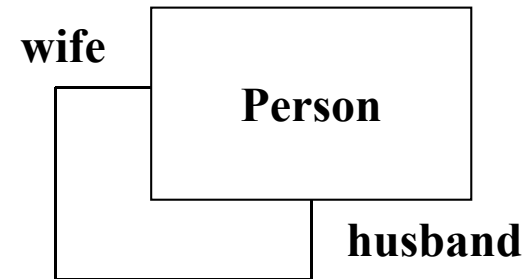
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

JALSI Koichiro Shimizu

Recursive Association



Connects



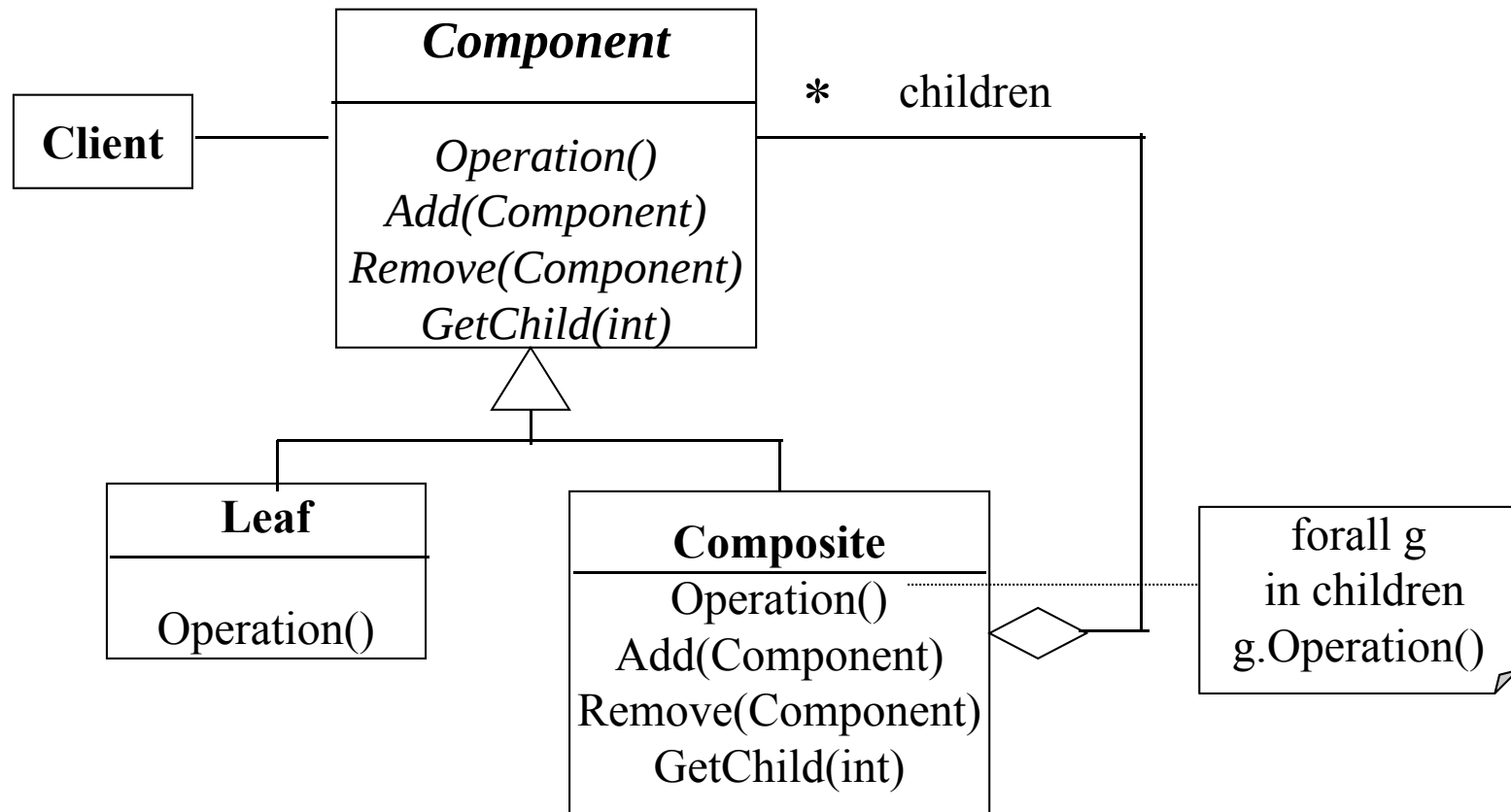
◀ **married to**

H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

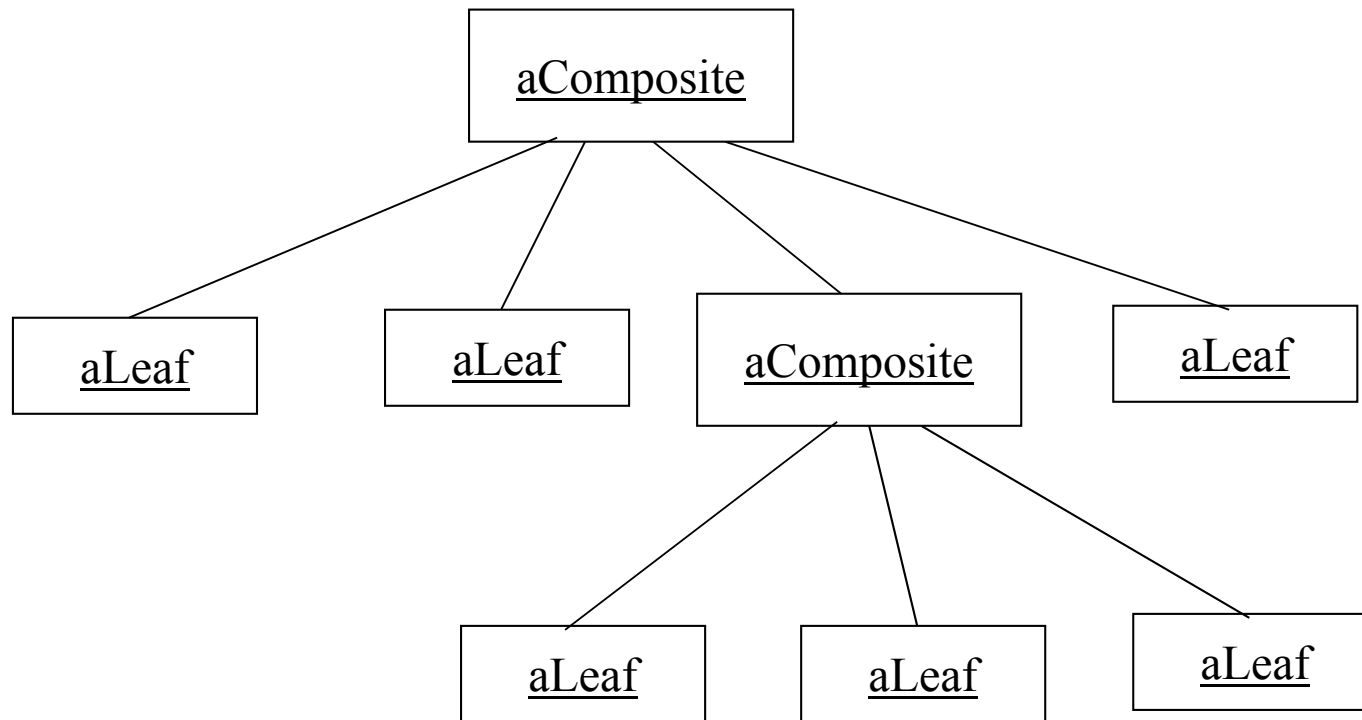
JAIST Koehnle Optimization

Composite

Clients use the Component class interface to interact with objects in the composite structure. If the recipient is a Leaf, then the request is handled directly. If the recipient is a Composite, then it usually forwards requests to its child components, possibly performing additional operations before and/or after forwarding.



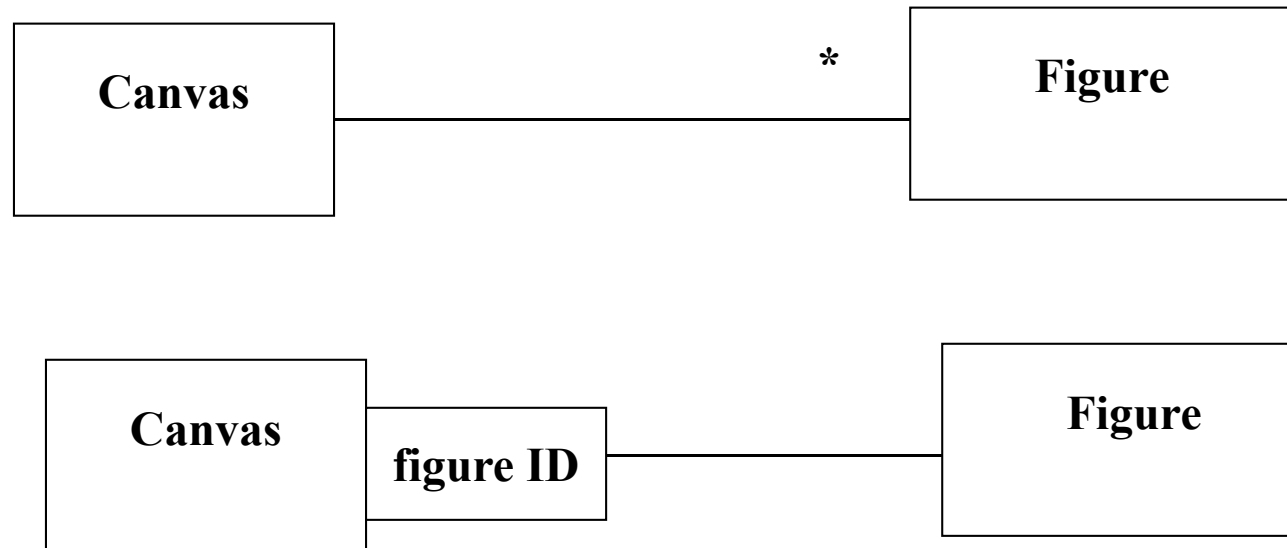
Object Diagram



E.Gamma, R.Helm, R.Johnson, J.Vlissides,"Design Patterns", ADDISON-WESLEY, 1995.

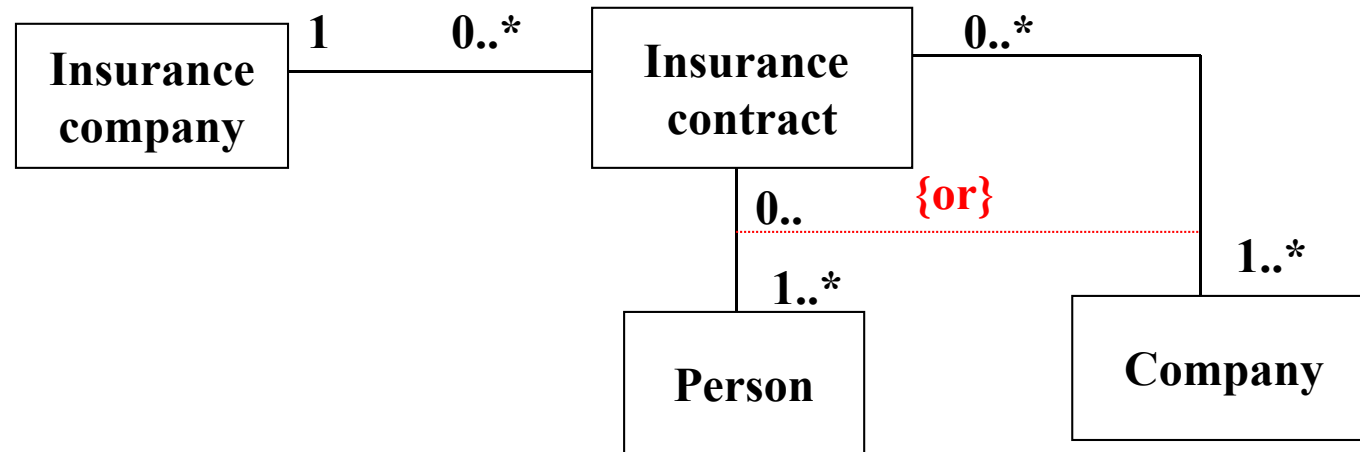
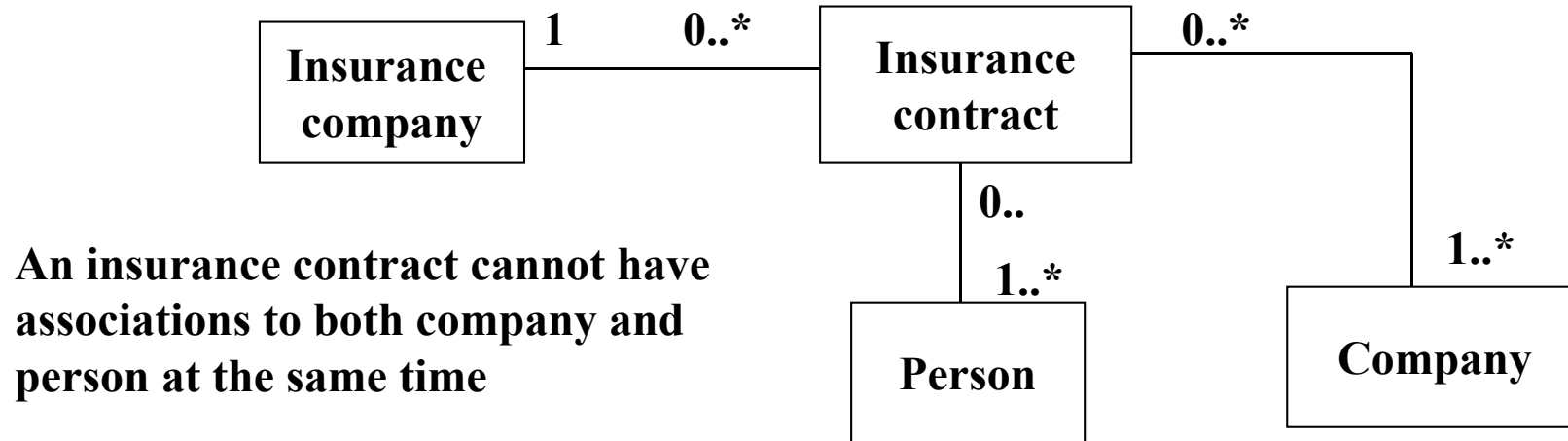
JAIST Koichiro Ochimizu

Qualifier

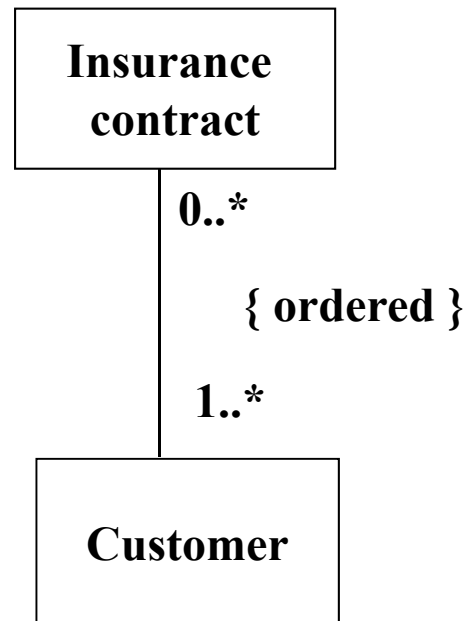


“one to many” to “one to one

OR-association



Ordered Association

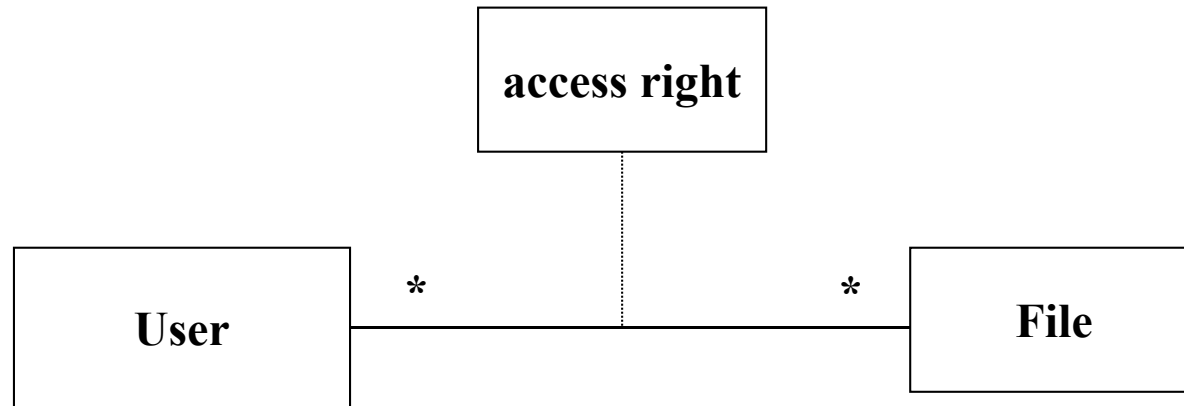


H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

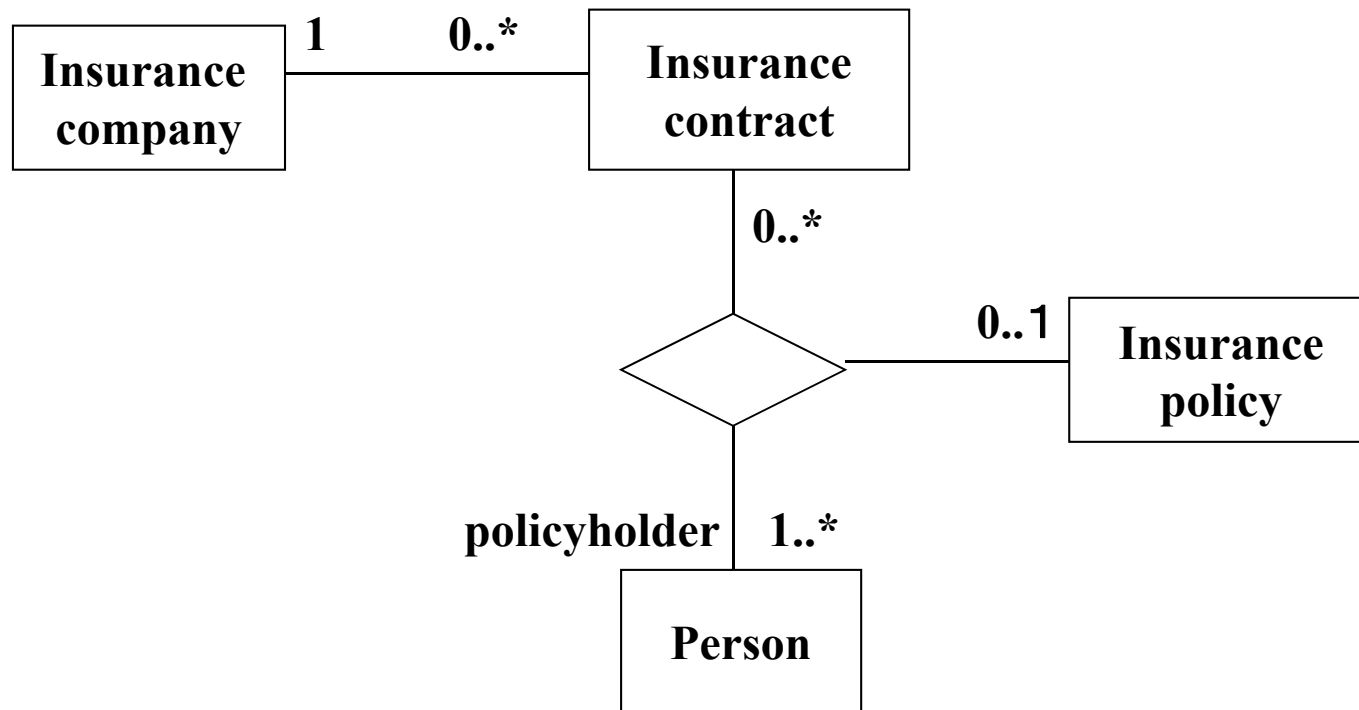
JAIST Koichiro Ochimizu

Association Class

Access right is an attribute of a link



Ternary Association



H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu

What is an Aggregation ?

- Aggregation is a special case of association
- “whole-part”, “is-part-of”
- Special kinds of aggregation
 - aggregation
 - Shared Aggregation
 - Composition Aggregation



Aggregation

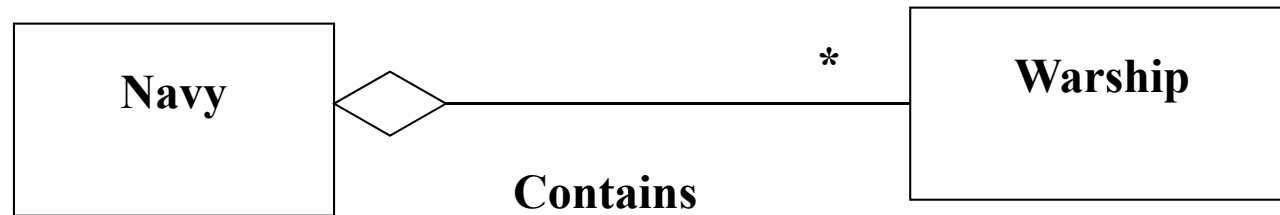


Composition: The part has the same life span with the whole

H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu

Aggregation



This example is not so good.

Shared Aggregation



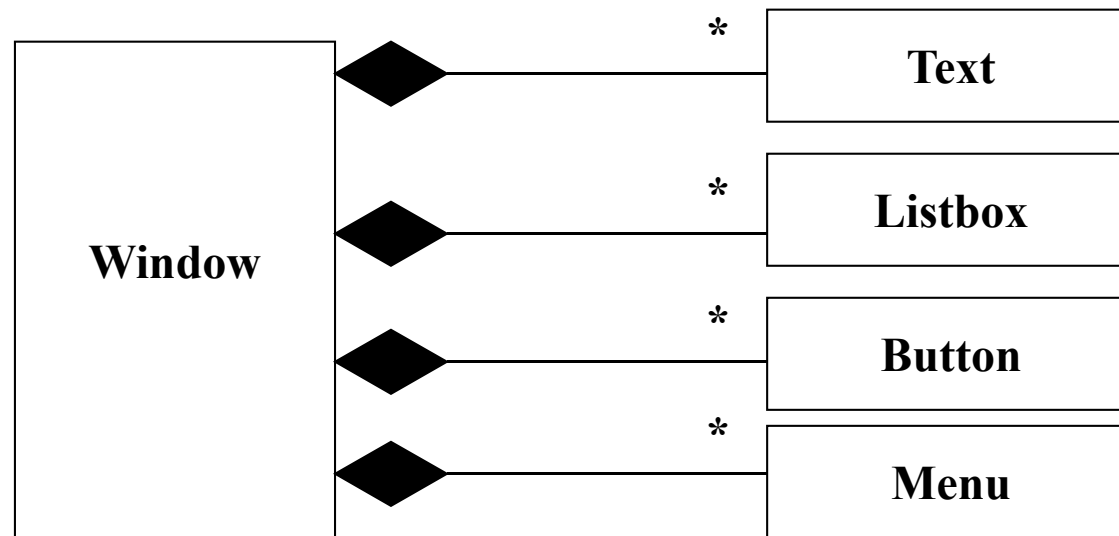
A team is composed of team members.

One person could be a member of many teams.

H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu

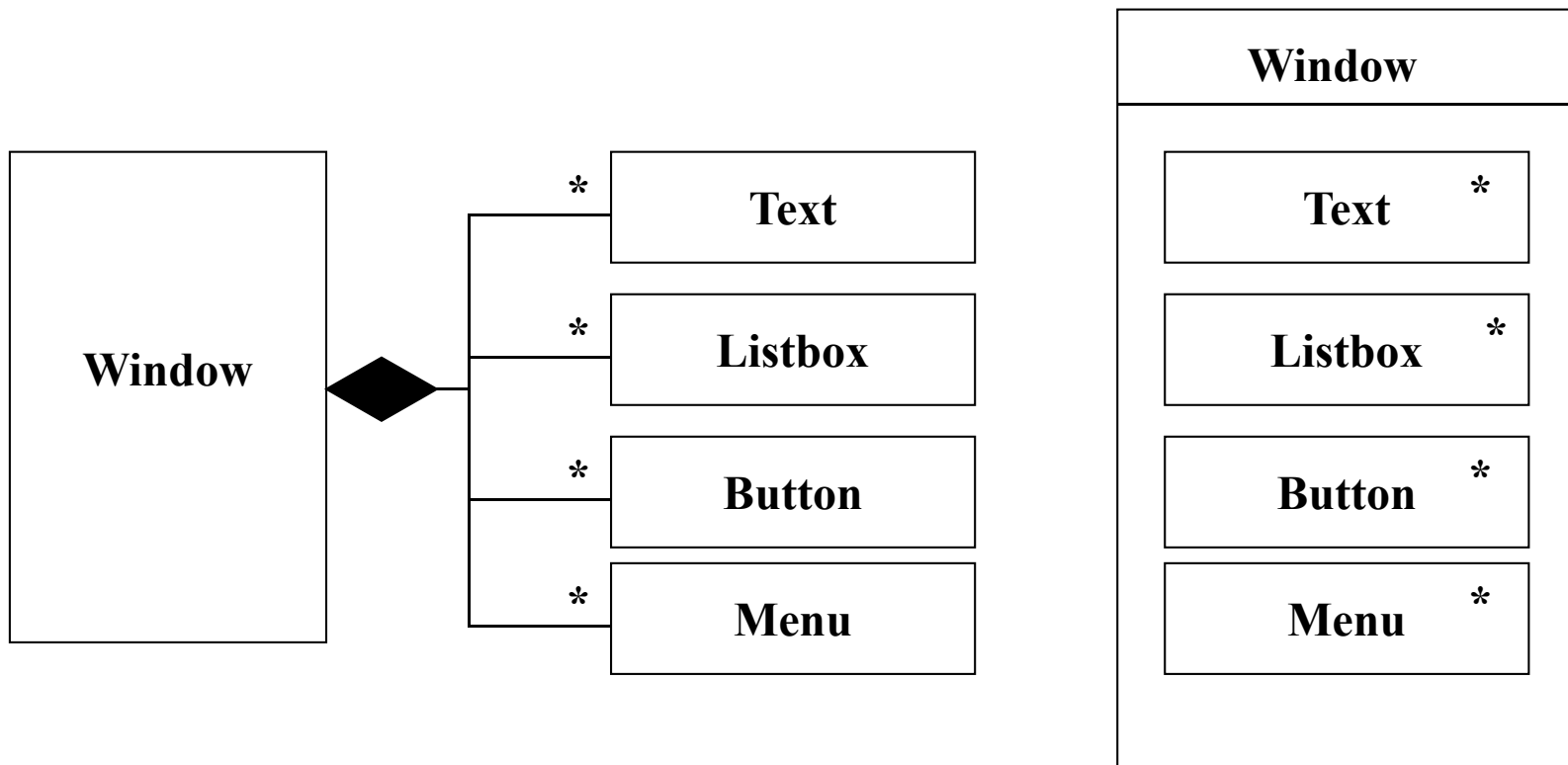
Composition Aggregation



H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu

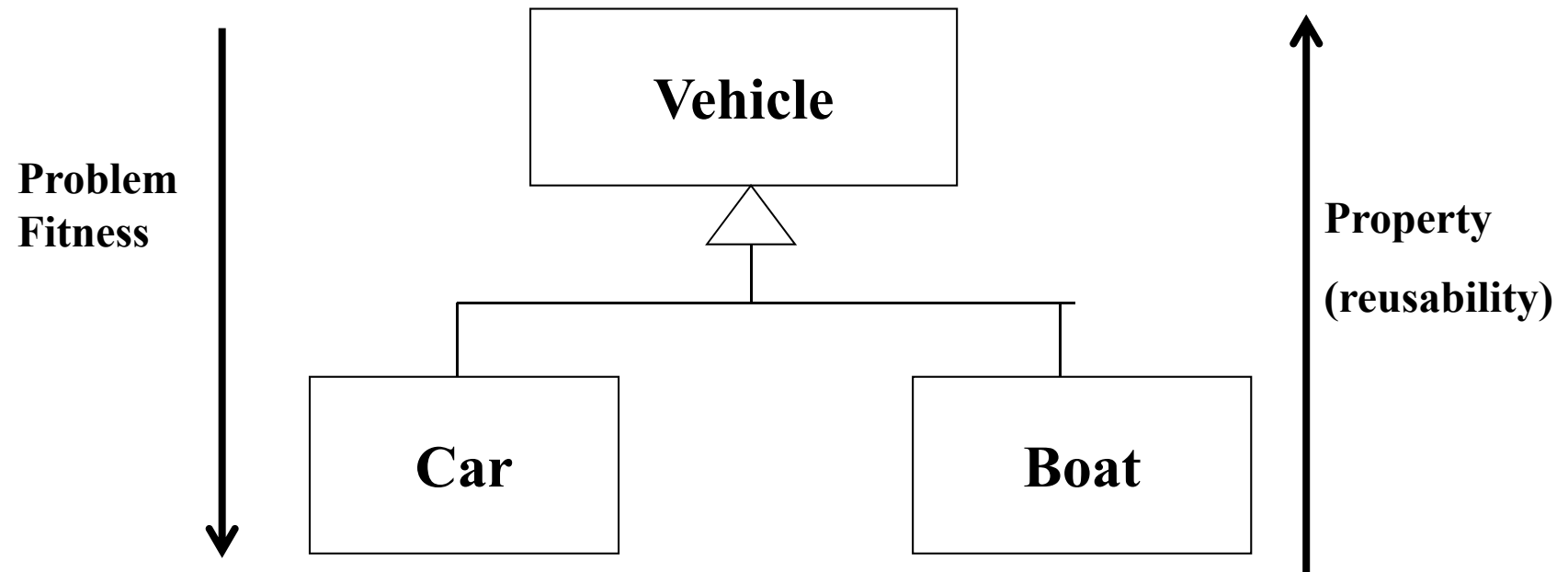
Another Expression of Aggregation



H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

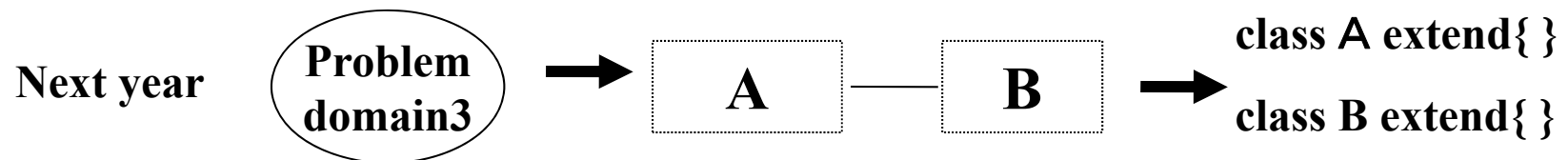
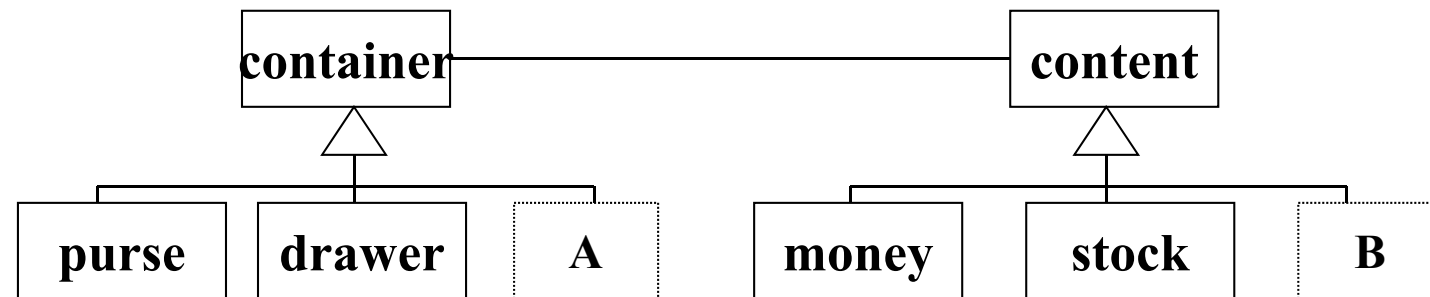
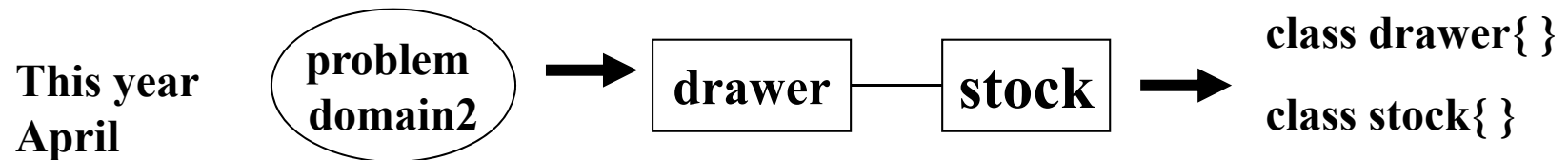
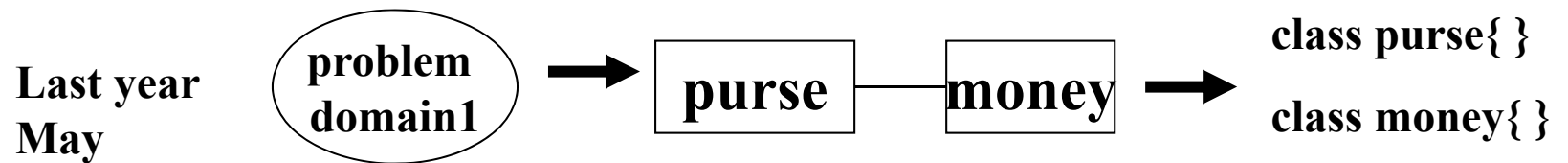
JAIST Koichiro Ochimizu

Generalization

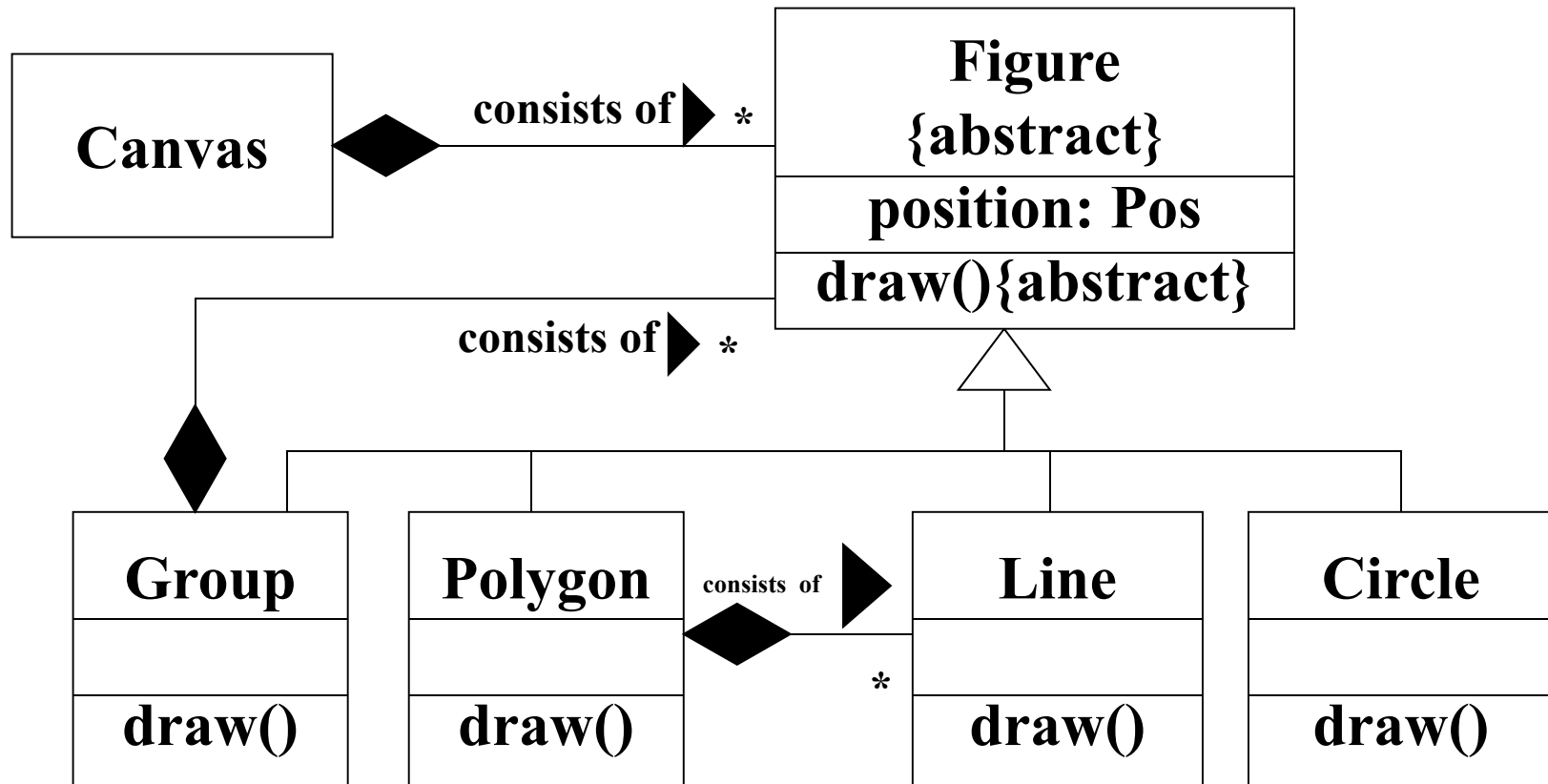


- Associations and Aggregation is used to describe the structure of the specific problem domain which we are interested in.
- Generalizations describe concepts which are common to several problem domains.

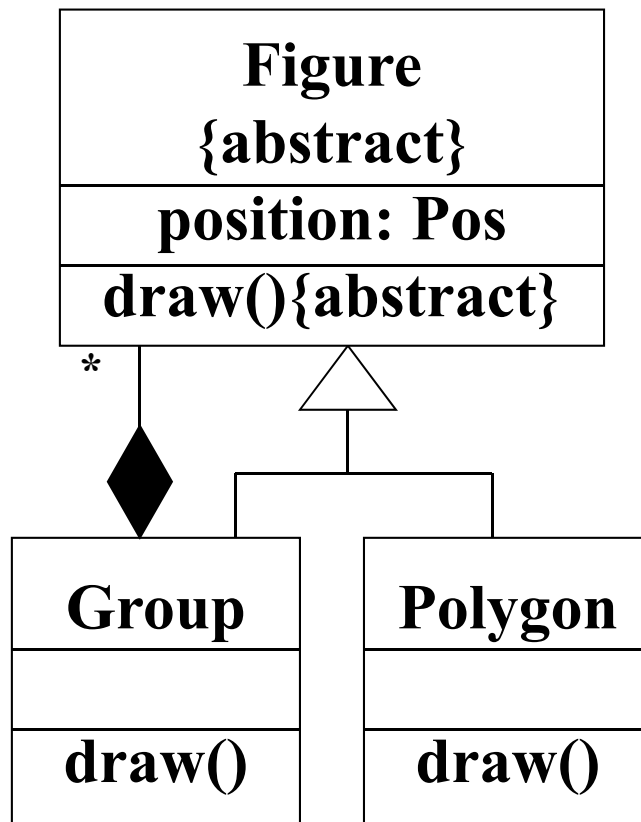
Generalization and Association



Combination of Inheritance and Aggregation



Java Implementation



```
abstract public class Figure
{
    abstract public void draw();
    protected Pos position;
};
```

```
public class Polygon extends Figure
{
    public void draw();
    /* draw polygon code */
}
};
```

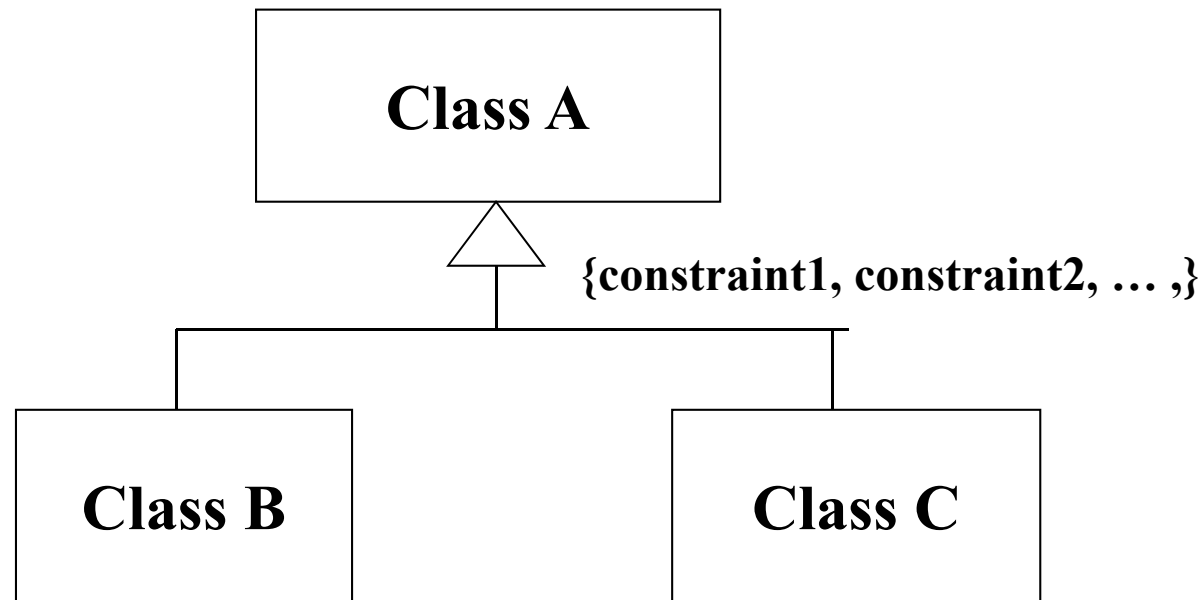
Constrained Generalization

Overlapping

Disjoint

Complete

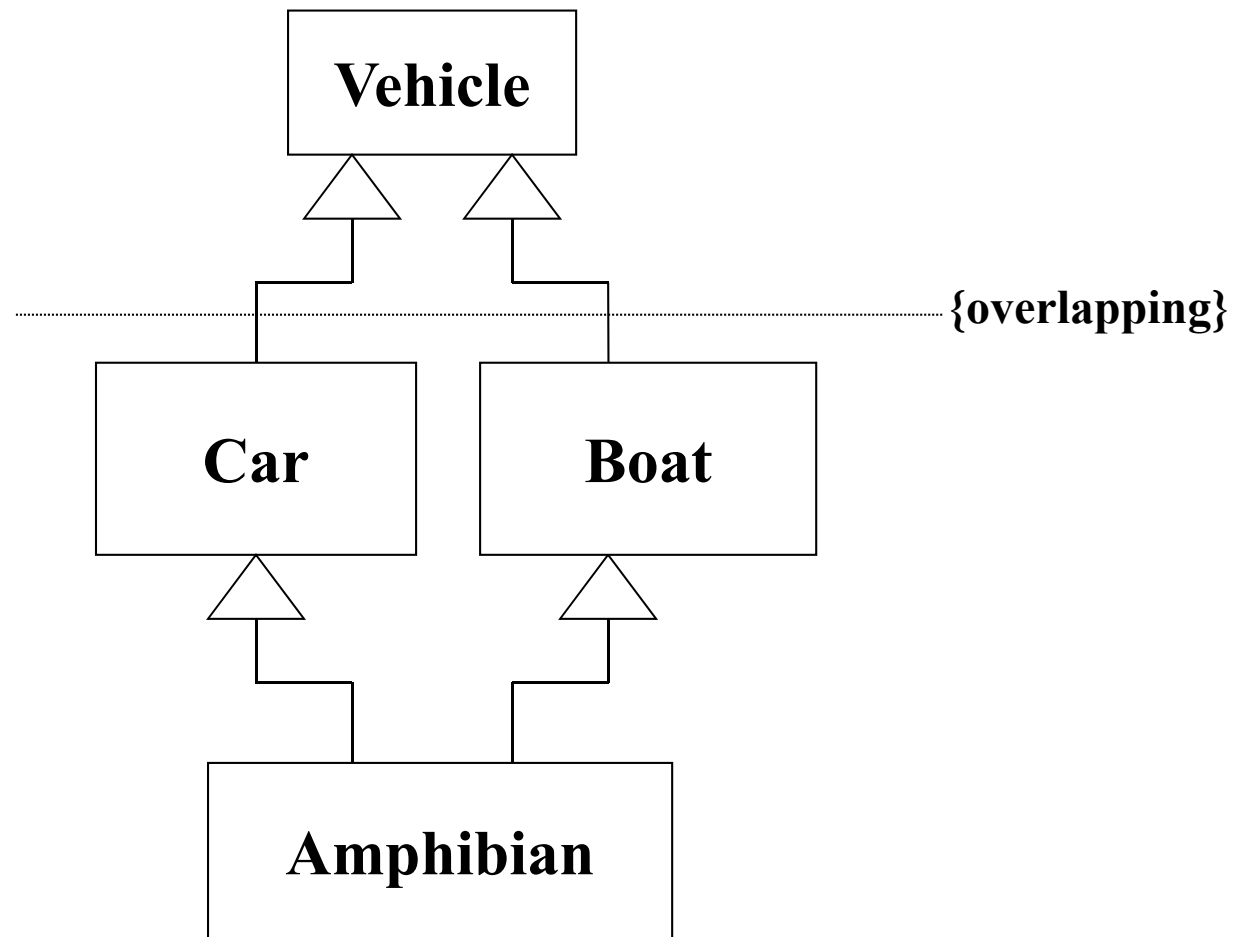
Incomplete



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu

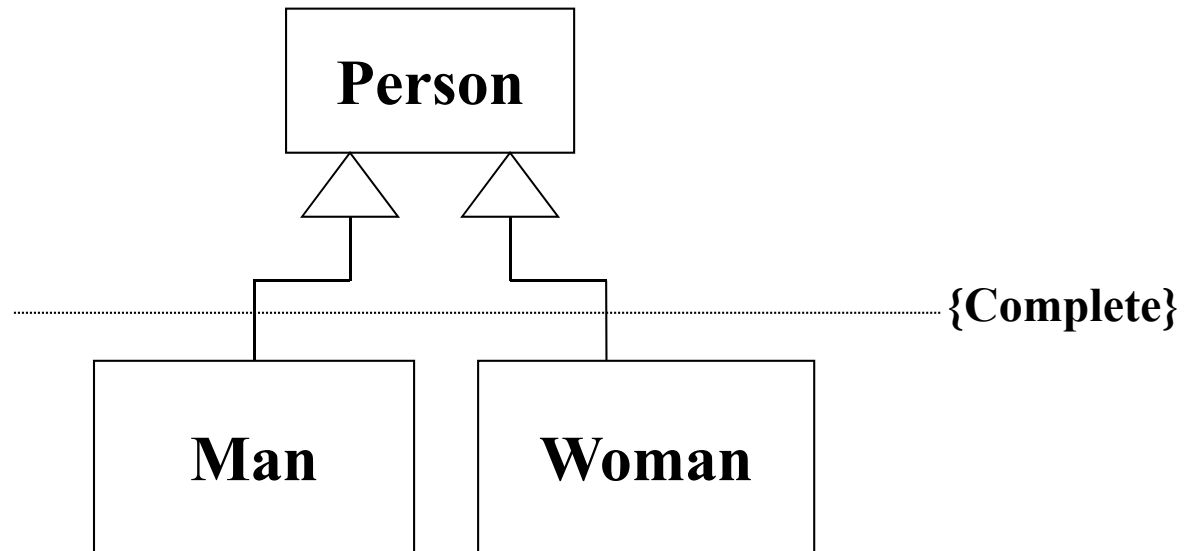
Overlapping and Disjoint



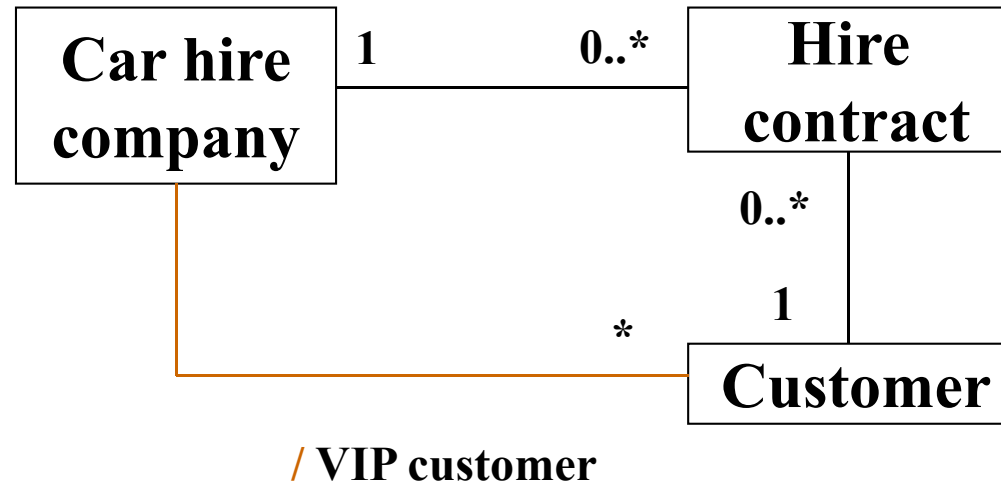
H.E. Eriksson and M. Penker, "UML Toolkit" John Wiley & Sons, Inc.

JAIST Koichiro Gotsumo

Complete and Incomplete



Derivation



- can be computed from other associations and attributes.
- The VIP customers are a derived association when company makes contracts with many customers.

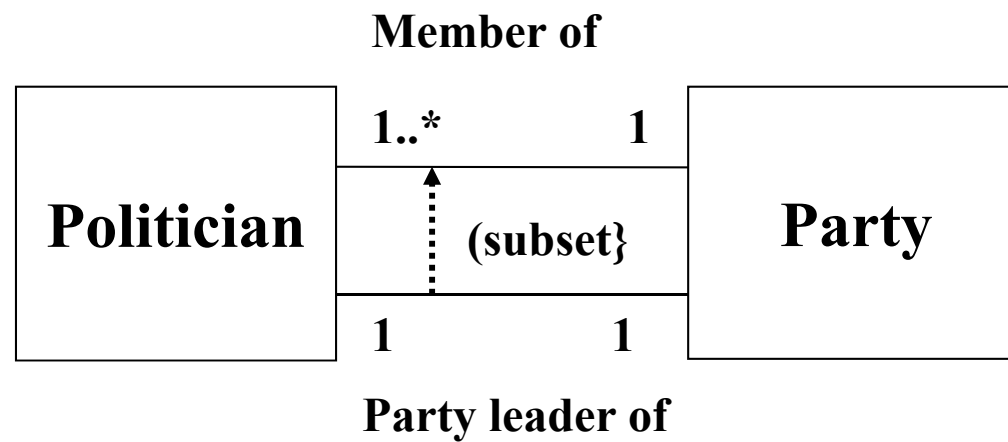
Derived Attribute

Article
cost price sales price /profit

{ profit = sales price – cost price }

H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

Subset Constraint



H.E. Eriksson and M. Penker, “UML Toolkit” John Wiley & Sons, Inc.

JAIST Koichiro Ochimizu