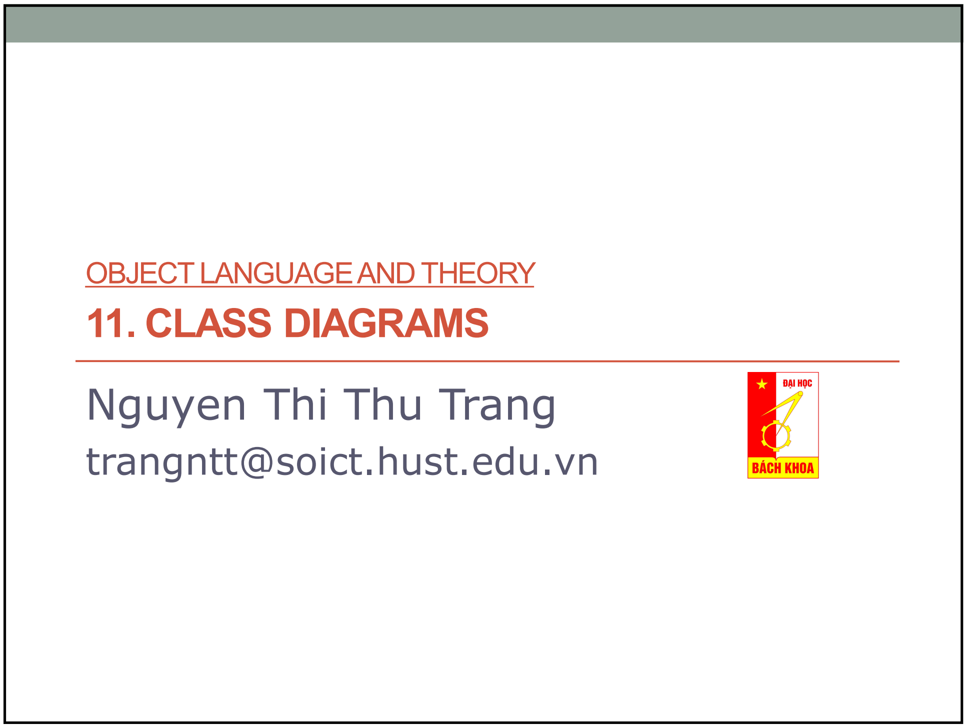


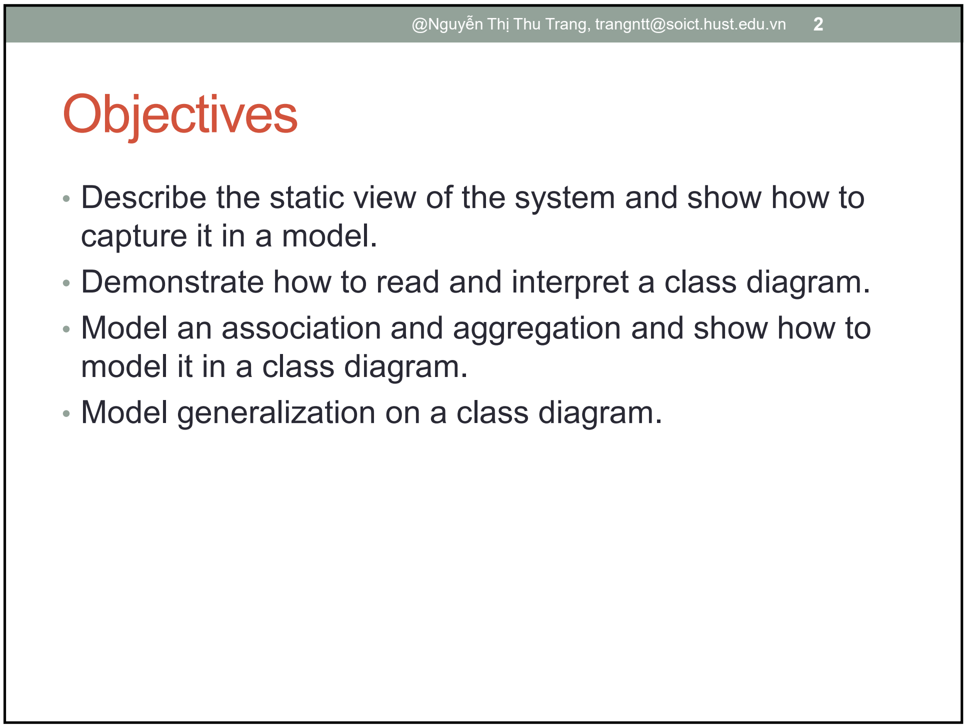
OBJECT LANGUAGE AND THEORY

11. CLASS DIAGRAMS

Nguyen Thi Thu Trang

trangntt@soict.hust.edu.vn

1



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.

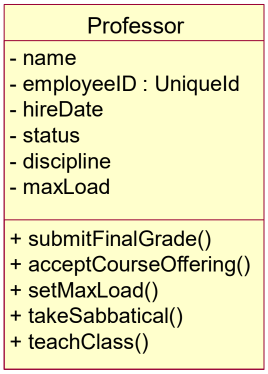
2

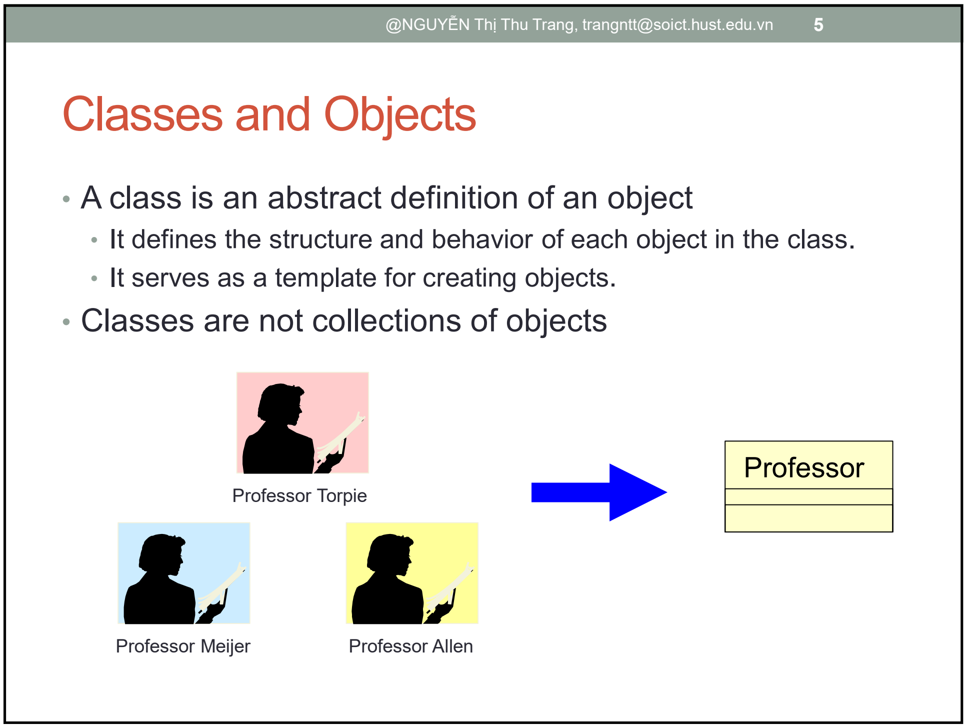
Content

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

1.1. Classes in the UML

- A class is represented using a rectangle with three compartments:
  - The class name
  - The structure (attributes)
  - The behavior (operations)

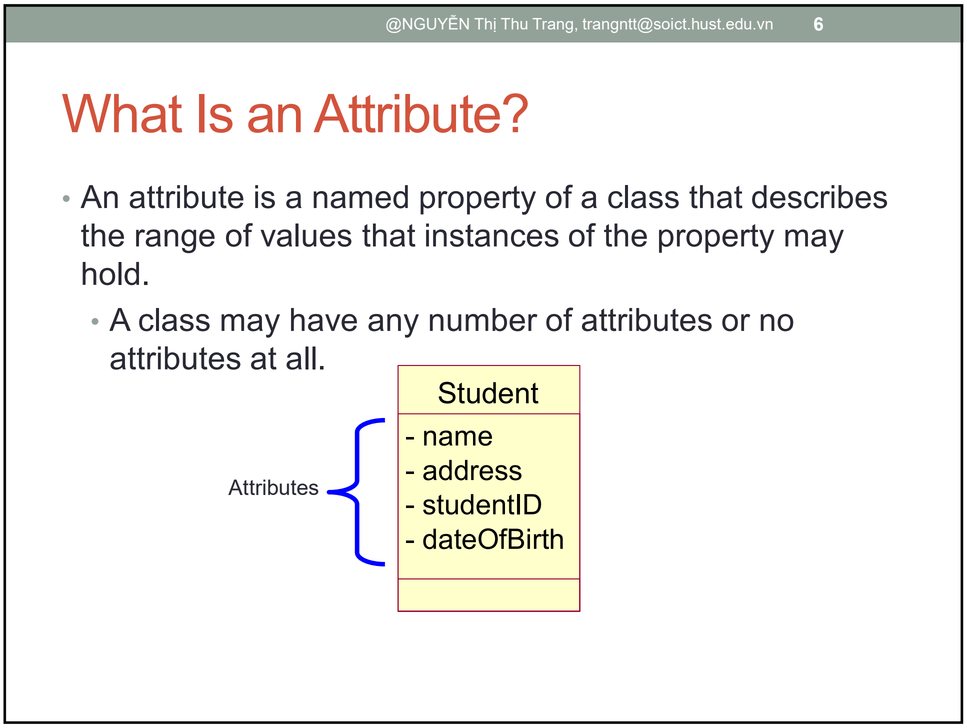




Classes and Objects

- A class is an abstract definition of an object
  - It defines the structure and behavior of each object in the class.
  - It serves as a template for creating objects.
- Classes are not collections of objects

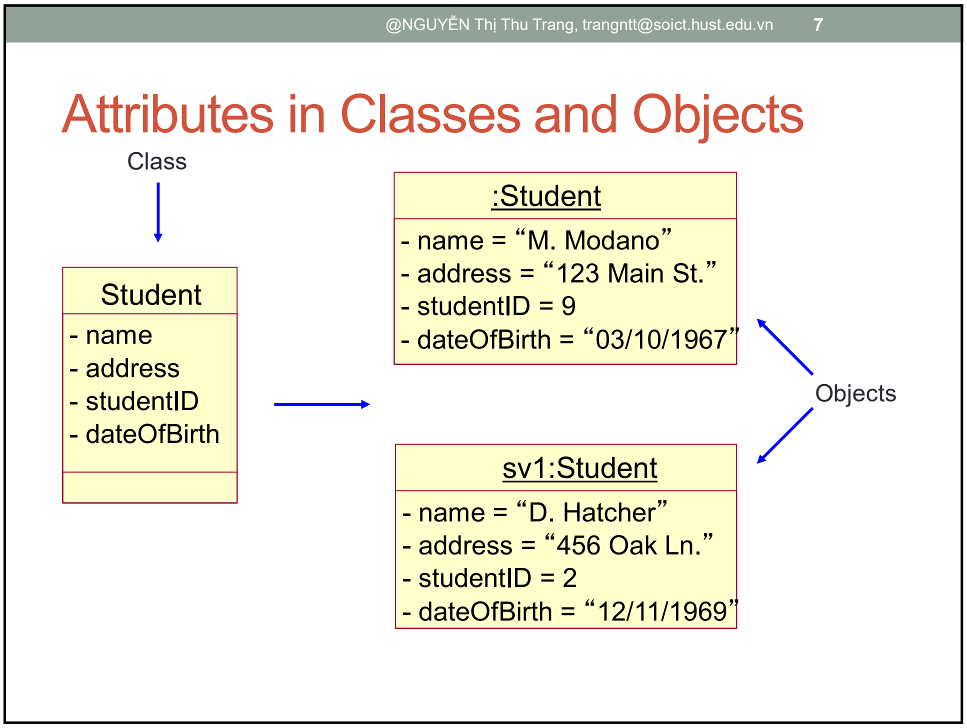
5



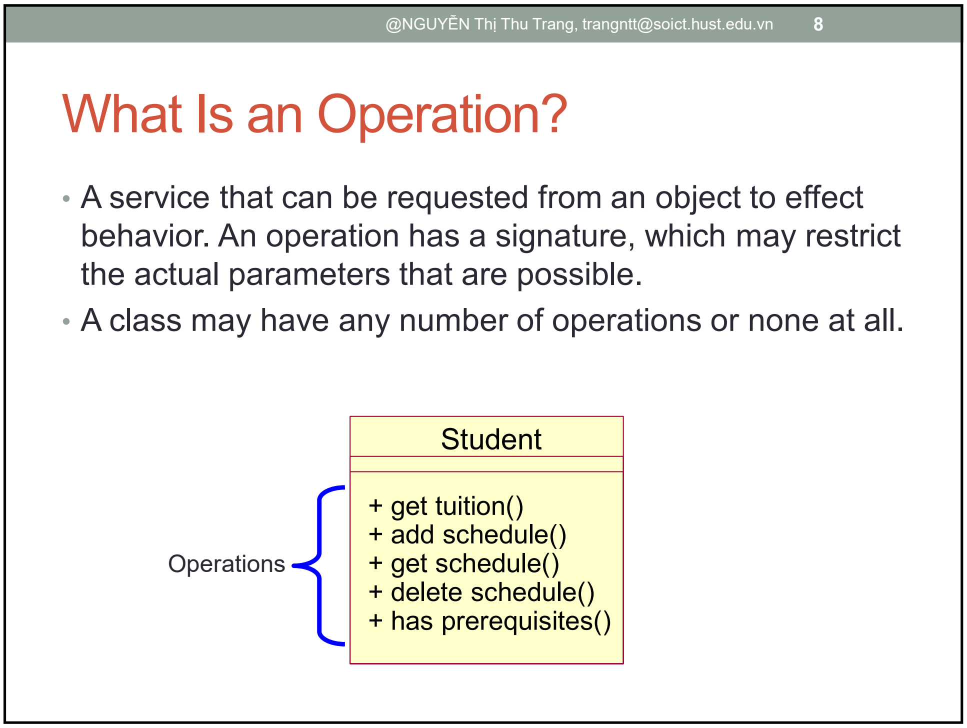
What Is an Attribute?

- An attribute is a named property of a class that describes the range of values that instances of the property may hold.
  - A class may have any number of attributes or no attributes at all.

6

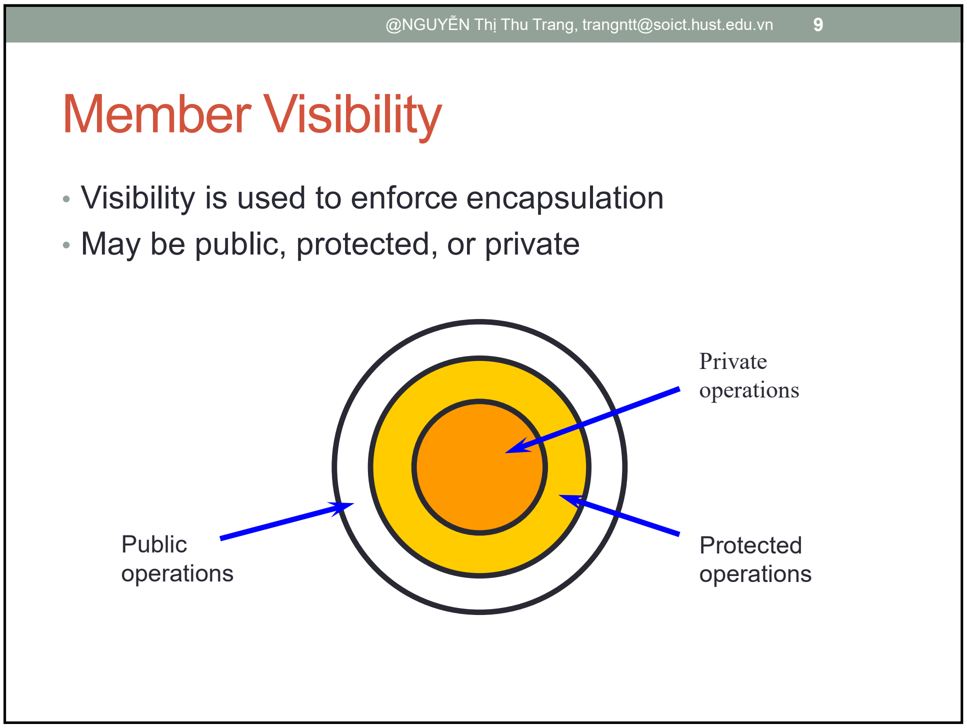


Attributes in Classes and Objects



What Is an Operation?

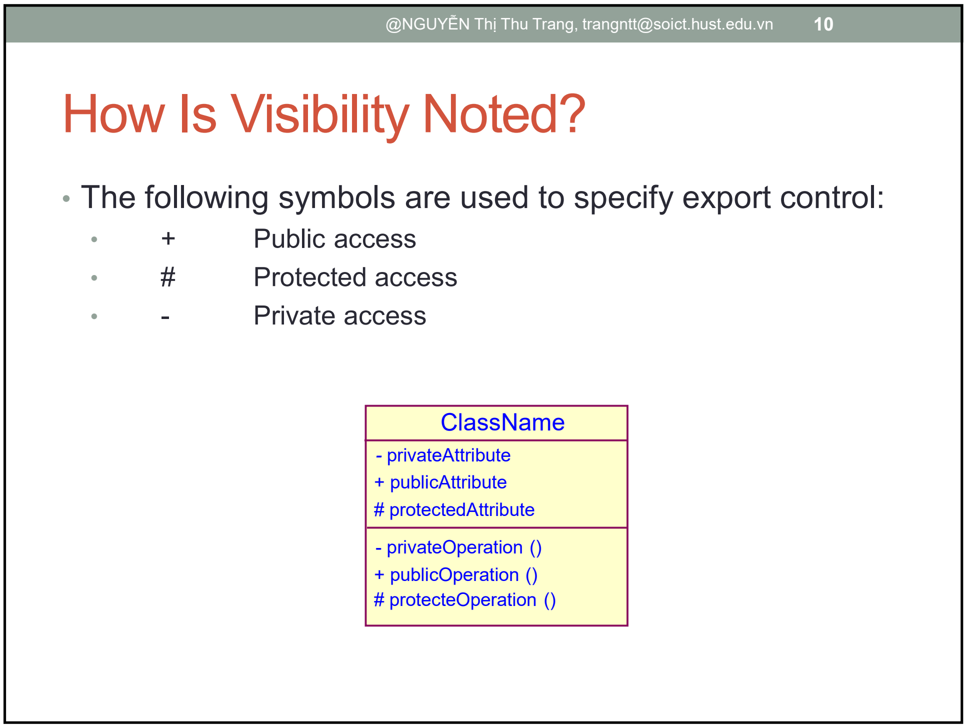
- A service that can be requested from an object to effect behavior. An operation has a signature, which may restrict the actual parameters that are possible.
- A class may have any number of operations or none at all.



Member Visibility

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

9



How Is Visibility Noted?

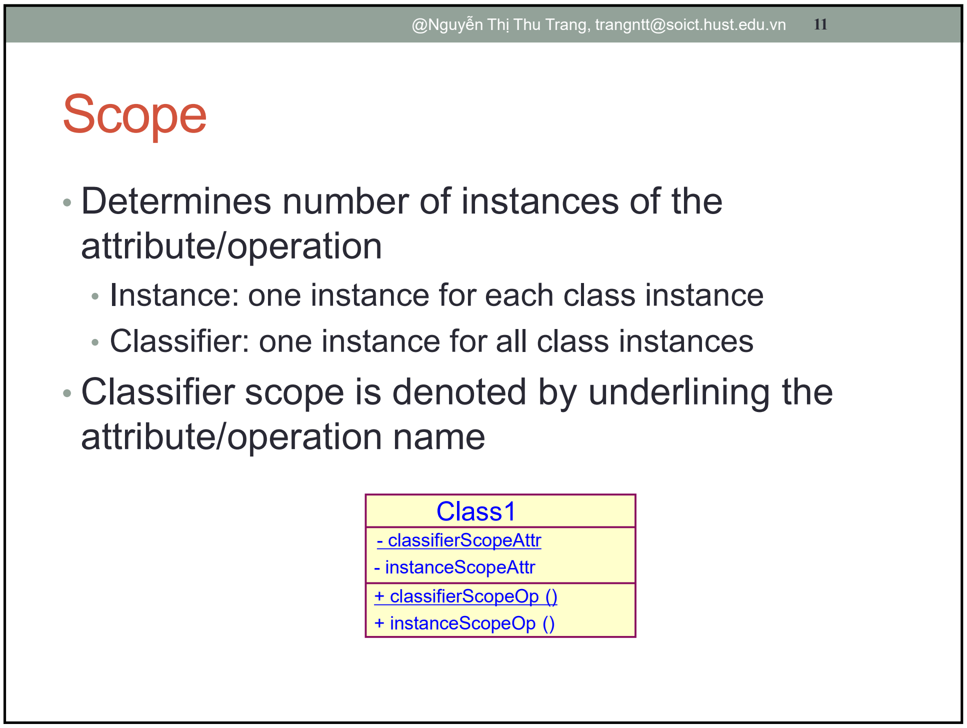
- The following symbols are used to specify export control:
  - + Public access
  - # Protected access
  - - Private access

ClassName

- privateAttribute
+ publicAttribute
# protectedAttribute

- privateOperation ()
+ publicOperation ()
# protecteOperation ()

10



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

Class1

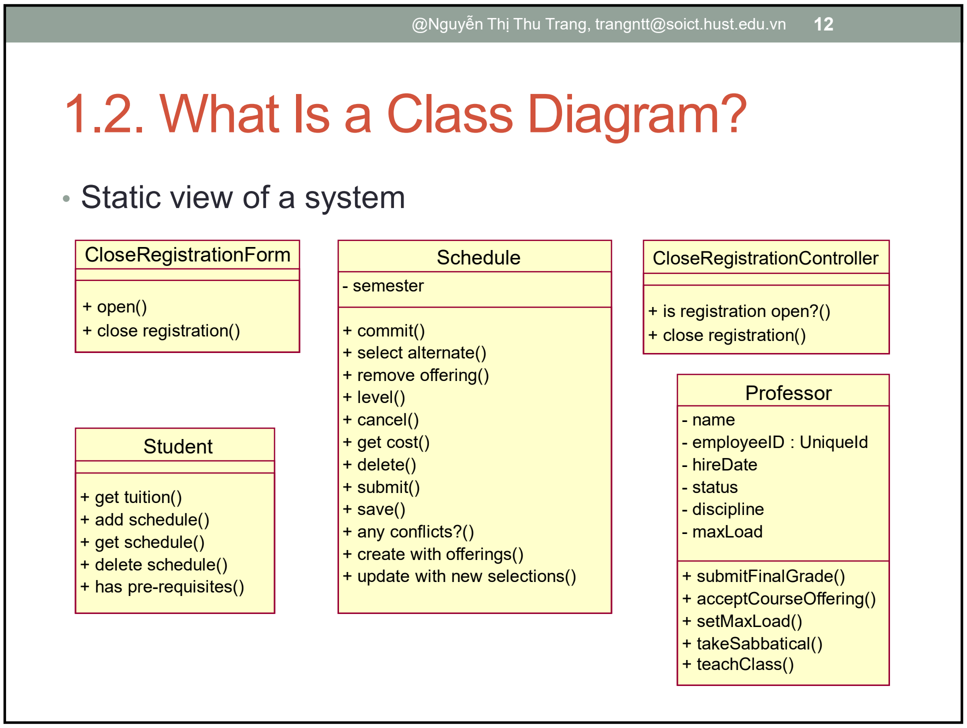
- classifierScopeAttr

- instanceScopeAttr

+ classifierScopeOp ()

+ instanceScopeOp ()

11



1.2. What Is a Class Diagram?

- Static view of a system

CloseRegistrationForm

+ open()

+ close registration()

Schedule

- semester

+ commit()

+ select alternate()

+ remove offering()

+ level()

+ cancel()

+ get cost()

+ delete()

+ submit()

+ save()

+ any conflicts?()

+ create with offerings()

+ update with new selections()

CloseRegistrationController

+ is registration open?()

+ close registration()

Student

+ get tuition()

+ add schedule()

+ get schedule()

+ delete schedule()

+ has pre-requisites()

Professor

- name

- employeeID : UniqueId

- hireDate

- status

- discipline

- maxLoad

+ submitFinalGrade()

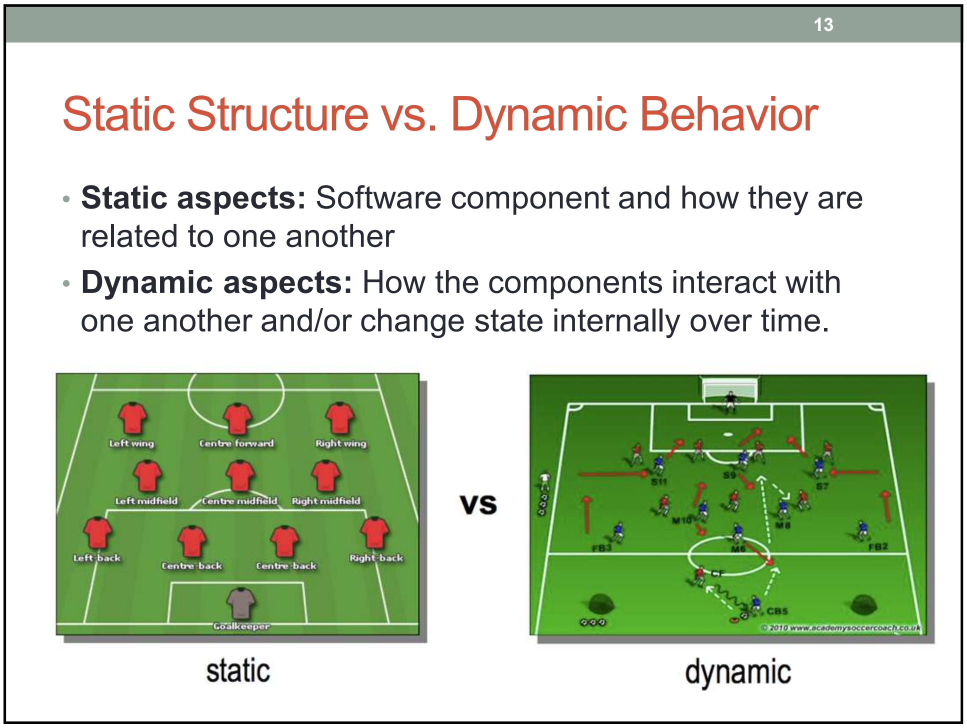
+ acceptCourseOffering()

+ setMaxLoad()

+ takeSabbatical()

+ teachClass()

12



Static Structure vs. Dynamic Behavior

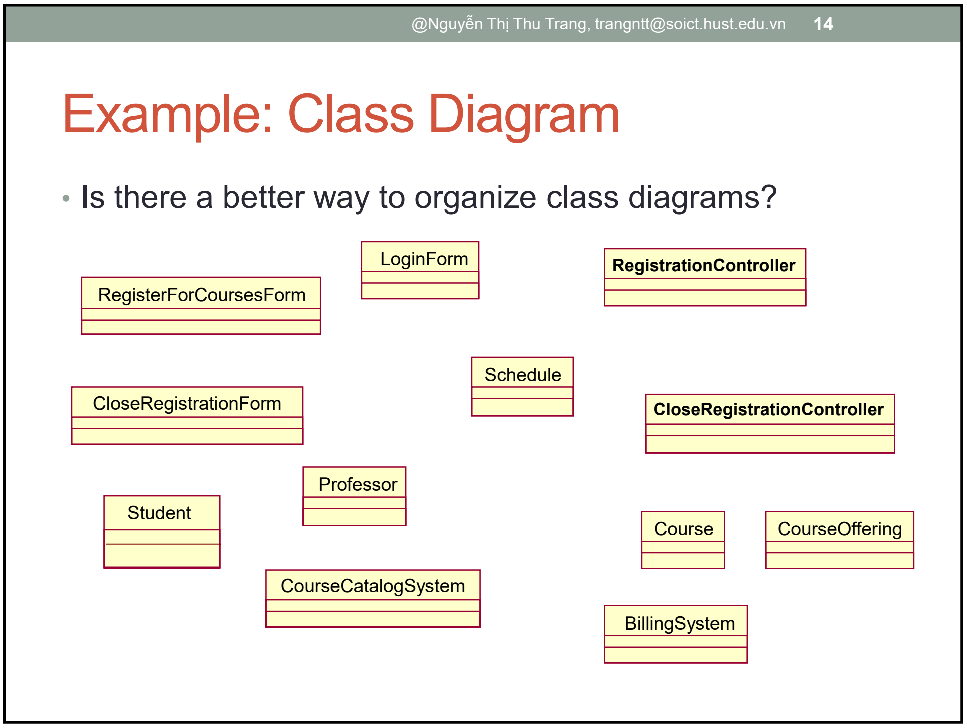
- **Static aspects:** Software component and how they are related to one another
- **Dynamic aspects:** How the components interact with one another and/or change state internally over time.

Left wing
Centre forward
Right wing
Left midfield
Centre midfield
Right midfield
Left back
Centre back
Centre back
Right back
Goalkeeper
vs
S11
S9
S7
M10
M8
FB3
M6
FB2
CF
CB5
© 2010 www.academysoccercoach.co.uk

static

dynamic

13



Example: Class Diagram

- Is there a better way to organize class diagrams?

LoginForm

RegisterForCoursesForm

RegistrationController

Schedule

CloseRegistrationForm

CloseRegistrationController

Professor

Student

Course

CourseOffering

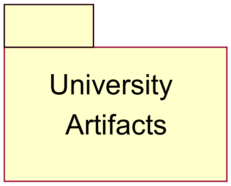
CourseCatalogSystem

BillingSystem

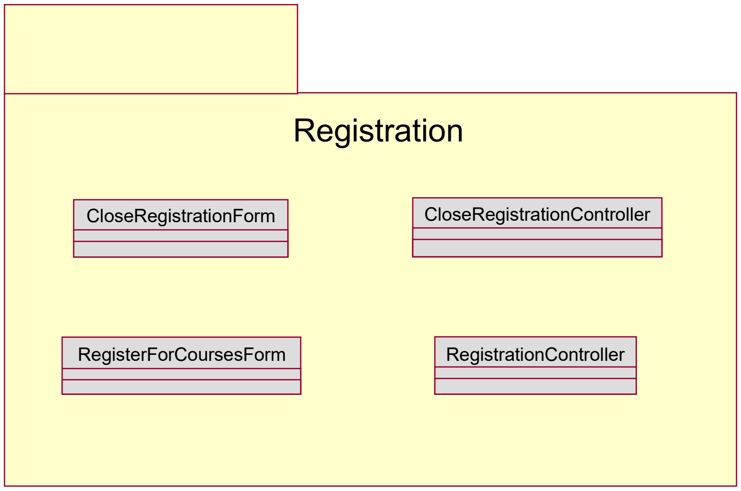
14

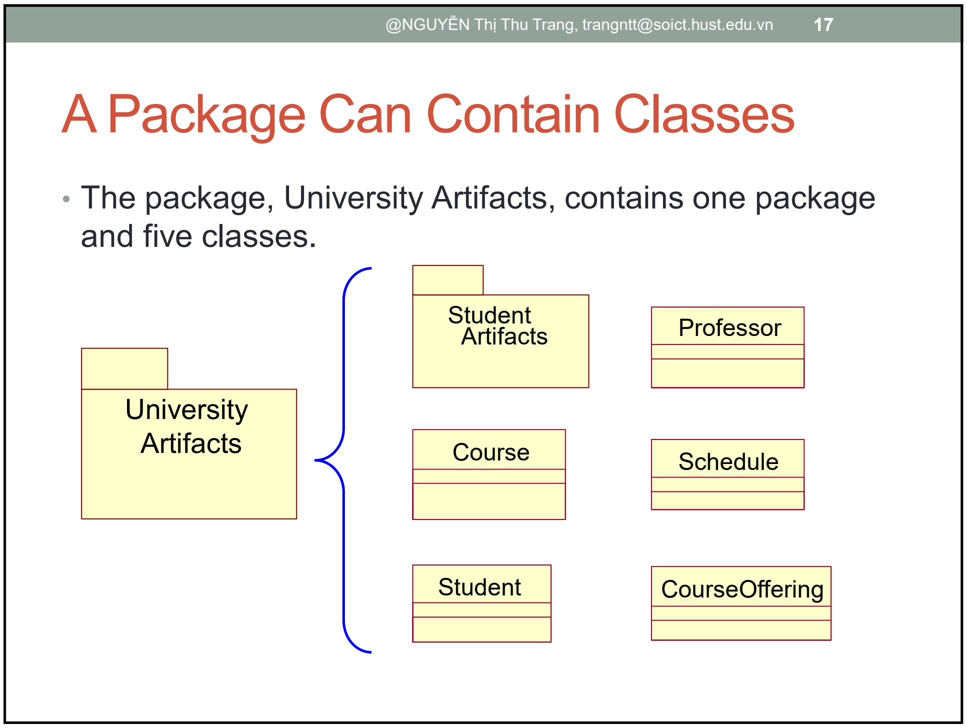
Review: What Is a Package?

- 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



Example: Registration Package



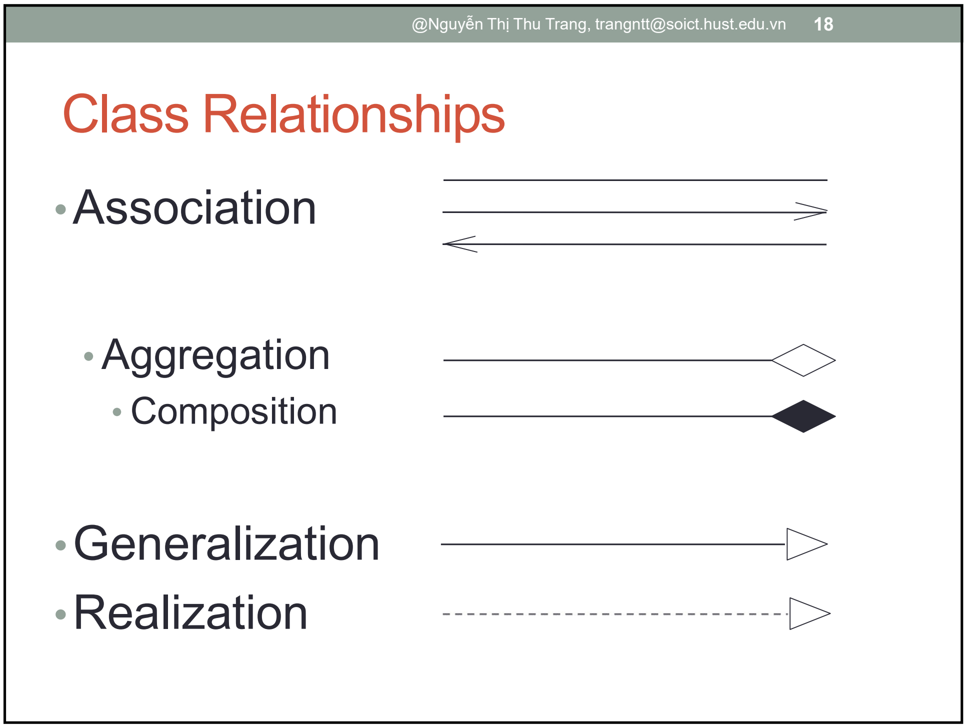


@NGUYỄN Thị Thu Trang, trangntt@soict.hust.edu.vn 17

A Package Can Contain Classes

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

17



@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 18

Class Relationships

- Association
  - Aggregation
    - Composition
- Generalization
- Realization

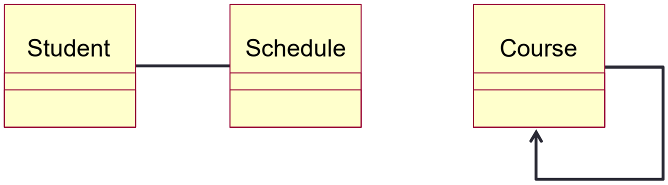
18

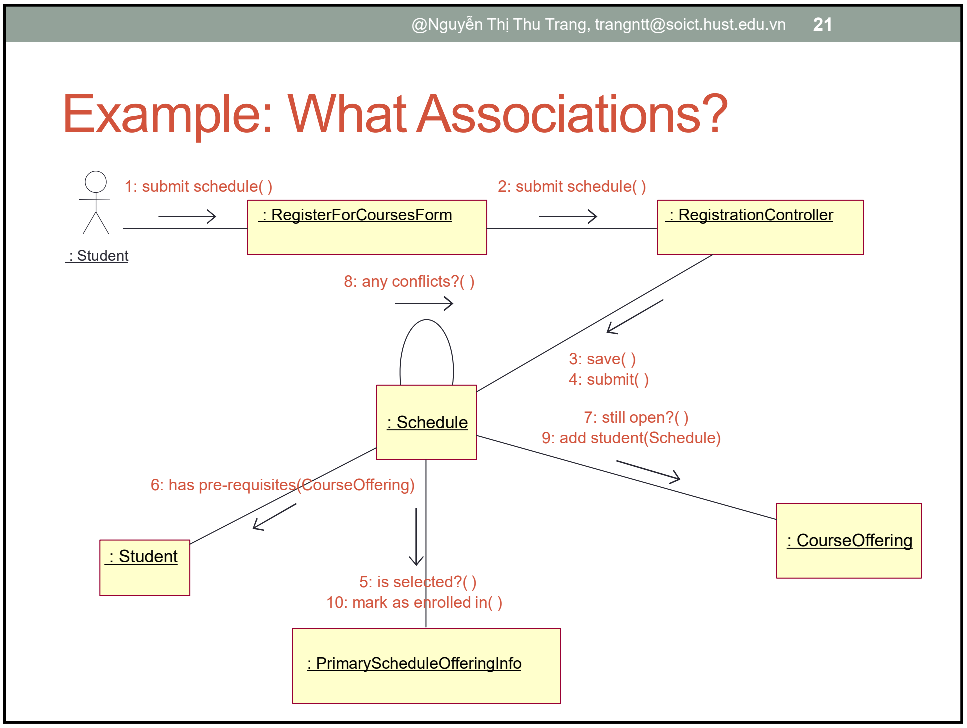
Content

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

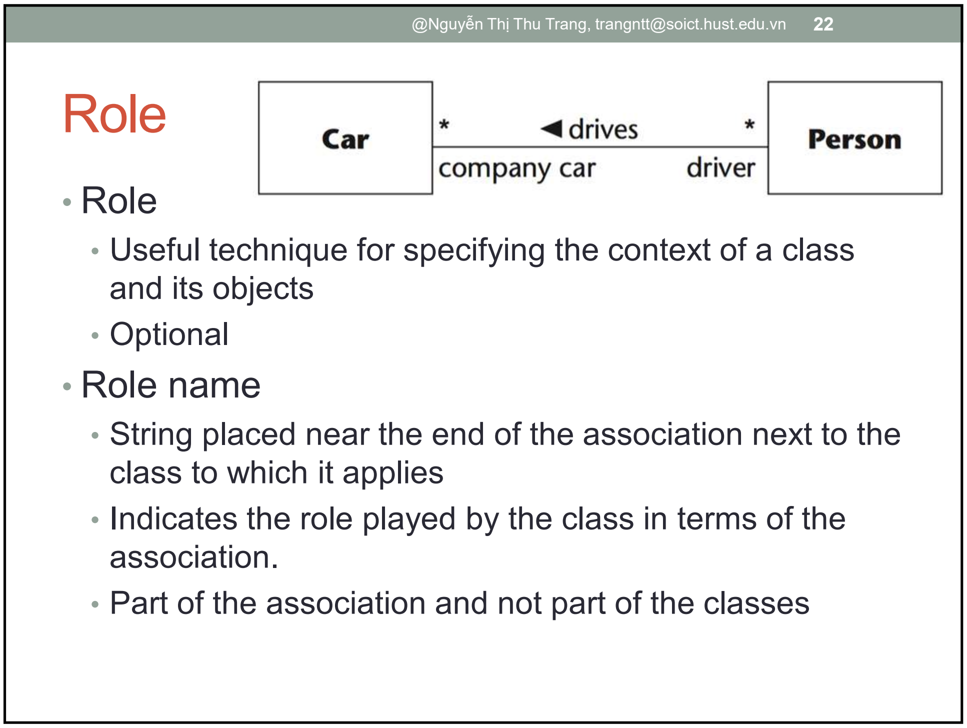
What Is an Association?

- The semantic relationship between two or more classifiers that specifies connections among their instances.
- A structural relationship specifying that objects of one thing are connected to objects of another thing.





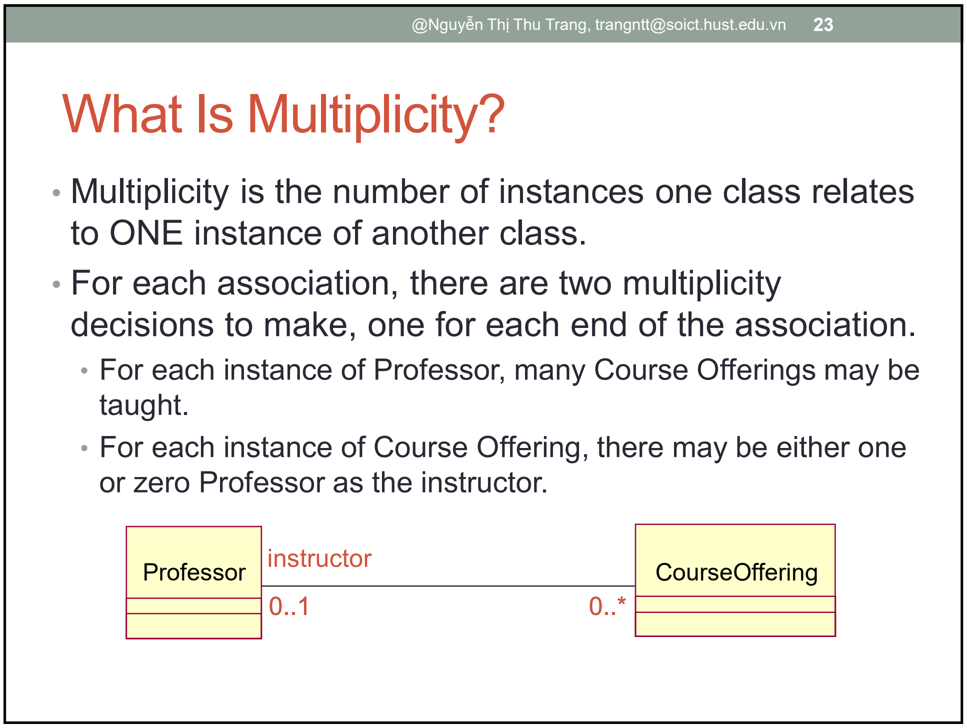
Example: What Associations?



Role

Car
*
◀ drives
*
Person
company car
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

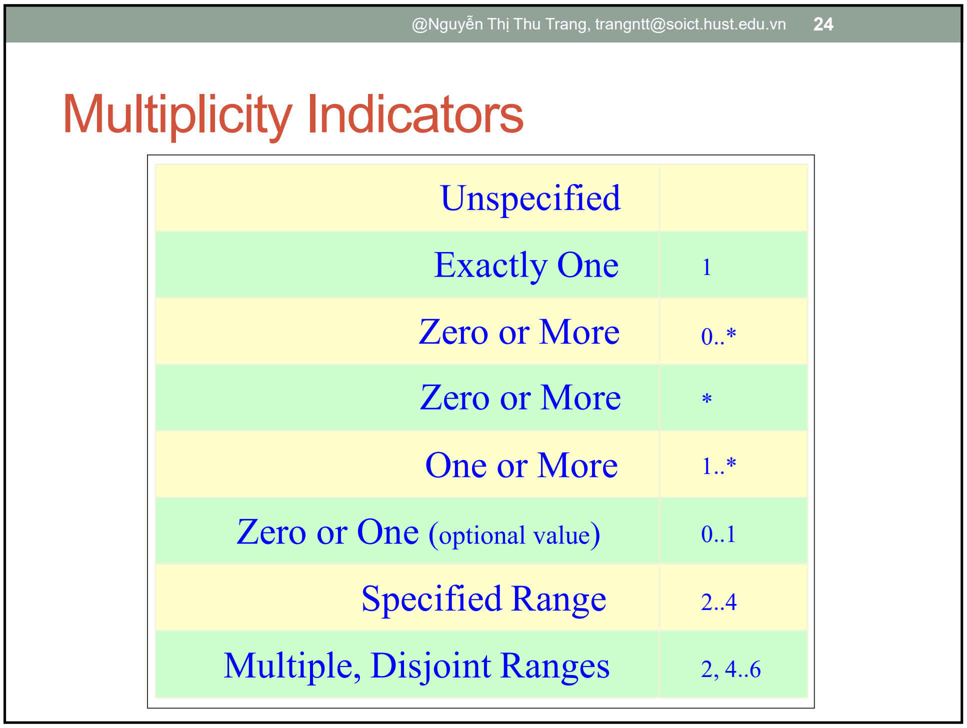


@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 23

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.

23

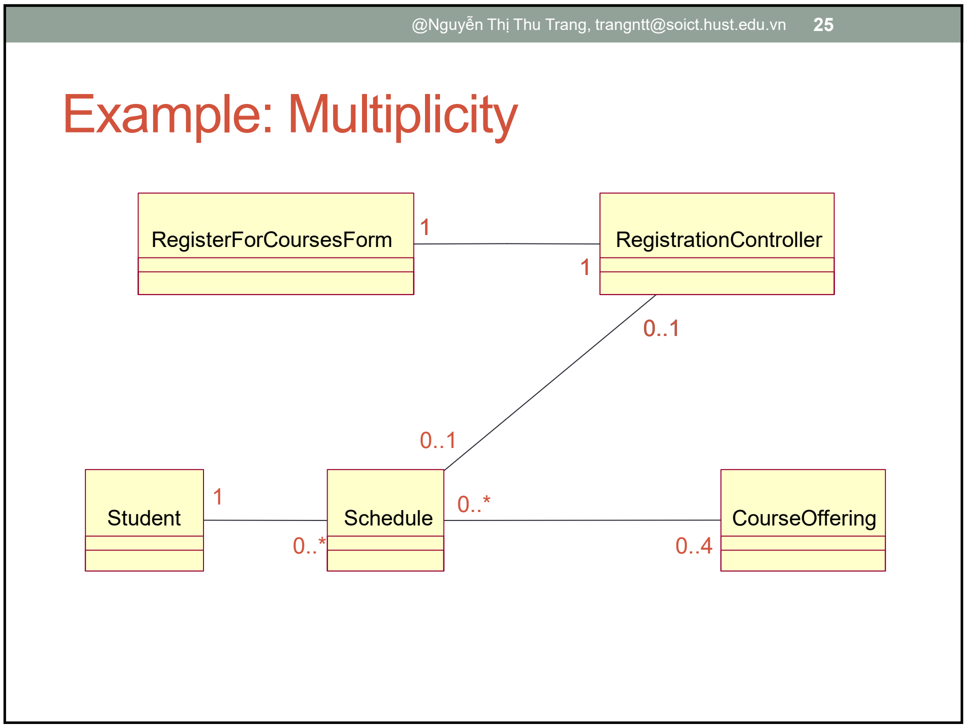


@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 24

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

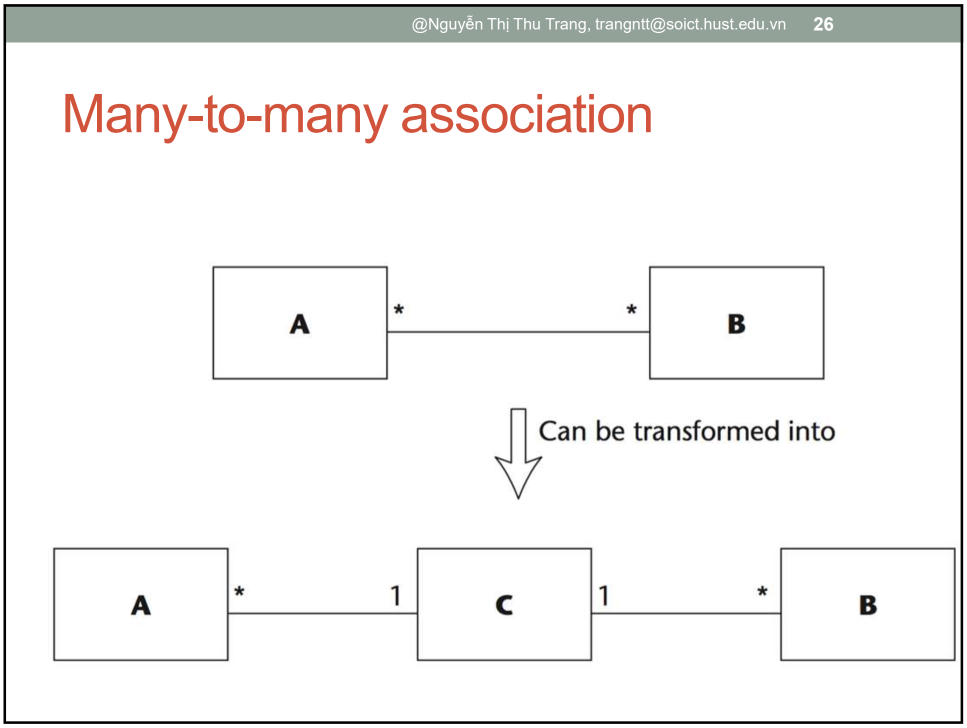
24



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

Example: Multiplicity

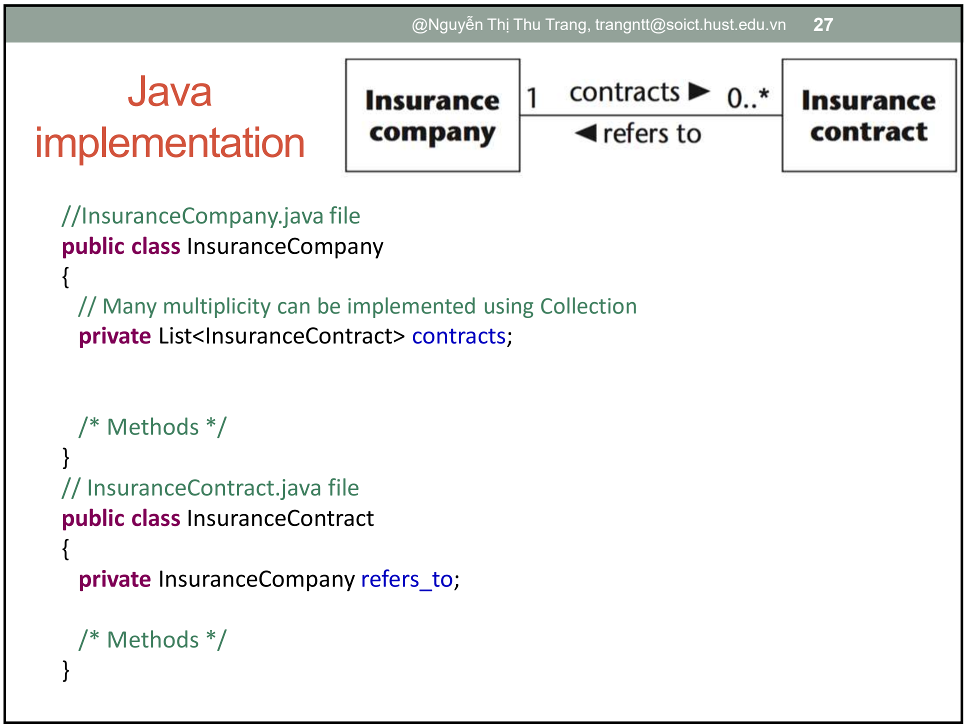
25



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

Many-to-many association

26



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

Java implementation

Insurance company 1 contracts ▶ 0..* Insurance contract

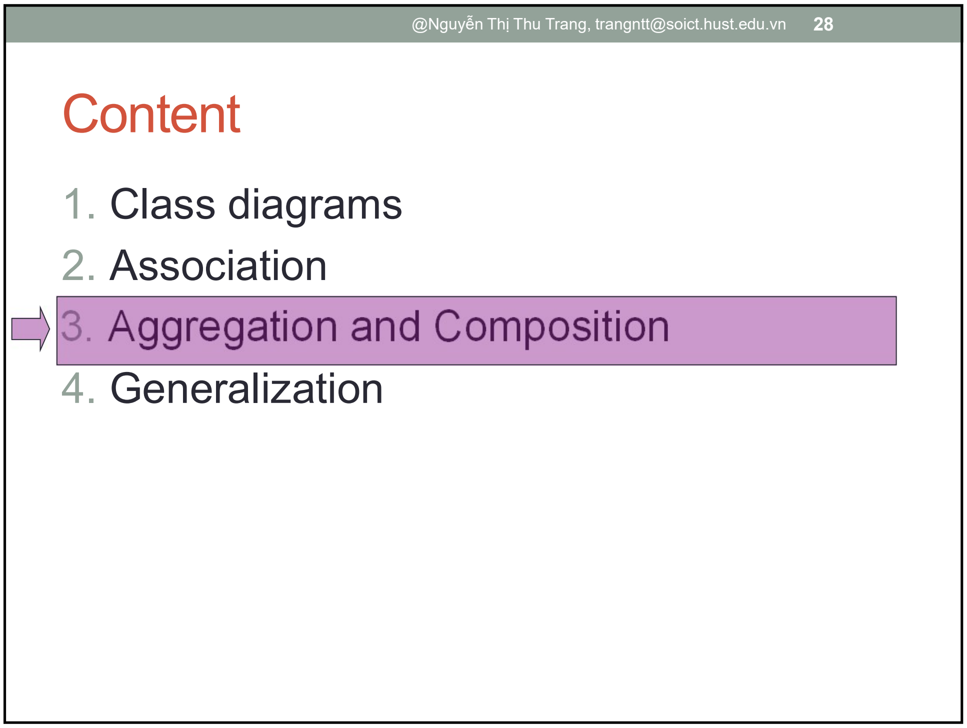
◀ refers to

```
//InsuranceCompany.java file
public class InsuranceCompany
{
  // Many multiplicity can be implemented using Collection
  private List<InsuranceContract> contracts;


  /* Methods */
}
// InsuranceContract.java file
public class InsuranceContract
{
  private InsuranceCompany refers_to;

  /* Methods */
}
```

27

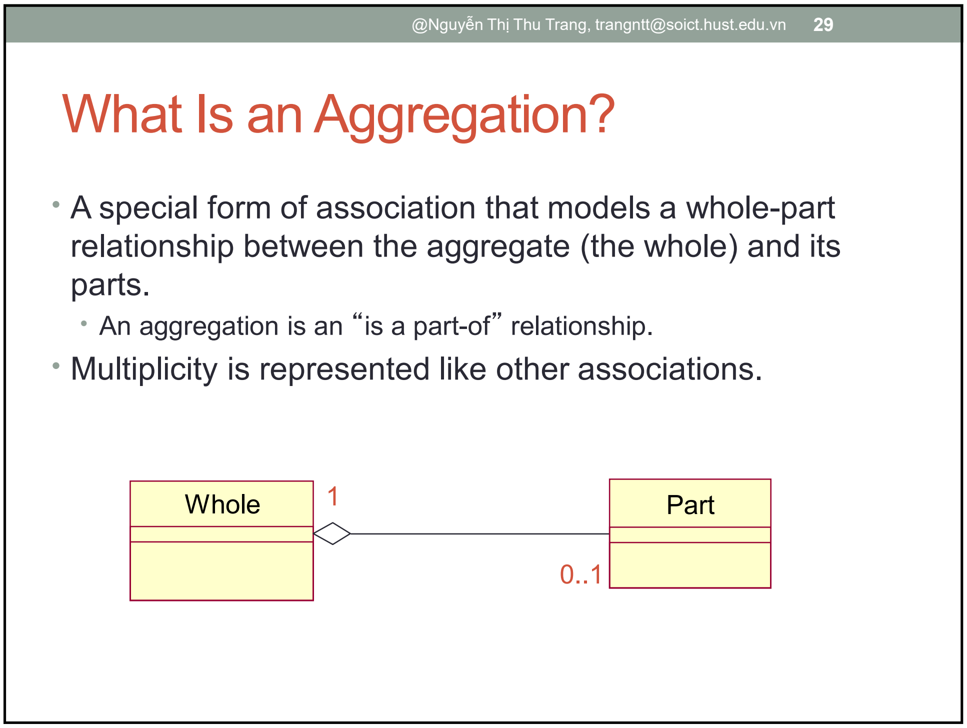


@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 28

Content

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

28



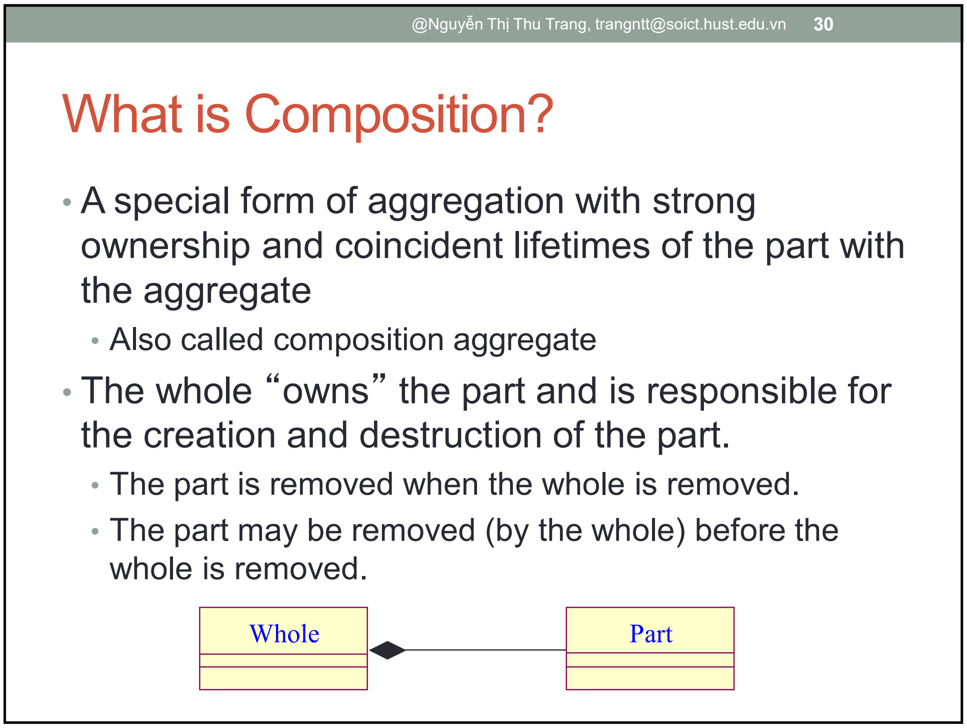
@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 29

What Is an Aggregation?

- A special form of association that models a whole-part relationship between the aggregate (the whole) and its parts.
  - An aggregation is an “is a part-of” relationship.
- Multiplicity is represented like other associations.

Whole
1
Part
0..1

29

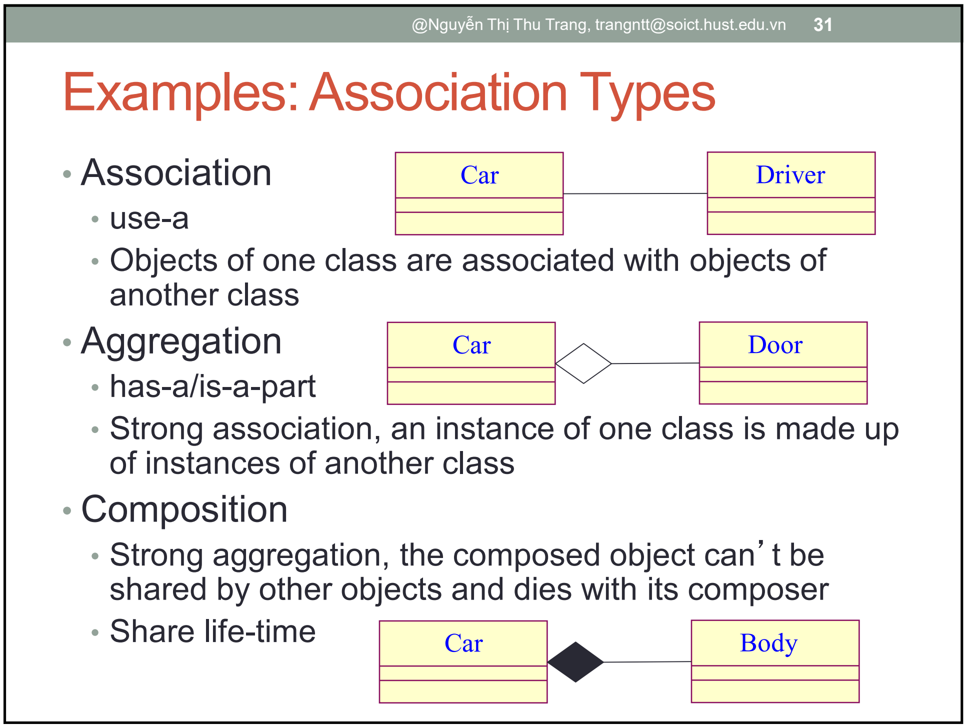


@Nguyễn Thị Thu Trang, trangntt@soict.hust.edu.vn 30

What is Composition?

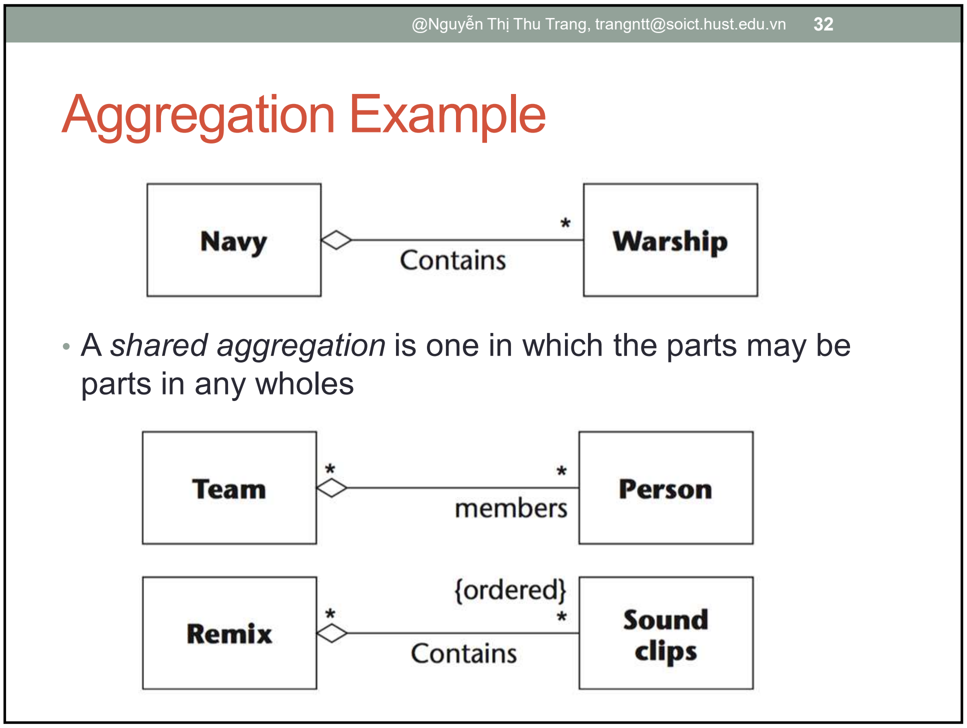
- A special form of aggregation with strong ownership and coincident lifetimes of the part with the aggregate
  - Also called composition aggregate
- The whole “owns” the part and is responsible for the creation and destruction of the part.
  - The part is removed when the whole is removed.
  - The part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

30



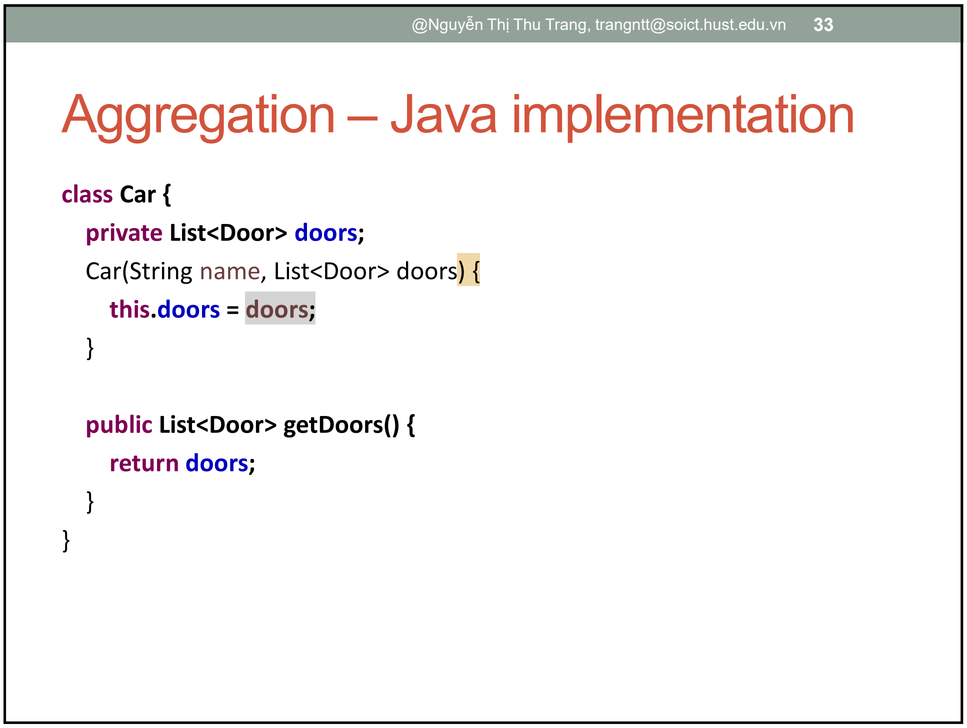
Examples: Association Types

- 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



Aggregation Example

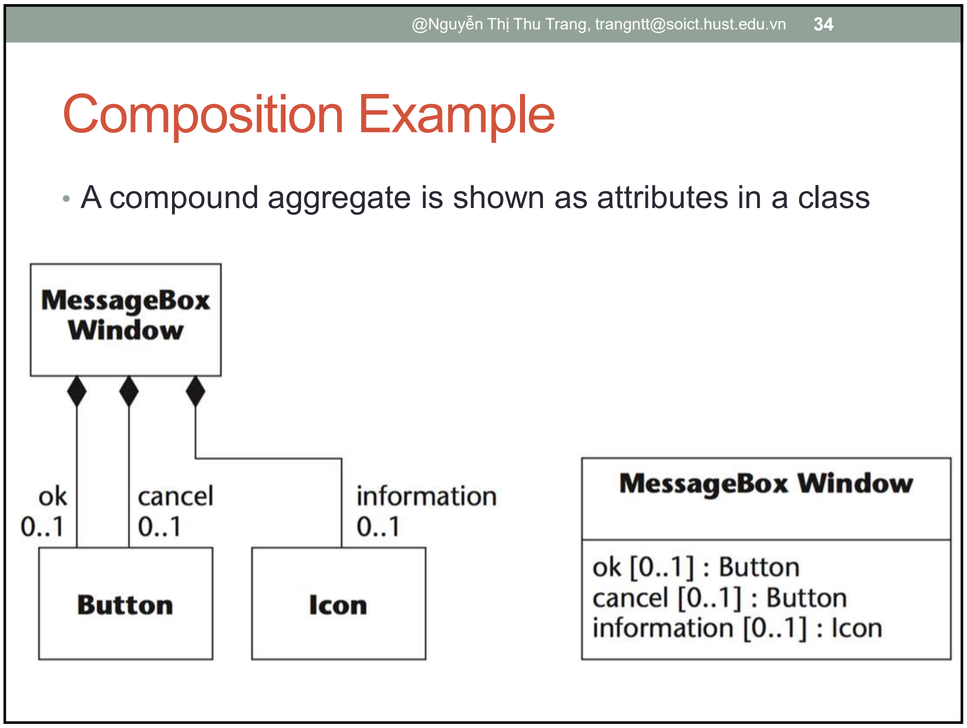
- A *shared aggregation* is one in which the parts may be parts in any wholes



Aggregation – Java implementation

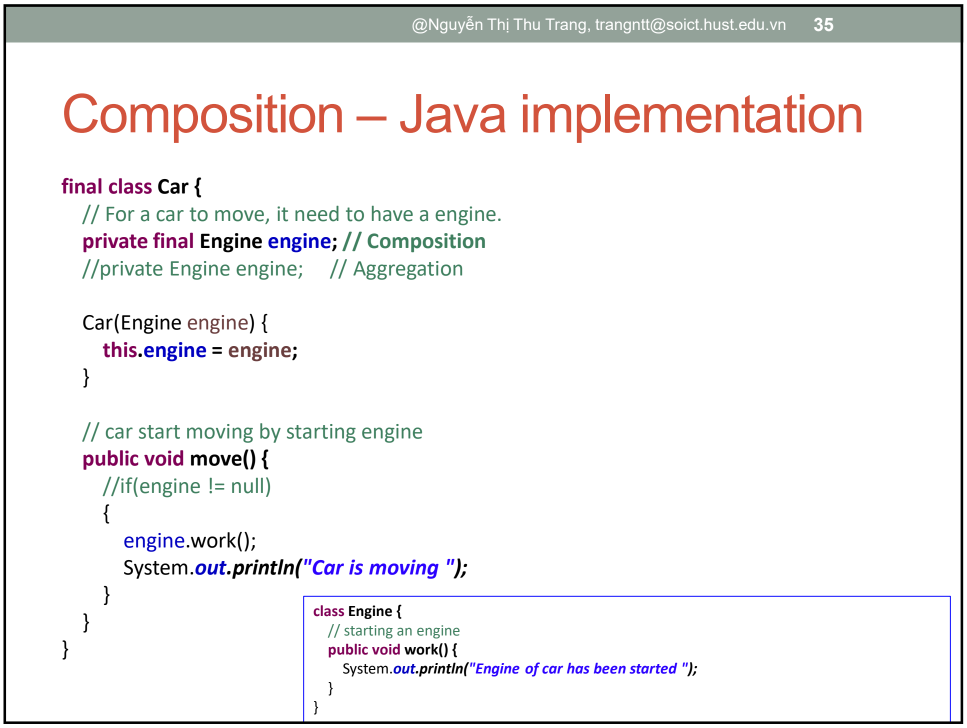
```
class Car {
   private List<Door> doors;
   Car(String name, List<Door> doors) {
      this.doors = doors;
   }

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



Composition Example

- A compound aggregate is shown as attributes in a class



Composition – Java implementation

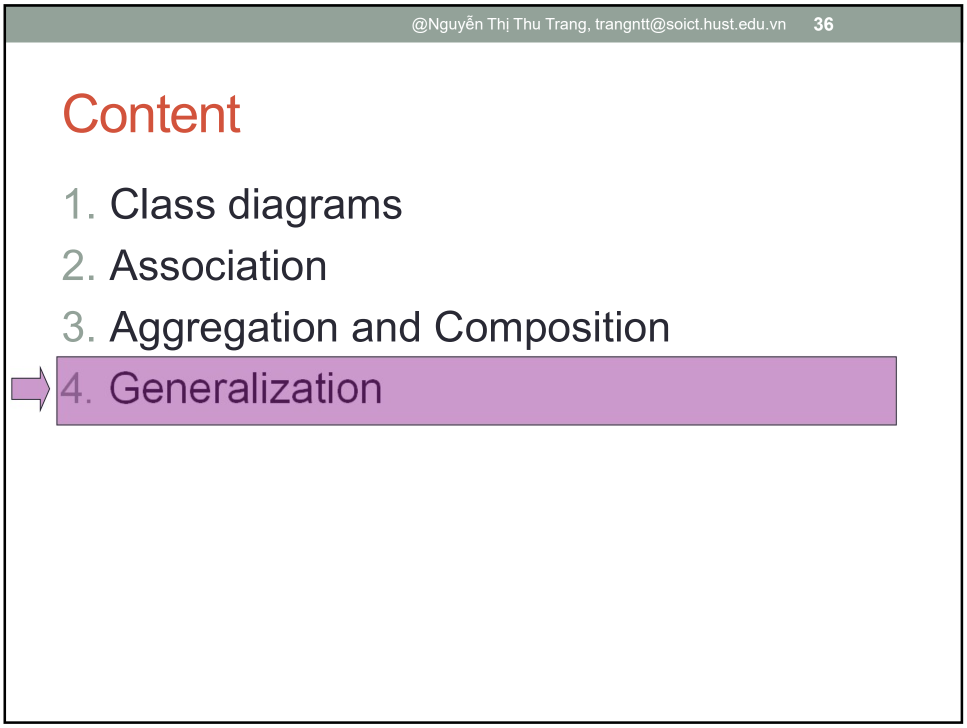
```
final class Car {
   // For a car to move, it need to have a engine.
   private final Engine engine; // Composition
   //private Engine engine;    // Aggregation

   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 ");
   }
}
```

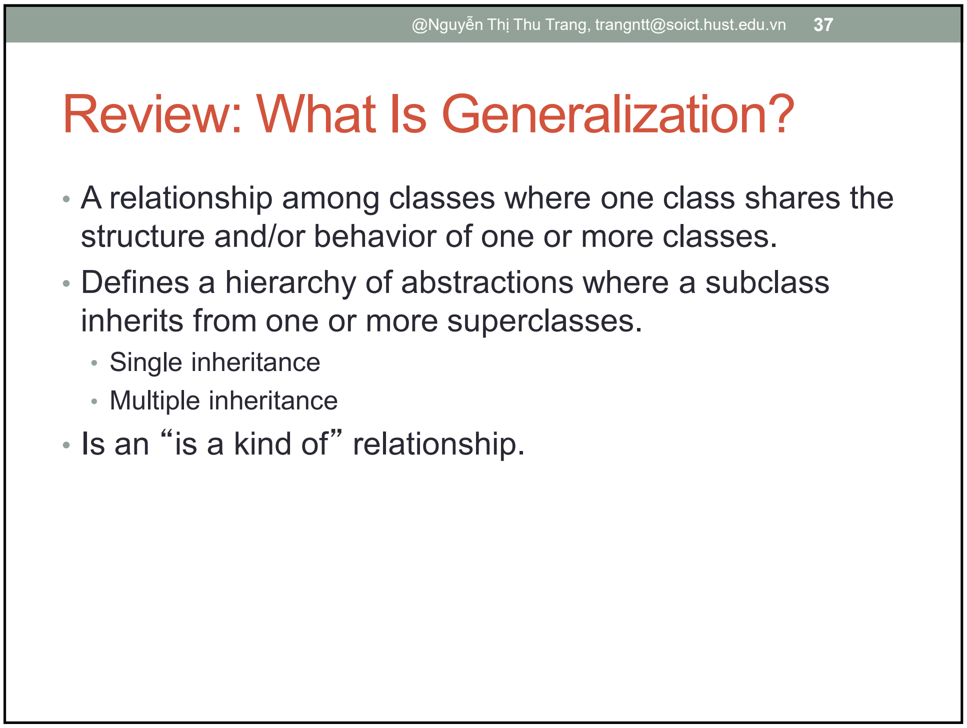
35



Content

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

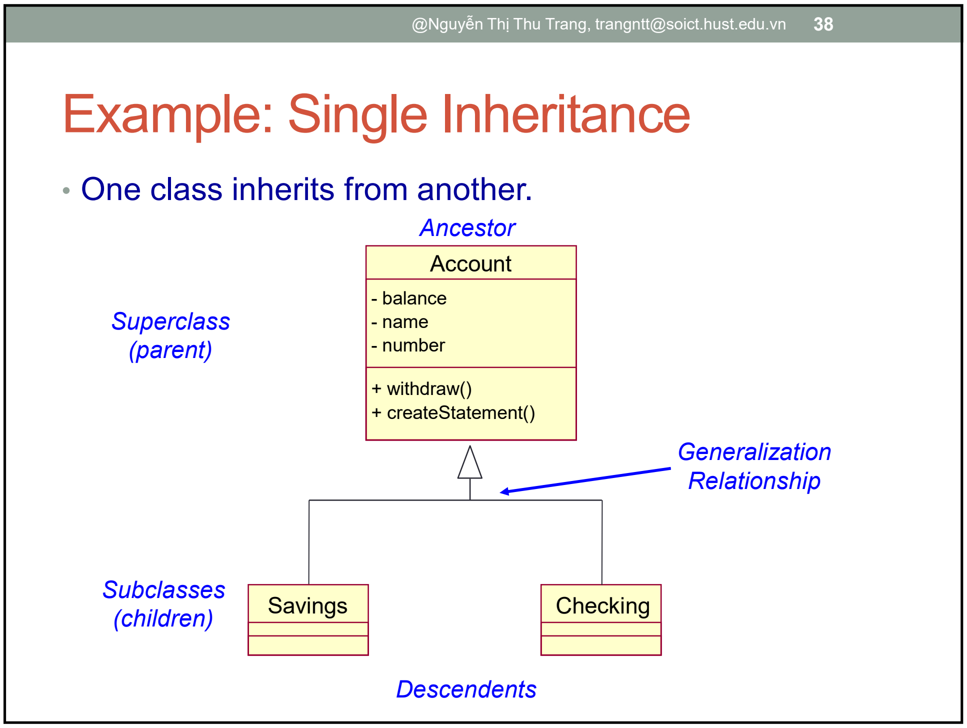
36



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.

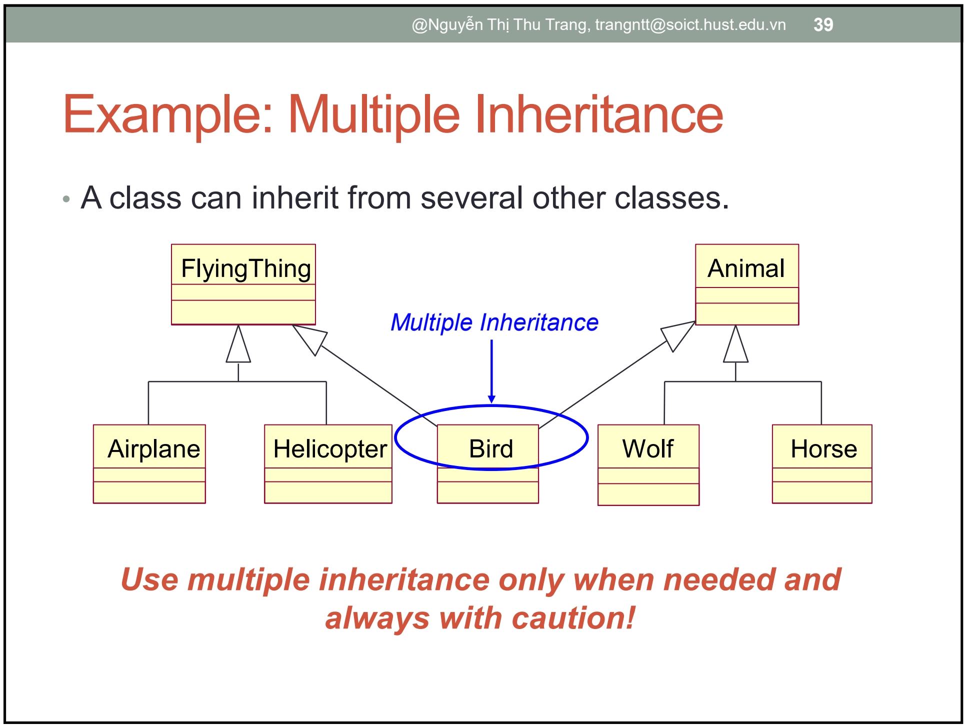
37



Example: Single Inheritance

- One class inherits from another.

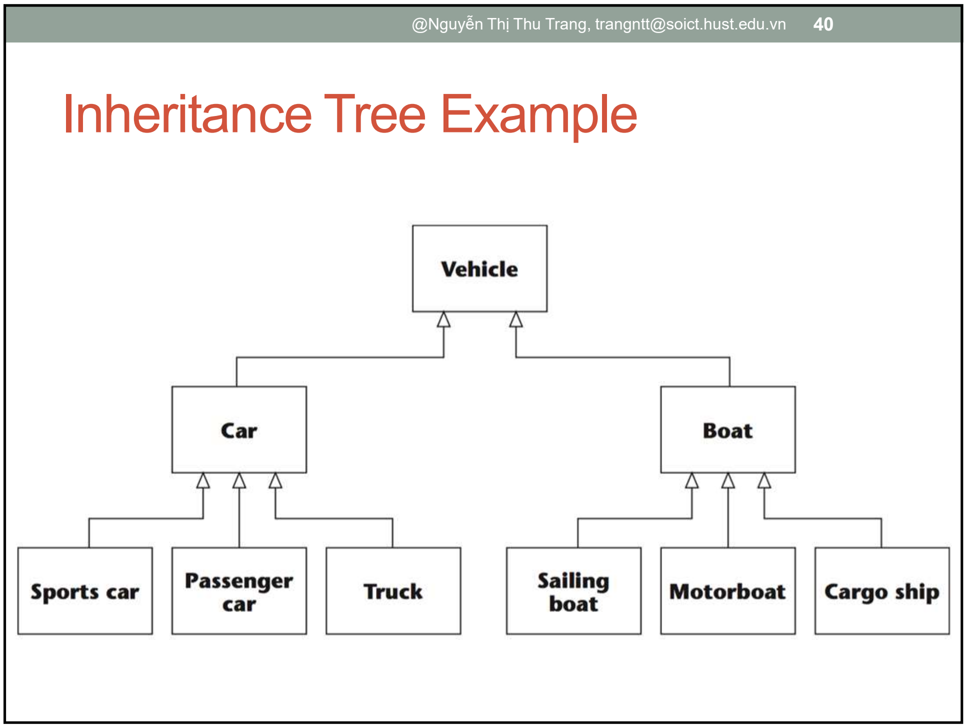
38



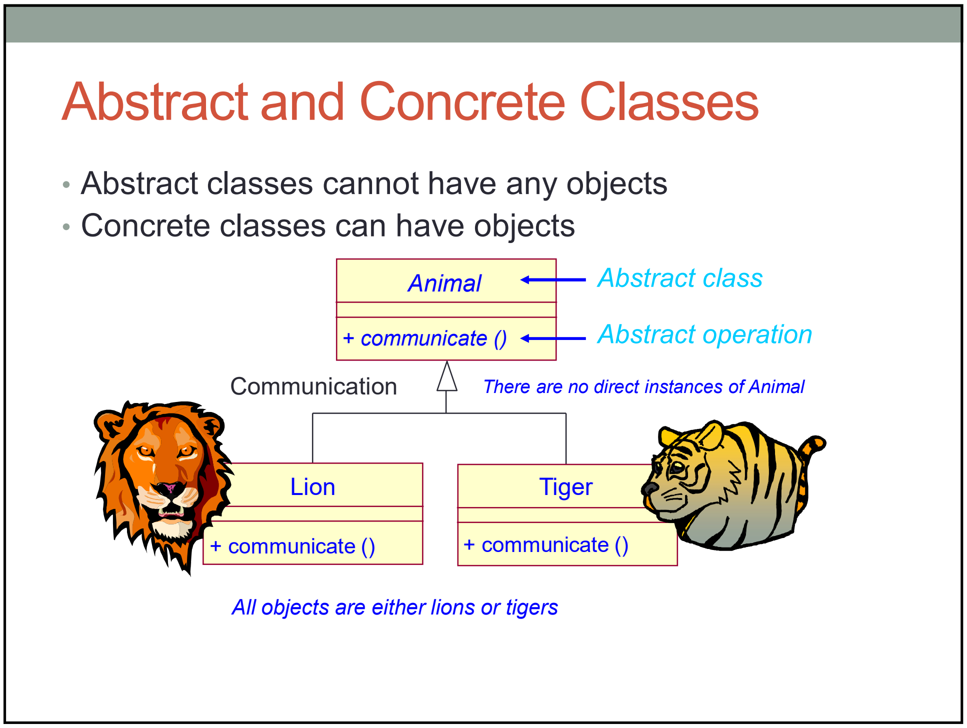
Example: Multiple Inheritance

- A class can inherit from several other classes.

Use multiple inheritance only when needed and always with caution!



Inheritance Tree Example

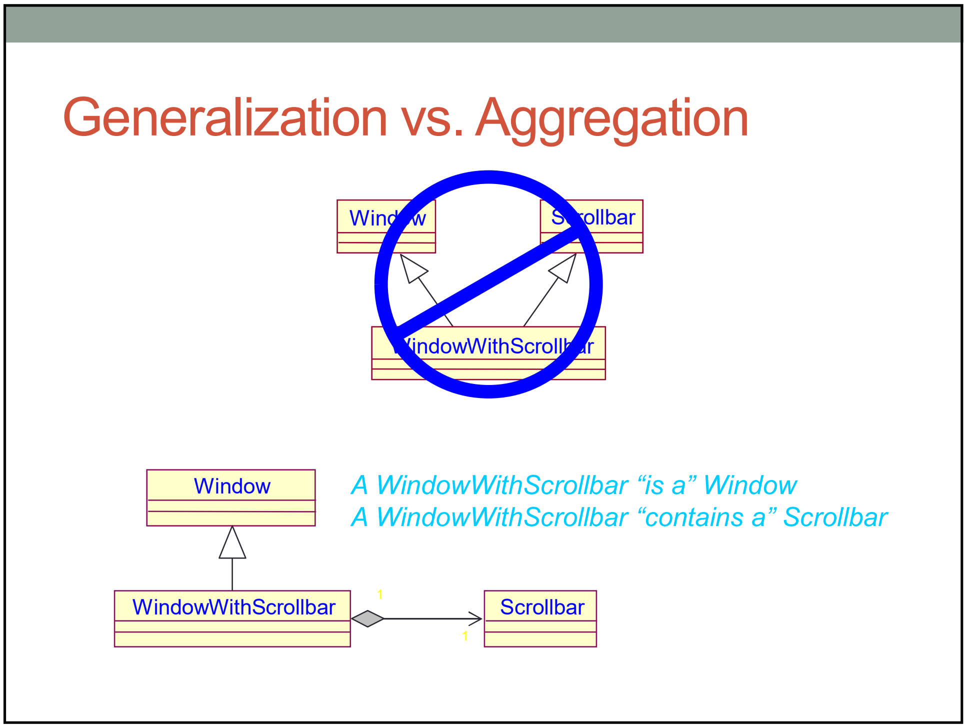


Abstract and Concrete Classes

- Abstract classes cannot have any objects
- Concrete classes can have objects

All objects are either lions or tigers

41

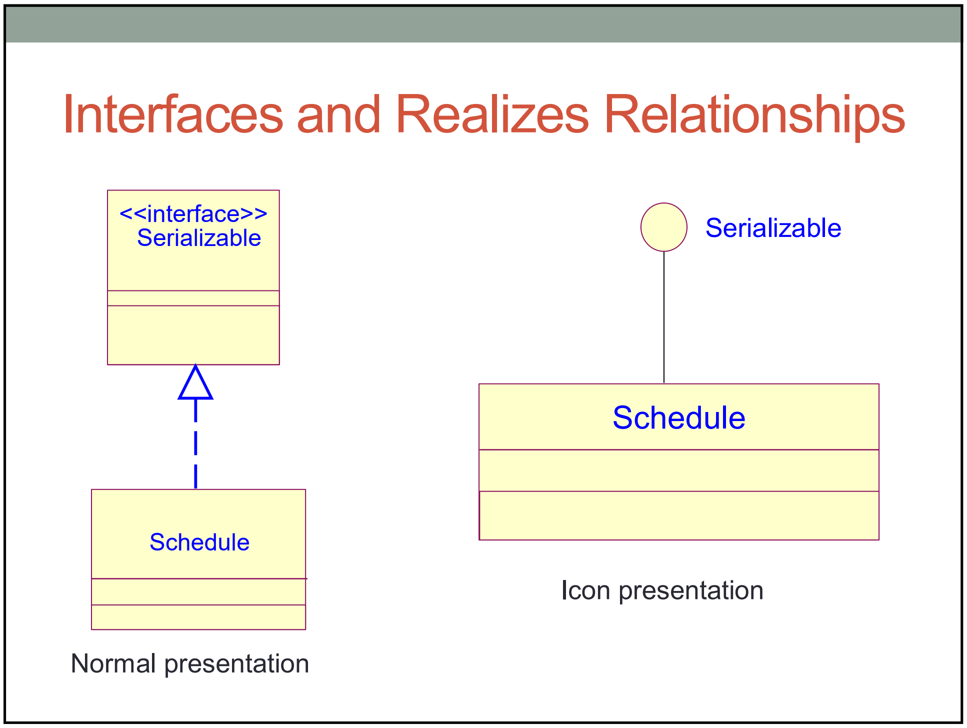


Generalization vs. Aggregation

A WindowWithScrollbar "is a" Window

A WindowWithScrollbar "contains a" Scrollbar

42



Interfaces and Realizes Relationships

<<interface>>
Serializable

Schedule

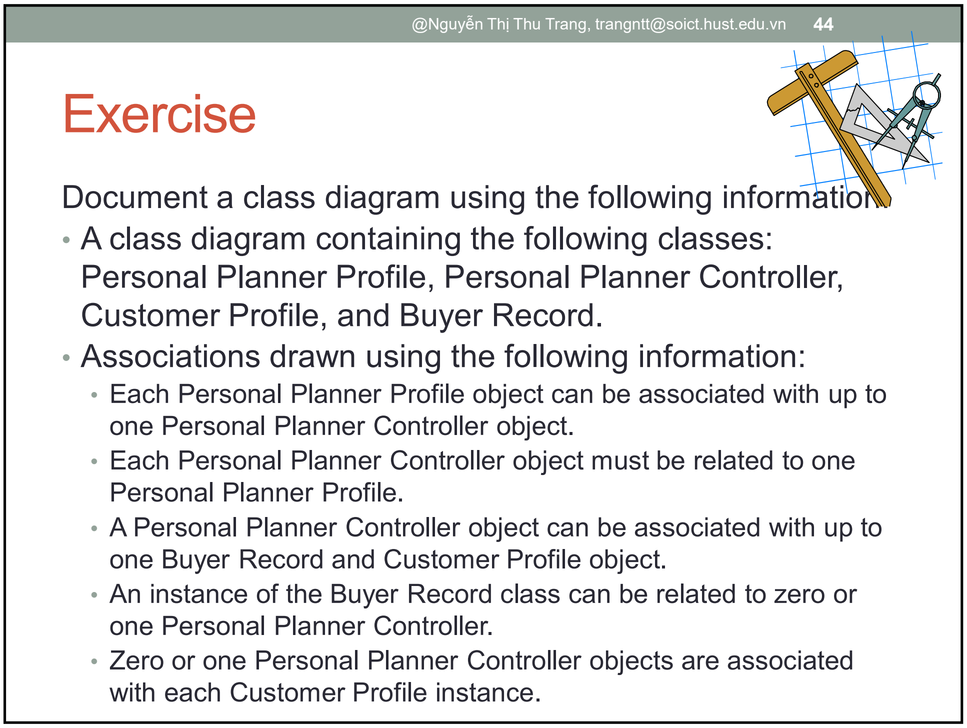
Normal presentation

Serializable

Schedule

Icon presentation

43



Exercise

Document a class diagram using the following information:

- A class diagram containing the following classes: Personal Planner Profile, Personal Planner Controller, Customer Profile, and Buyer Record.
- Associations drawn using the following information:
  - Each Personal Planner Profile object can be associated with up to one Personal Planner Controller object.
  - Each Personal Planner Controller object must be related to one Personal Planner Profile.
  - A Personal Planner Controller object can be associated with up to one Buyer Record and Customer Profile object.
  - An instance of the Buyer Record class can be related to zero or one Personal Planner Controller.
  - Zero or one Personal Planner Controller objects are associated with each Customer Profile instance.

44