Module 9: Relationships in Class Diagrams

Dr. Tran Minh Triet

Acknowledgement

- Slides
  - Course CS202: Programming Systems

    Instructor: MSc. Karla Fant,

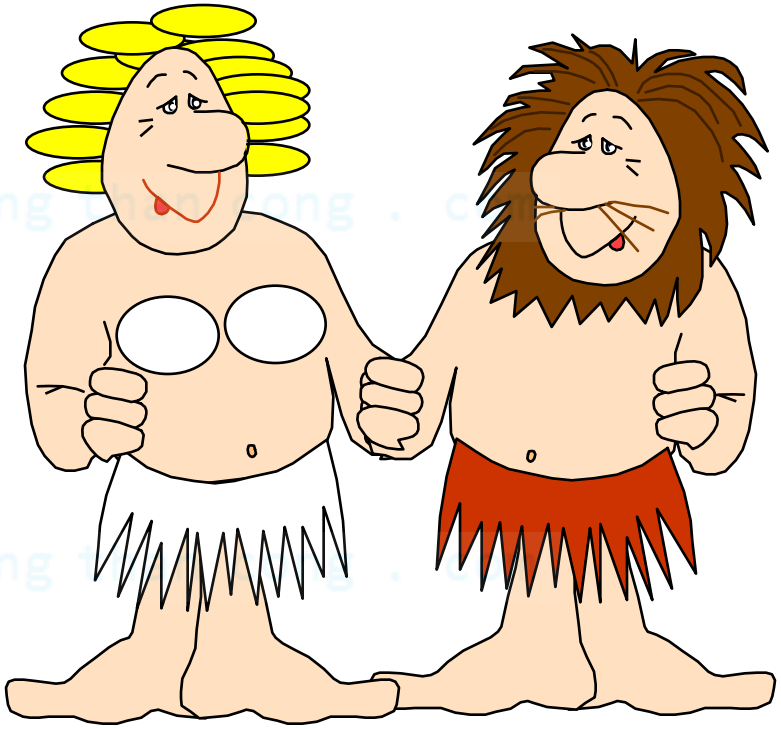
    Portland State University
  - Course CS202: Programming Systems

    Instructor: Dr. Dinh Ba Tien,

    University of Science, VNU-HCMC
  - Course DEV275: Essentials of Visual Modeling with UML 2.0

    IBM Software Group

Relationships



Outline

- Relationships in class diagrams
- Generalization
- Association
- Aggregation
- Composition
- Dependency

The Need for Relationships

- All systems encompass many classes and objects
- Objects contribute to the behavior of a system by collaborating with one another
  - Collaboration is accomplished through relationships
- Important types of relationships:
  - Generalization
  - Association/Aggregation/Composition
  - Dependency

Associations

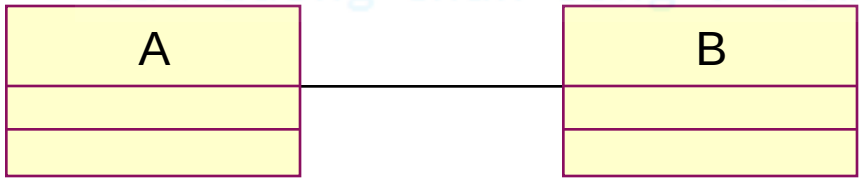
- An association is a bi-directional semantic connection between classes
  - This implies that there is a link between objects in the associated classes
- Associations are represented on class diagrams by a (solid) line connecting the associated classes
- Data may flow in either direction or both directions across a link

Association

- Association relationship between class A and class B
  - there is at least one attribute of class B in class A

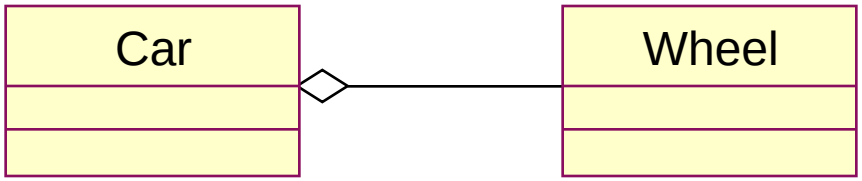
or

  - there is at least one attribute of class A in class B



Aggregation

- Aggregation is a specialized form of association in which a whole is related to its part(s)
  - Aggregation is known as a "part-of" or containment relationship
- An aggregation is represented as an association with a diamond next to the class denoting the aggregate (whole)



Aggregation Tests

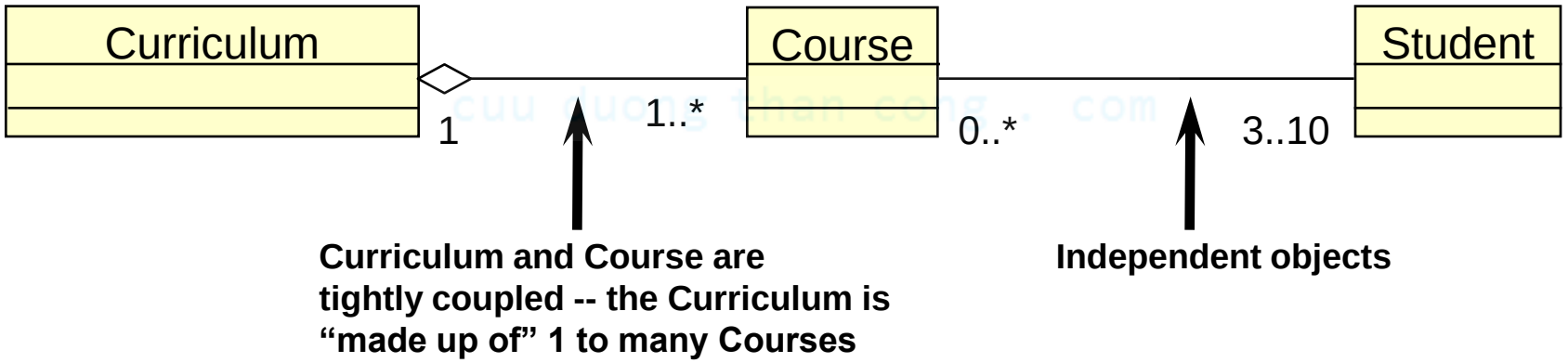
- Is the phrase "part of" used to describe the relationship?
  - A Door is "part of" a Car
- Are some operations on the whole automatically applied to its parts?
  - Move the Car, Move the Door

Aggregation Tests

- Are some attribute values propagated from the whole to all or some of its parts?
  - The Car is blue, the Door is Blue
- Is there an intrinsic asymmetry to the relationship where one class is subordinate to the other?
  - A Door IS part of a Car, a Car IS NOT part of a Door

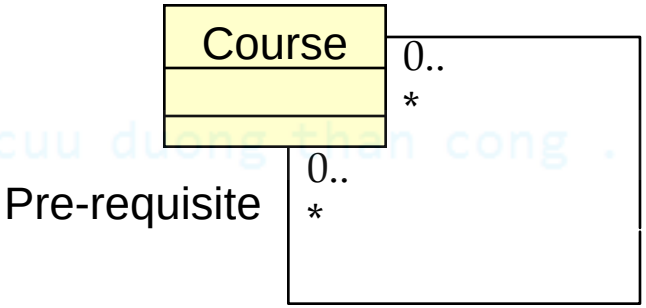
Association or Aggregation?

- If two objects are tightly bound by a whole-part relationship
  - The relationship is an aggregation
- If two objects are usually considered as independent, even though they are often linked
  - The relationship is an association



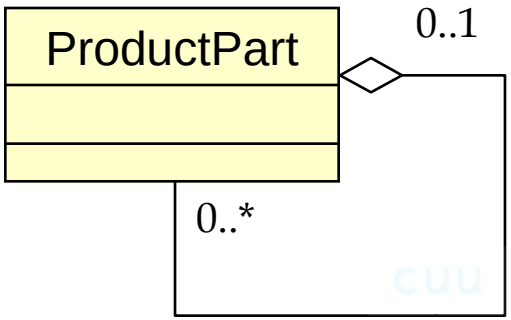
Reflexive Associations

- In a reflexive association, objects in the same class are related
  - Indicates that multiple objects in the same class collaborate together in some way



Reflexive Aggregates

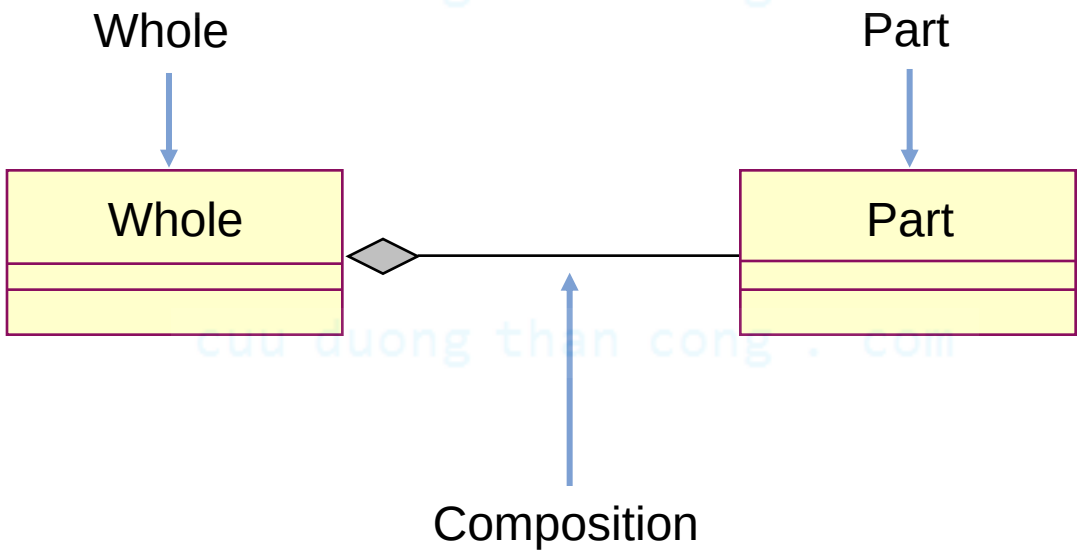
- Aggregates can also be reflexive
  - Classic bill of materials type problem
- This indicates a recursive relationship



One ProductPart object contains zero or more ProductPart objects

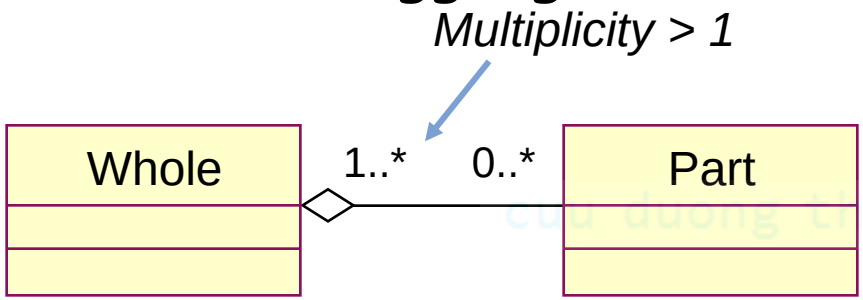
Composition

- A form of aggregation with strong ownership and coincident lifetimes
  - The parts cannot survive the whole/aggregate

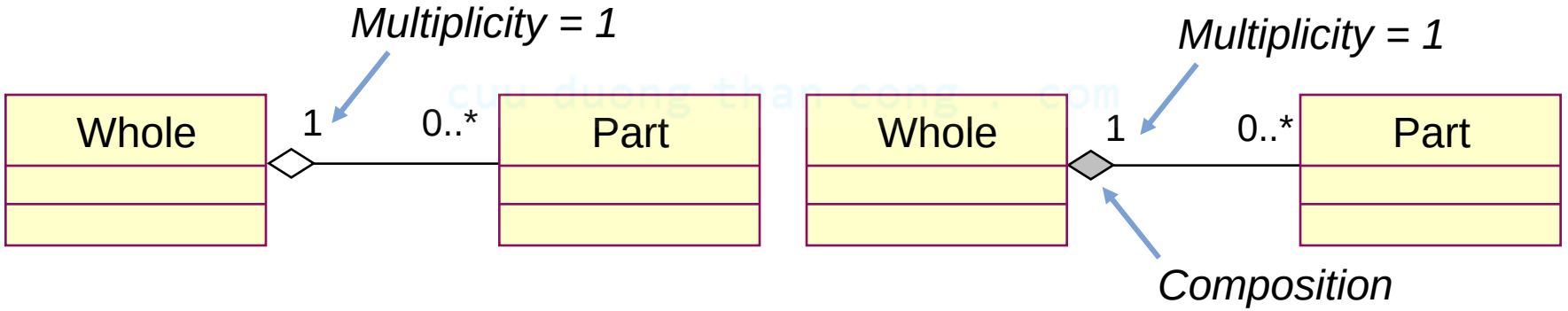


Aggregation: Shared vs. Non-shared

- Shared Aggregation



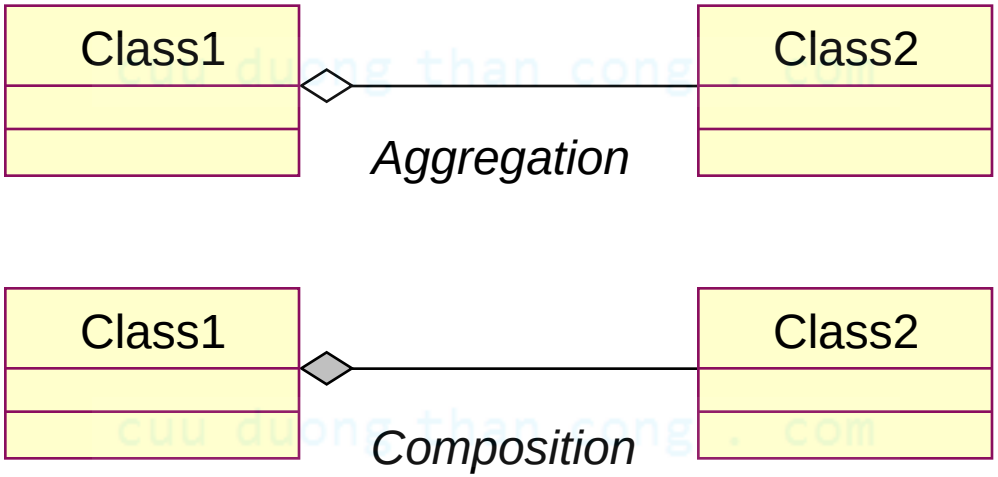
- Non-shared Aggregation



By definition, composition is non-shared aggregation.

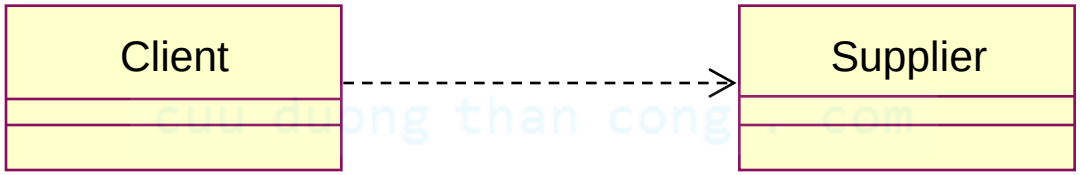
Aggregation or Composition?

- Consideration
  - Lifetimes of Class1 and Class2



Define Dependency

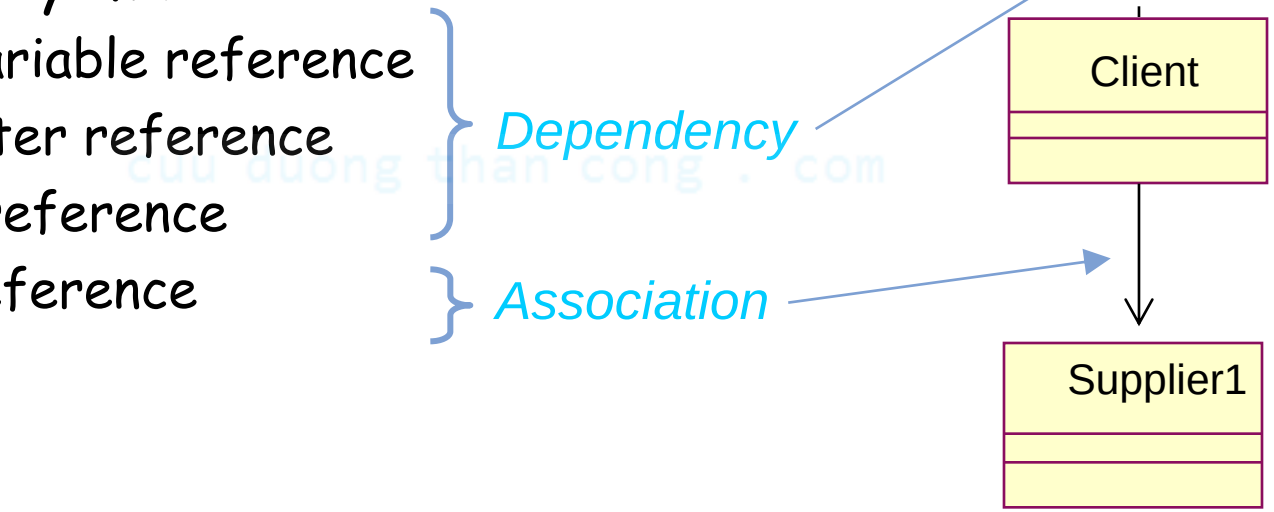
- What Is a Dependency?
  - A relationship between two objects



- Purpose
  - Determine where structural relationships are NOT required
- Things to look for :
  - What causes the supplier to be visible to the client

Dependencies vs. Associations

- Associations are structural relationships
- Dependencies are non-structural relationships
- In order for objects to "know each other" they must be visible
  - Local variable reference
  - Parameter reference
  - Global reference
  - Field reference

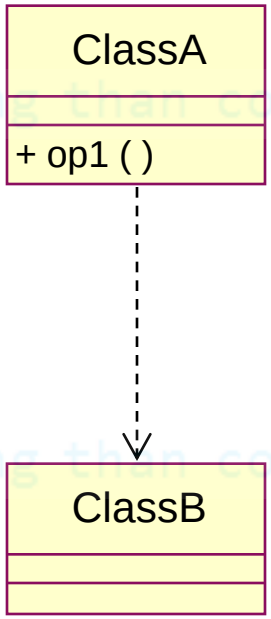


Associations vs. Dependencies in Collaborations

- An instance of an association is a link
  - All links become associations unless they have global, local, or parameter visibility
  - Relationships are context-dependent
- Dependencies are transient links with:
  - A limited duration
  - A context-independent relationship
  - A summary relationship

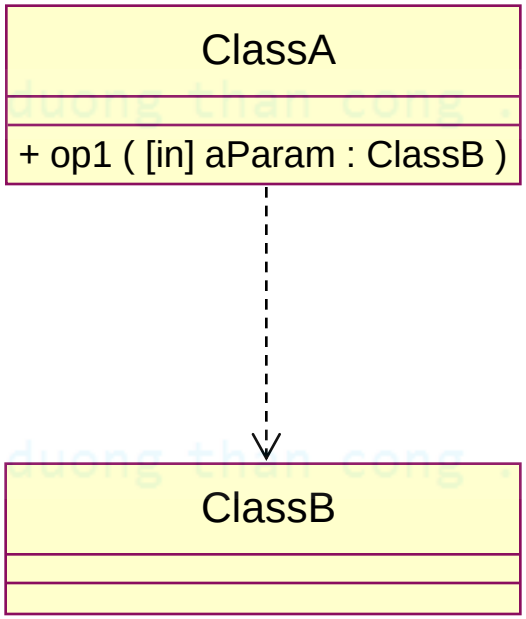
Local Variable Visibility

- The op1() operation contains a local variable of type ClassB



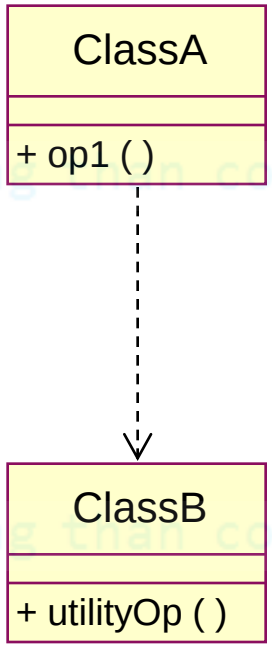
Parameter Visibility

- The ClassB instance is passed to the ClassA instance



Global Visibility

- The ClassUtility instance is visible because it is global



Identifying Dependencies: Considerations

- Permanent relationships — Association (field visibility)
- Transient relationships — Dependency
  - Multiple objects share the same instance
    - Pass instance as a parameter (parameter visibility)
    - Make instance a managed global (global visibility)
  - Multiple objects don't share the same instance (local visibility)
- How long does it take to create/destroy?
  - Expensive? Use field, parameter, or global visibility
  - Strive for the lightest relationships possible