



OBJECT LANGUAGE AND THEORY

12. CLASS DIAGRAMS


Nguyen Thi Thu Trang
trangntt@soict.hust.edu.vn



@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 2

Objectives


- Describe the static view of the system and show how to capture it in a model.
- Demonstrate how to read and interpret a class diagram.
- Model an association and aggregation and show how to model it in a class diagram.
- Model generalization on a class diagram.



@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 3

Content

- ➔ 1. Class diagrams
- 2. Association
- 3. Aggregation and Composition
- 4. Generalization




@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 4

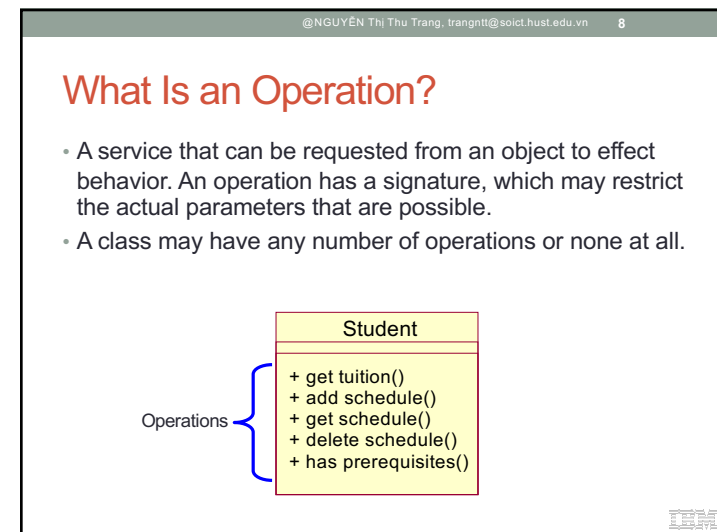
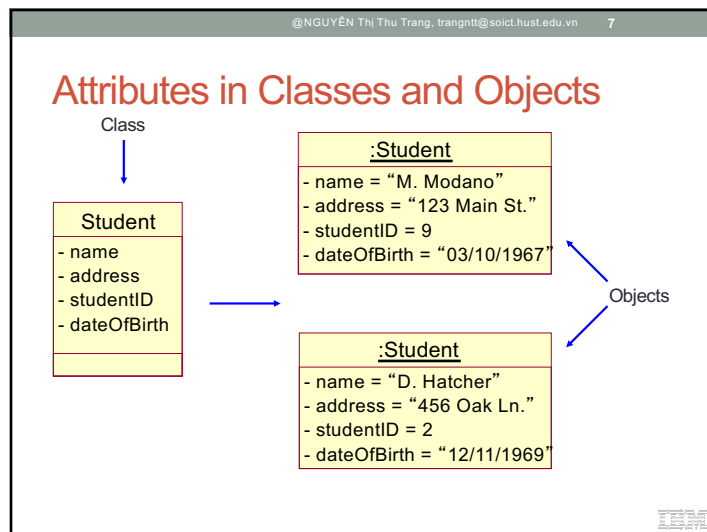
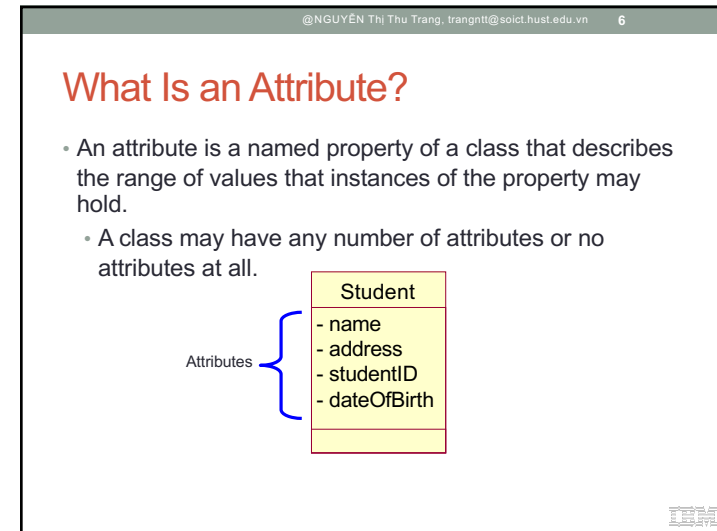
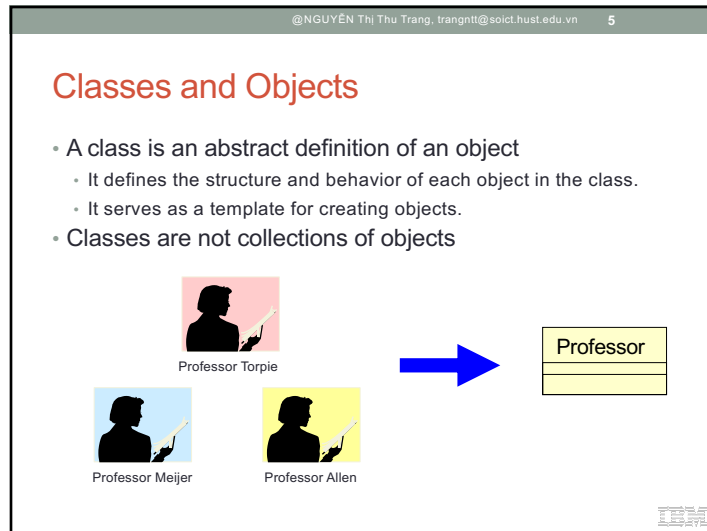
4.1. Classes in the UML

- A class is represented using a rectangle with three compartments:

- The class name
- The structure (attributes)
- The behavior (operations)

Professor
- name
- employeeID : UniqueId
- hireDate
- status
- discipline
- maxLoad
+ submitFinalGrade()
+ acceptCourseOffering()
+ setMaxLoad()
+ takeSabbatical()
+ teachClass()





@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 9

Operation Visibility

- Visibility is used to enforce encapsulation
- May be public, protected, or private

Public operations

Protected operations

Private operations

IBM

@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 10

How Is Visibility Noted?

- The following symbols are used to specify export control:
 - + Public access
 - # Protected access
 - - Private access

ClassName
- privateAttribute
+ publicAttribute
protectedAttribute
- privateOperation ()
+ publicOperation ()
protectedOperation ()

IBM

@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 11

4.2. Package in UML

- A general purpose mechanism for organizing elements into groups.
- A model element that can contain other model elements.
- A package can be used:
 - To organize the model under development
 - As a unit of configuration management

University Artifacts

IBM

@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 12

A Package Can Contain Classes

- The package, University Artifacts, contains one package and five classes.

University Artifacts

Student Artifacts

Professor

Course

Schedule

Student

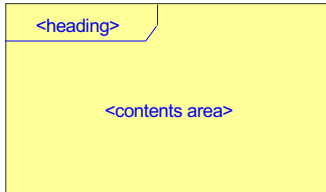
CourseOffering

IBM

@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 13

4.3. Diagram Depiction

- Each diagram has a frame, a heading compartment in the upper left corner, and a contents area.
- If the frame provides no additional value, it may be omitted and the border of the diagram area provided by the tool will be the implied frame.

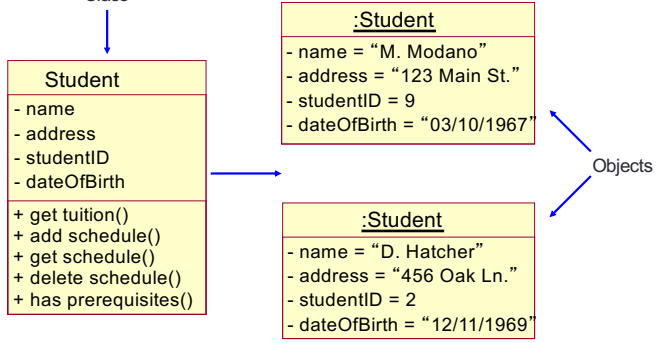


The diagram shows a yellow rectangular frame. In the top-left corner, there is a small compartment labeled "<heading>". The rest of the frame is labeled "<contents area>".

@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 14

Review: Classes and Objects in UML

Class

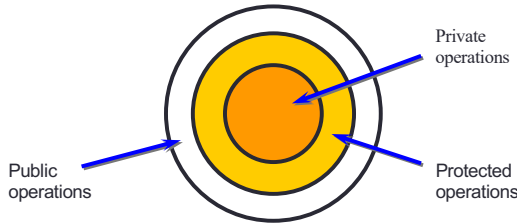


The diagram illustrates the relationship between a class and its objects. On the left, a class box labeled "Student" contains attributes: - name, - address, - studentID, - dateOfBirth, and methods: + get tuition(), + add schedule(), + get schedule(), + delete schedule(), + has prerequisites(). An arrow points from the class box to two object boxes on the right. Both object boxes are labeled ":Student" and contain instance-specific attribute values. The top object has: - name = "M. Modano", - address = "123 Main St.", - studentID = 9, - dateOfBirth = "03/10/1967". The bottom object has: - name = "D. Hatcher", - address = "456 Oak Ln.", - studentID = 2, - dateOfBirth = "12/11/1969". A label "Objects" with arrows points to both object boxes.

@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 15

Review: Operation Visibility

- Visibility is used to enforce encapsulation
- May be public (+), protected (#), or private (-)

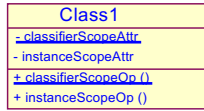


The diagram uses three concentric circles to represent visibility levels. The outermost circle is labeled "Public operations". The middle circle is labeled "Protected operations". The innermost circle is labeled "Private operations".

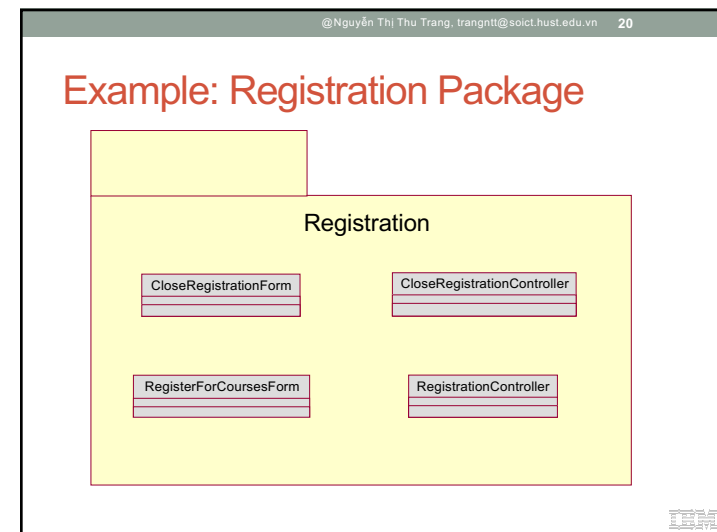
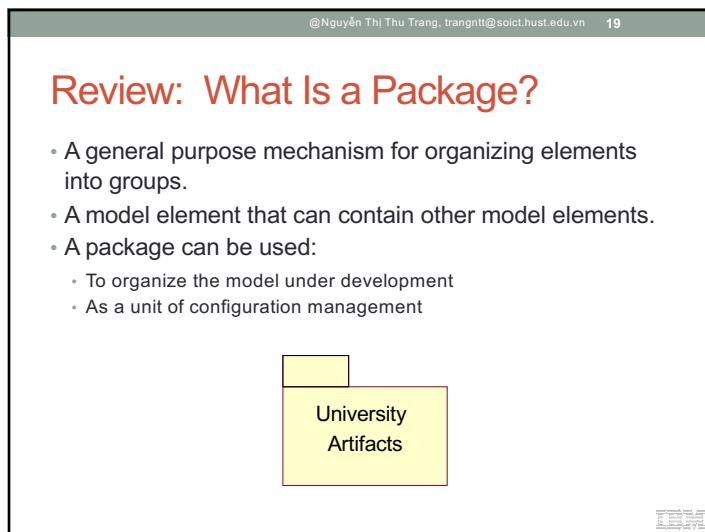
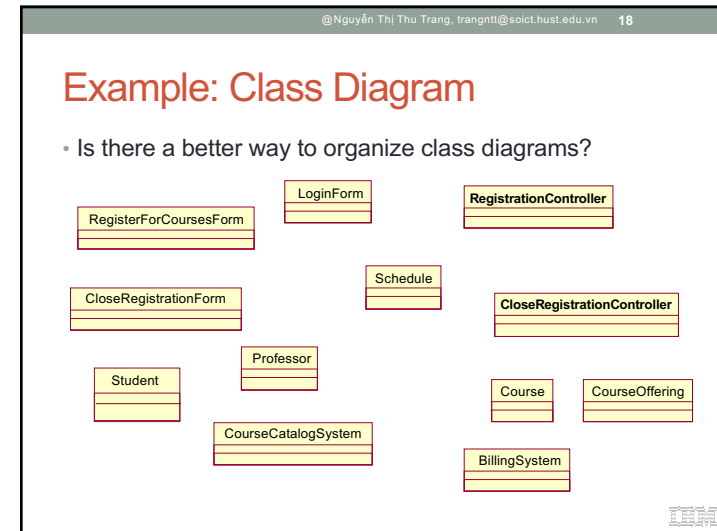
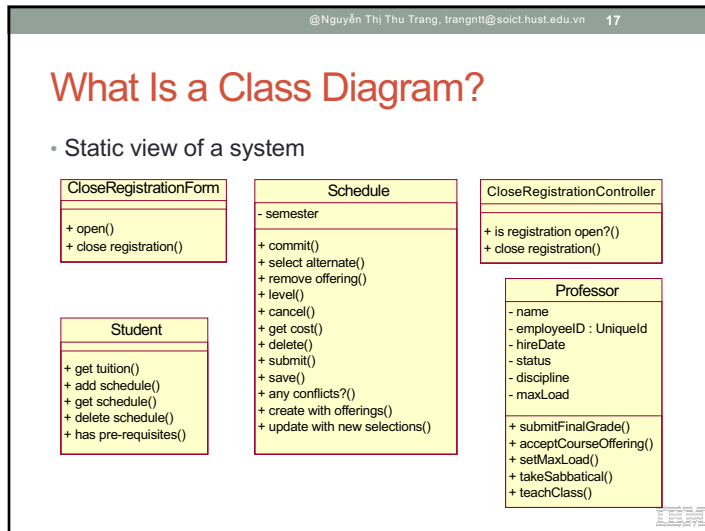
@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 16

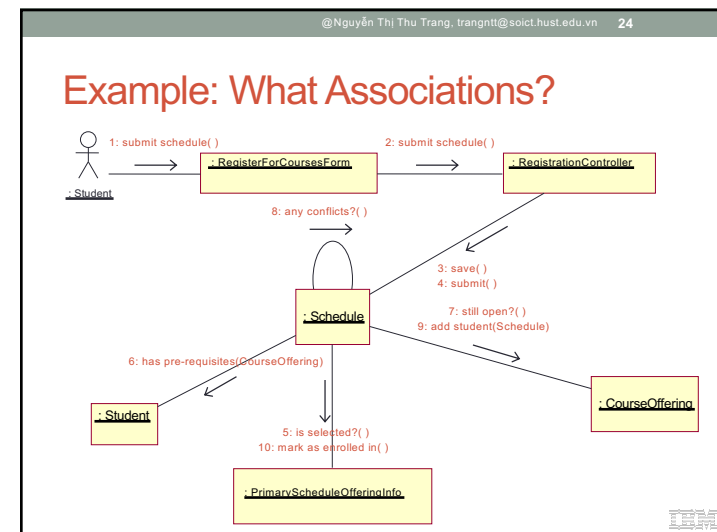
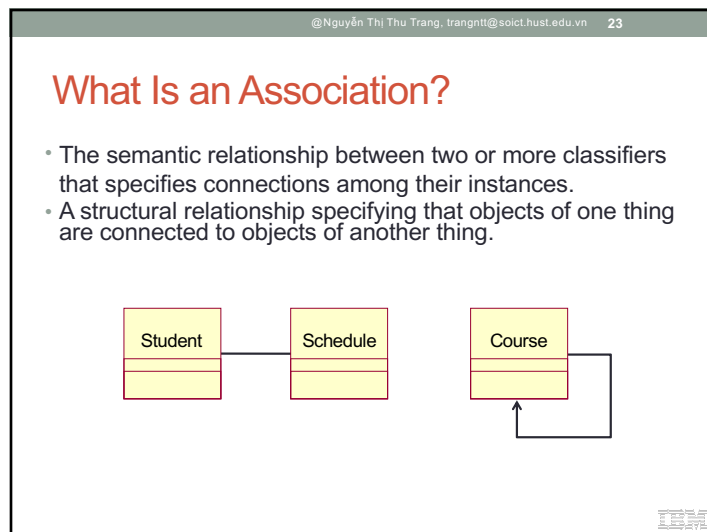
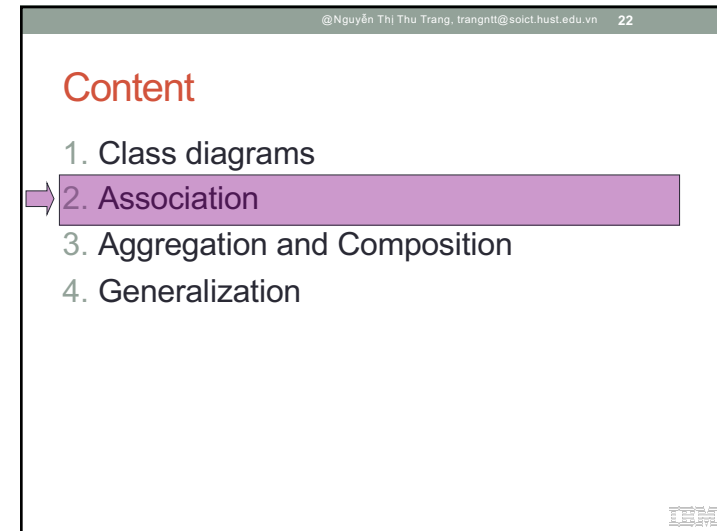
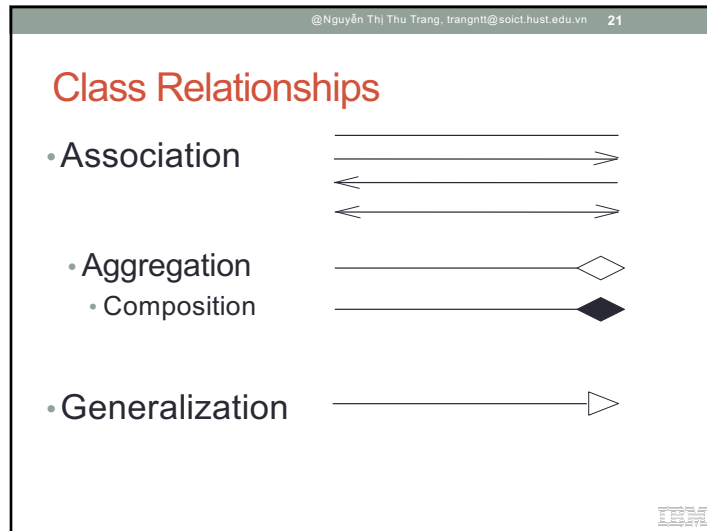
Scope

- Determines number of instances of the attribute/operation
 - Instance: one instance for each class instance
 - Classifier: one instance for all class instances
- Classifier scope is denoted by underlining the attribute/operation name



The diagram shows a class box labeled "Class1". It contains two attributes: - classifierScopeAttr (underlined) and - instanceScopeAttr. It also contains two methods: + classifierScopeOp () (underlined) and + instanceScopeOp ().





@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 25

Role

```

classDiagram
    class Car
    class Person
    Car "*" -- "*" Person : drives
    note for Car "company car"
    note for Person "driver"
  
```

- Role
 - Useful technique for specifying the context of a class and its objects
 - Optional
- Role name
 - String placed near the end of the association next to the class to which it applies
 - Indicates the role played by the class in terms of the association.
 - Part of the association and not part of the classes

IBM

@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 26

What Is Multiplicity?

- Multiplicity is the number of instances one class relates to ONE instance of another class.
- For each association, there are two multiplicity decisions to make, one for each end of the association.
 - For each instance of Professor, many Course Offerings may be taught.
 - For each instance of Course Offering, there may be either one or zero Professor as the instructor.

```

classDiagram
    class Professor
    class CourseOffering
    Professor "0..1" -- "0..*" CourseOffering : instructor
  
```

IBM

@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 27

Multiplicity Indicators

Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

IBM

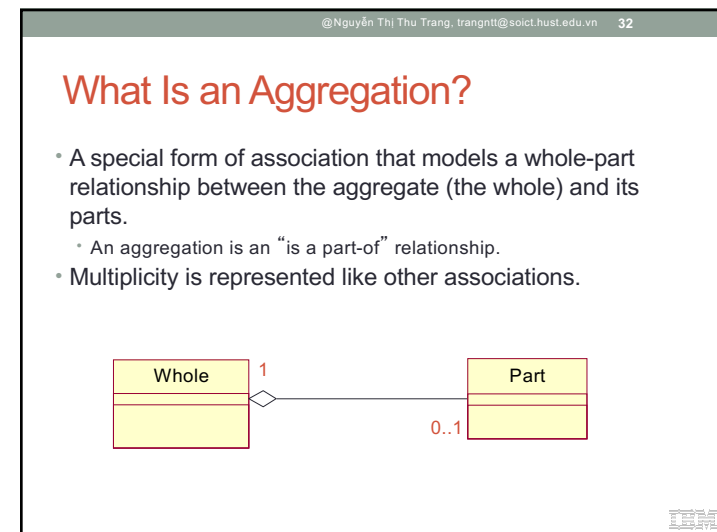
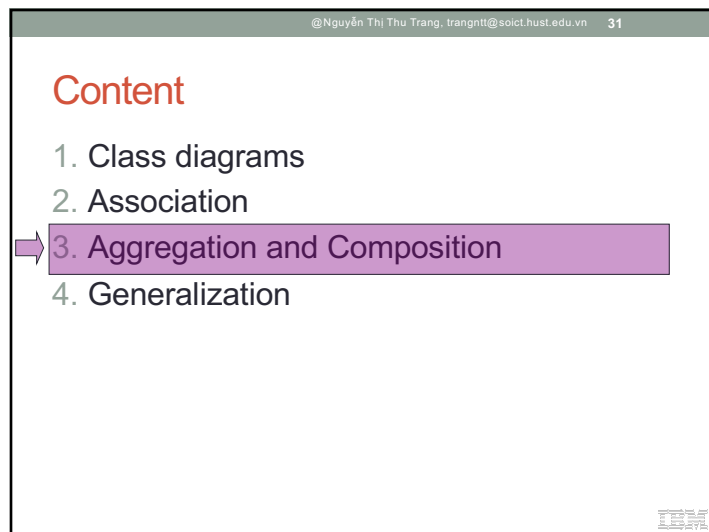
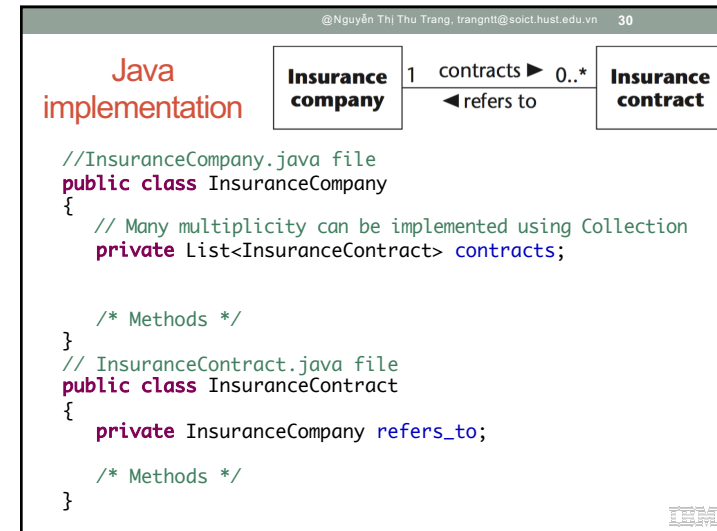
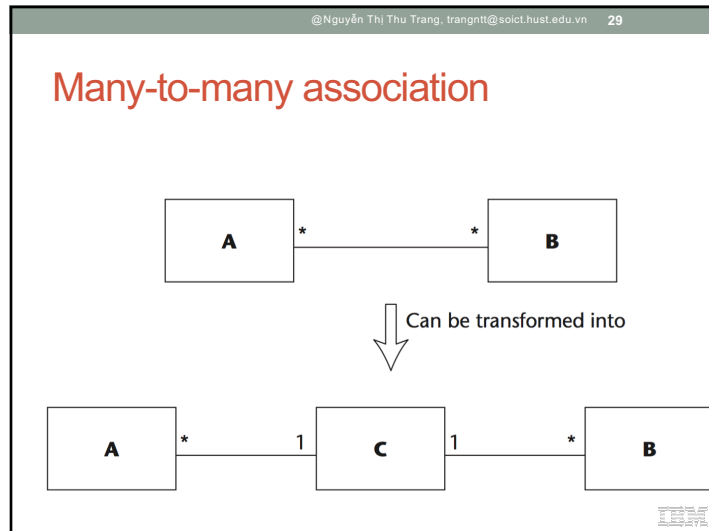
@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 28

Example: Multiplicity

```

classDiagram
    class RegisterForCoursesForm
    class RegistrationController
    class Student
    class Schedule
    class CourseOffering
    RegisterForCoursesForm "1" -- "1" RegistrationController
    RegistrationController "0..1" -- "0..4" CourseOffering
    Student "1" -- "0..*" Schedule
    Schedule "0..*" -- "0..*" CourseOffering
  
```


IBM



@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 33

What is Composition?

- A special form of aggregation with strong ownership and coincident lifetimes of the part with the aggregate
 - Also called composition aggregate
- The whole “owns” the part and is responsible for the creation and destruction of the part.
 - The part is removed when the whole is removed.
 - The part may be removed (by the whole) before the whole is removed.


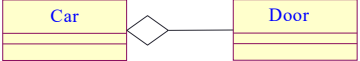



UML

@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 34

Examples: Association Types

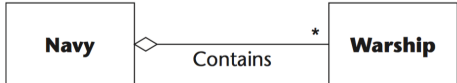
- Association
 - use-a
 - Objects of one class are associated with objects of another class
- Aggregation
 - has-a/is-a-part
 - Strong association, an instance of one class is made up of instances of another class
- Composition
 - Strong aggregation, the composed object can't be shared by other objects and dies with its composer
 - Share life-time

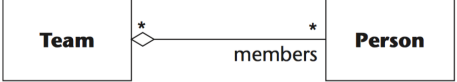
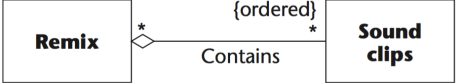
UML

@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 35

Aggregation Example



- A *shared aggregation* is one in which the parts may be parts in any wholes

UML

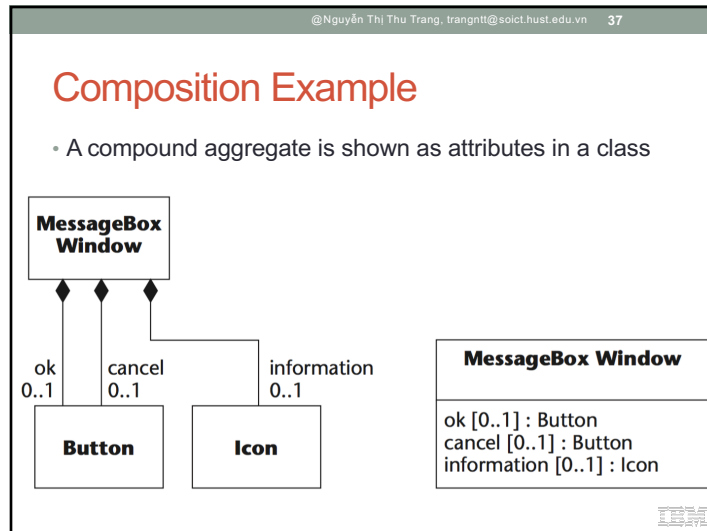
@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 36

Aggregation – Java implementation

```
class Car {
    private List<Door> doors;
    Car(String name, List<Door> doors) {
        this.doors = doors;
    }

    public List<Door> getDoors() {
        return doors;
    }
}
```

UML



Composition – Java implementation

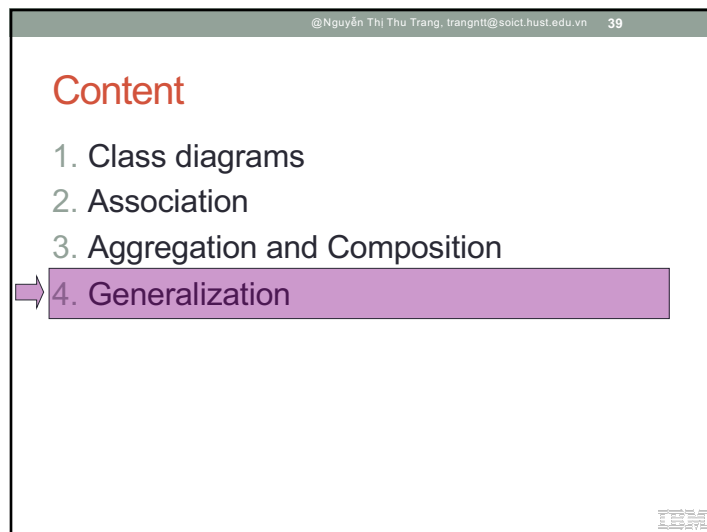
```

final class Car {
    // For a car to move, it need to have a engine.
    private final Engine engine; // Composition
    //private Engine engine; // Aggregation

    Car(Engine engine) {
        this.engine = engine;
    }

    // car start moving by starting engine
    public void move() {
        //if(engine != null)
        {
            engine.work();
            System.out.println("Car is moving ");
        }
    }
}

class Engine {
    // starting an engine
    public void work() {
        System.out.println("Engine of car has been started ");
    }
}
  
```



Review: What Is Generalization?

- A relationship among classes where one class shares the structure and/or behavior of one or more classes.
- Defines a hierarchy of abstractions where a subclass inherits from one or more superclasses.
 - Single inheritance
 - Multiple inheritance
- Is an “is a kind of” relationship.

