# Class Diagram

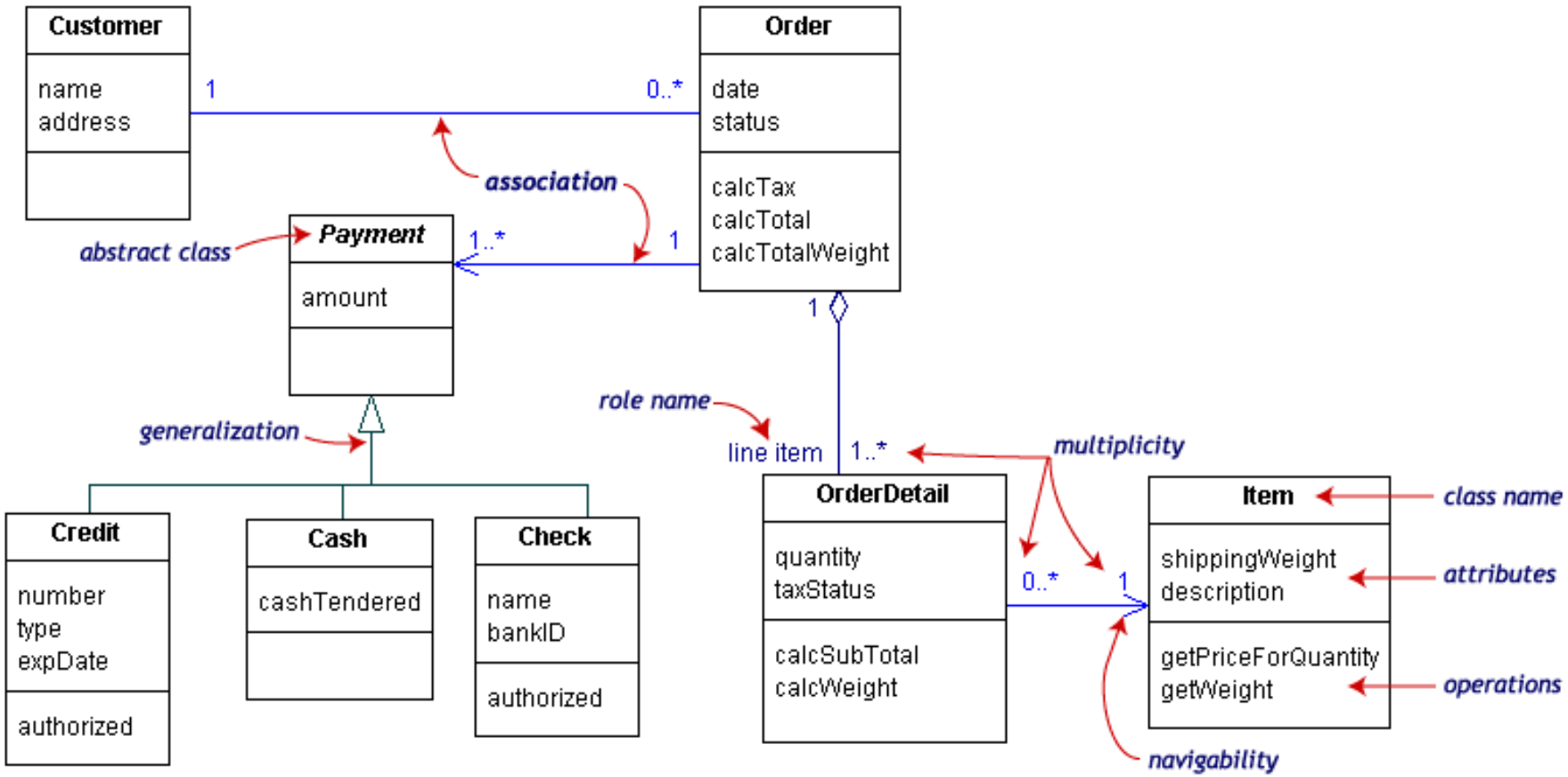
Chonnam National University
School of Electronics and Computer Engineering

Kyungbaek Kim

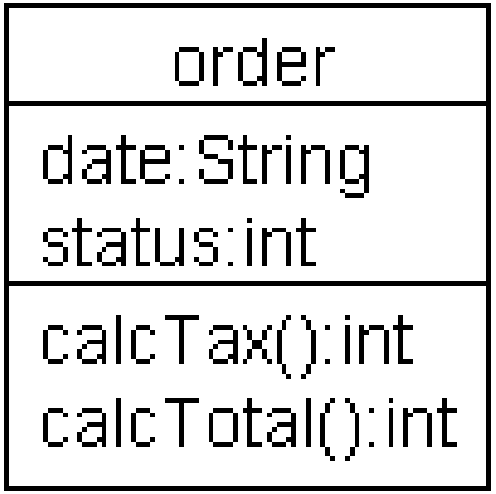
# Class Diagram

- A type of static structure diagram
- Describes the structure of a system
  - Classes
  - Attributes
  - Operations (or methods)
  - Relationships among the classes
- Frequently used by Object-Oriented Design

# Example of Class Diagram



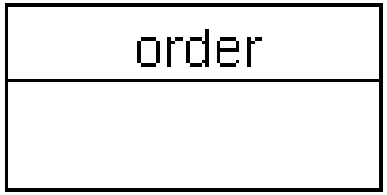
# A simple class



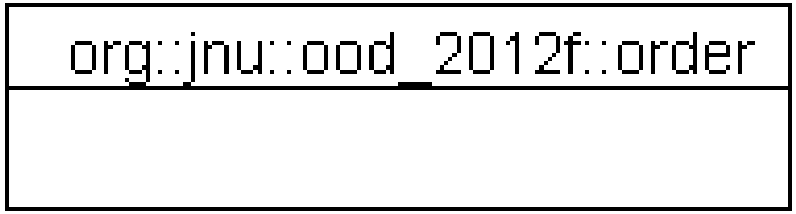

- A class with three sections
  - Upper part
    - The name of class
    - Mandatory
  - Middle part
    - The attributes
    - Optional
  - Lower part
    - The methods or operations
    - Optional

# Class Name

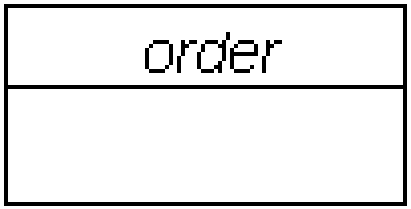
- Every class has an unique name
- Distinct to the other classes
- Simple name → using only class name
- Path name → including package name
- Abstract class → use italic font




Simple Name




Path Name



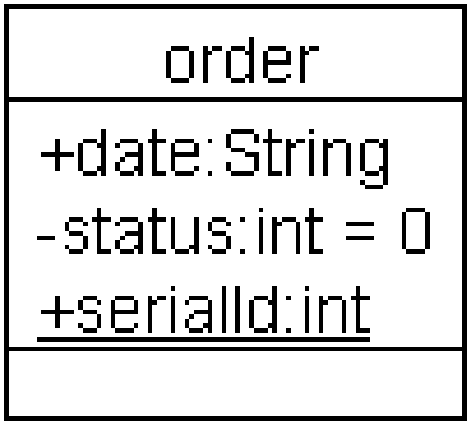

Abstract

# Attribute

- Represented with nouns
- Format

**Visibility Name : Type = Default_Value**

  - Visibility
    - + : public
    - - : private
    - # : protected
    - underline : static

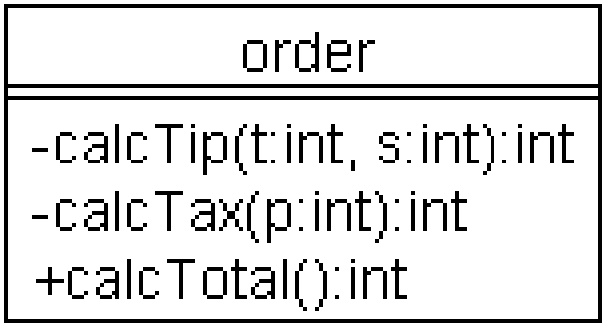


# Operations

- Represented with verbs
- Format

**Visibility Name (Parameter-List) : Return-Type-Expression**

  - Parameter-List
    - Use tuples as (Parameter Name : Parameter Type)



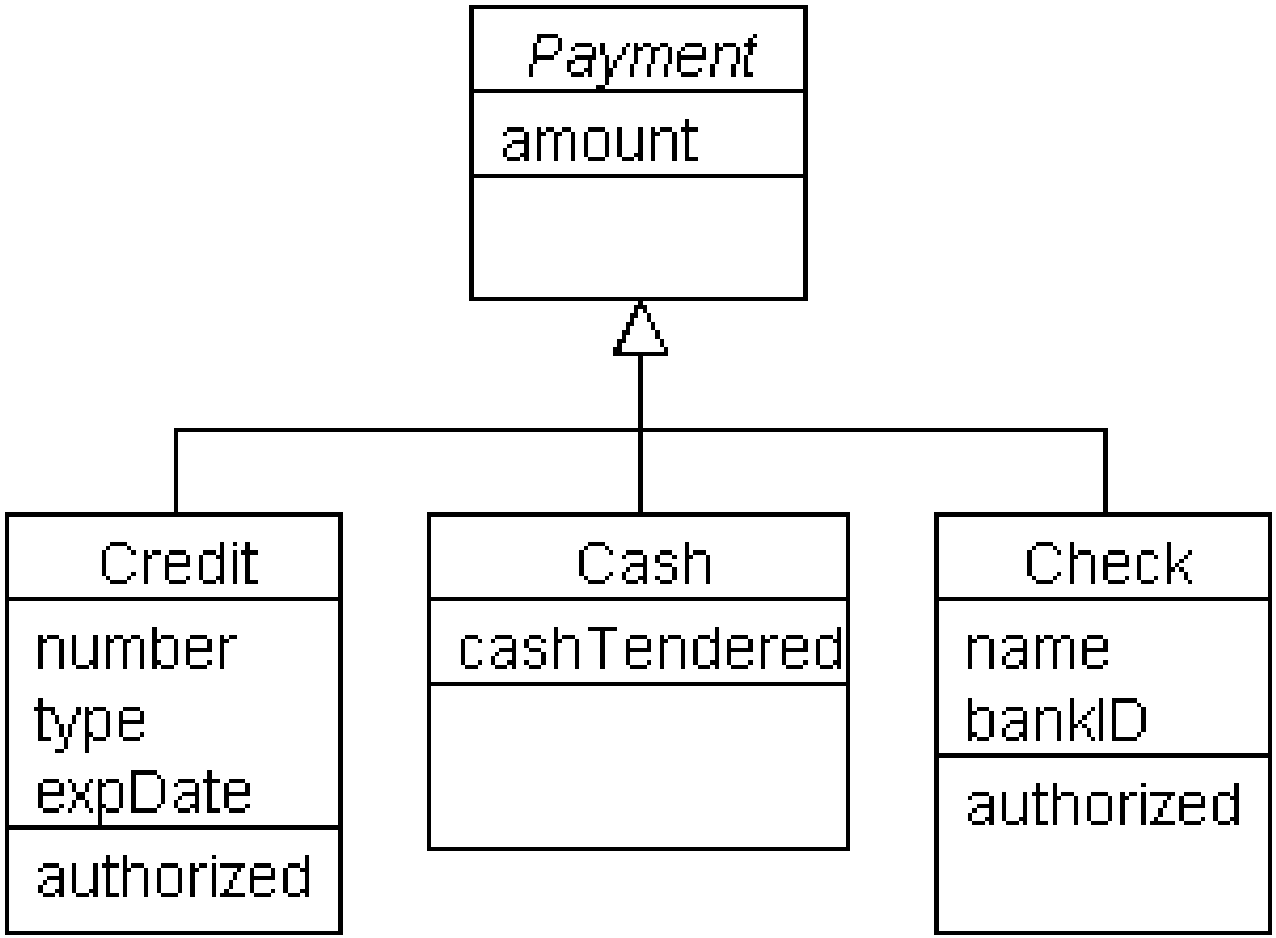
# Relationships

- Logical or physical connections between classes
- Types of relationships
  - Generalization
  - Realization
  - Association
  - Aggregation
  - Composition
  - Dependency

# Generalization

- "is a" relationship
  - e.g., A human is a mammal. A mammal is an animal.
- Two related classes
  - Subclass : a specialized form of superclass
  - Superclass : generalization of subclass
- Inheritance in Object-Oriented Language

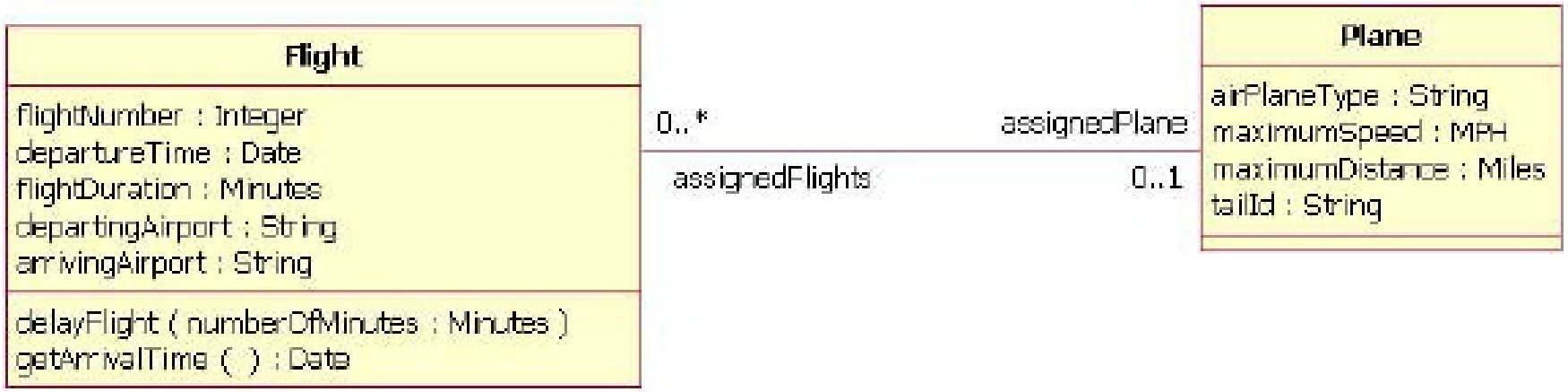
# Drawing of Generalization



# Association

- Represents a family of links
- Relationship between instances
- Binary associations are normally represented as a line
  - An association can be **named**
  - The ends of an association can be annotated with **Role names, Ownership indicators, Multiplicity, Visibility** and others
- Types, in the aspect of *navigability,* that is, the ability of sending a query
  - Bidirectional Association
  - Unidirectional Association

# Bidirectional Association




- Two classes know each other
- In the example
  - A Plane instance can be assigned to 0 or many Flight instances
  - A Flight instance can be assigned to 0 or 1 Plane instance

# Unidirectional Association

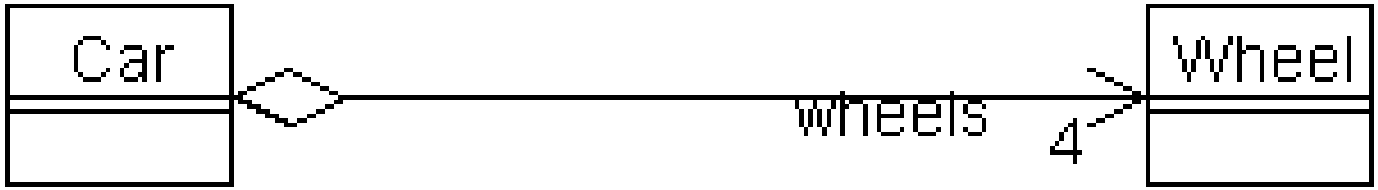

- Only one of two classes knows the relationship
- In the example
  - An OverdrawnAccountReport instance can be assigned to 0 or many BankAccount instances
  - BankAccount instance does not know the relationship

# Multiplicity

- Potential Multiplicity Values
  - 0..1 : Zero or one
  - 1 : Only one
  - 0..* : Zero or many
  - * : Zero or many
  - 1..* : One orr many
  - 3 : Only three
  - 0..4 : Zero to four

# Aggregation

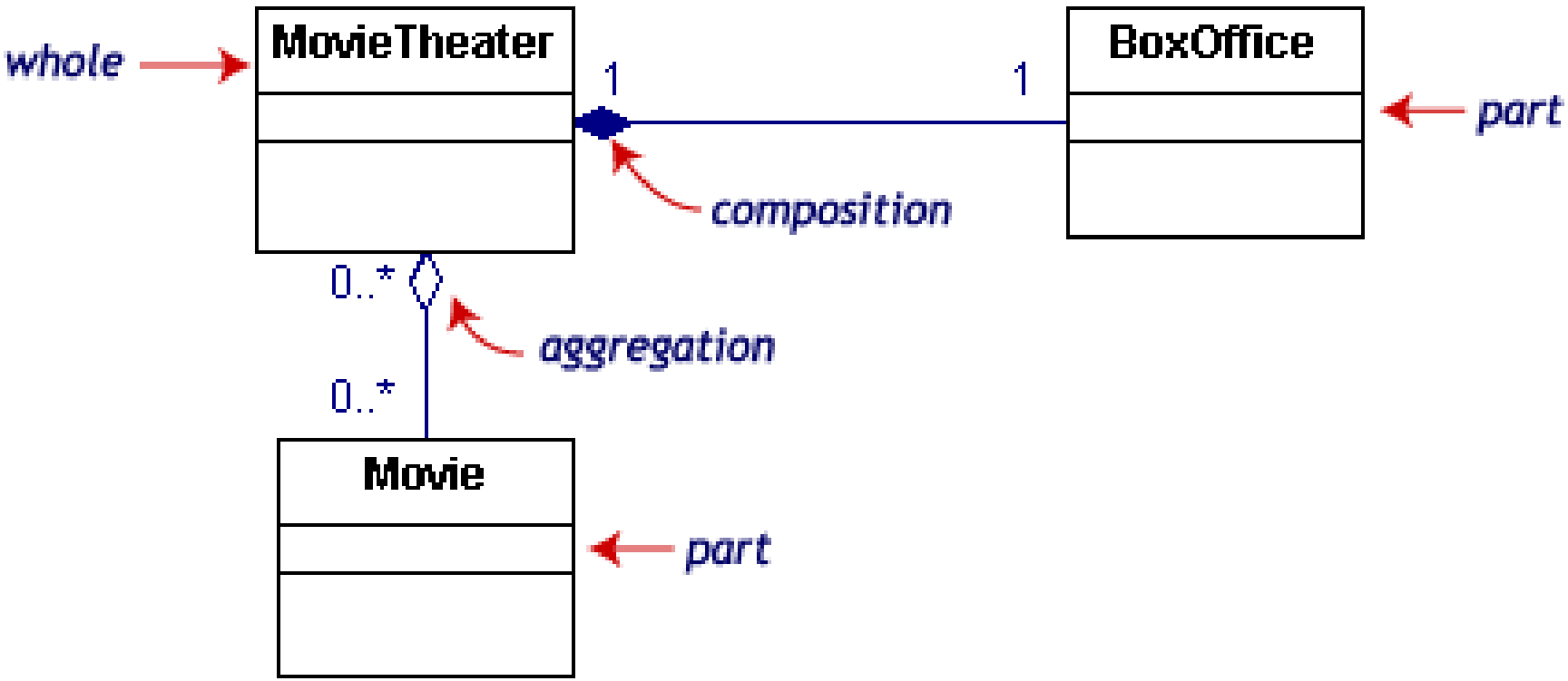
- Relationship between whole and part
- “has a” relationship
  - e.g., a car has four wheels
- Whole and parts are independent to each other
  - Have different lifetime



# Composition

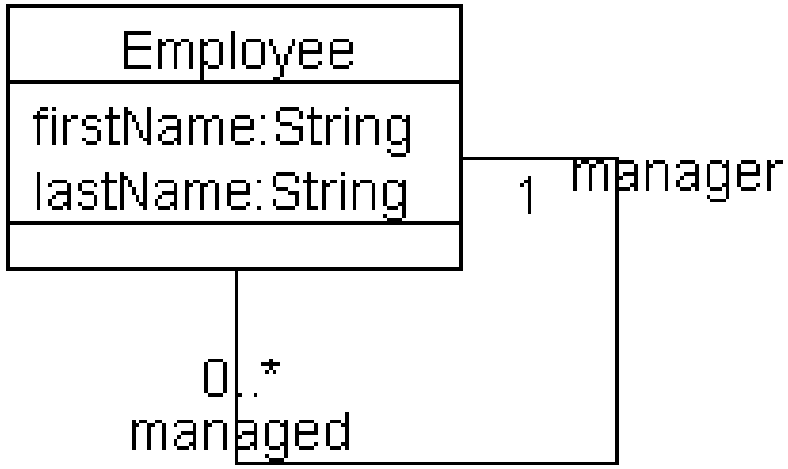
- Same concept to Aggregation
- Except one thing
  - Whole and parts are dependent to each other
  - Have the same lifetime

# Example of Aggregation and Composition



# Reflexive association

- One class can be associated with itself

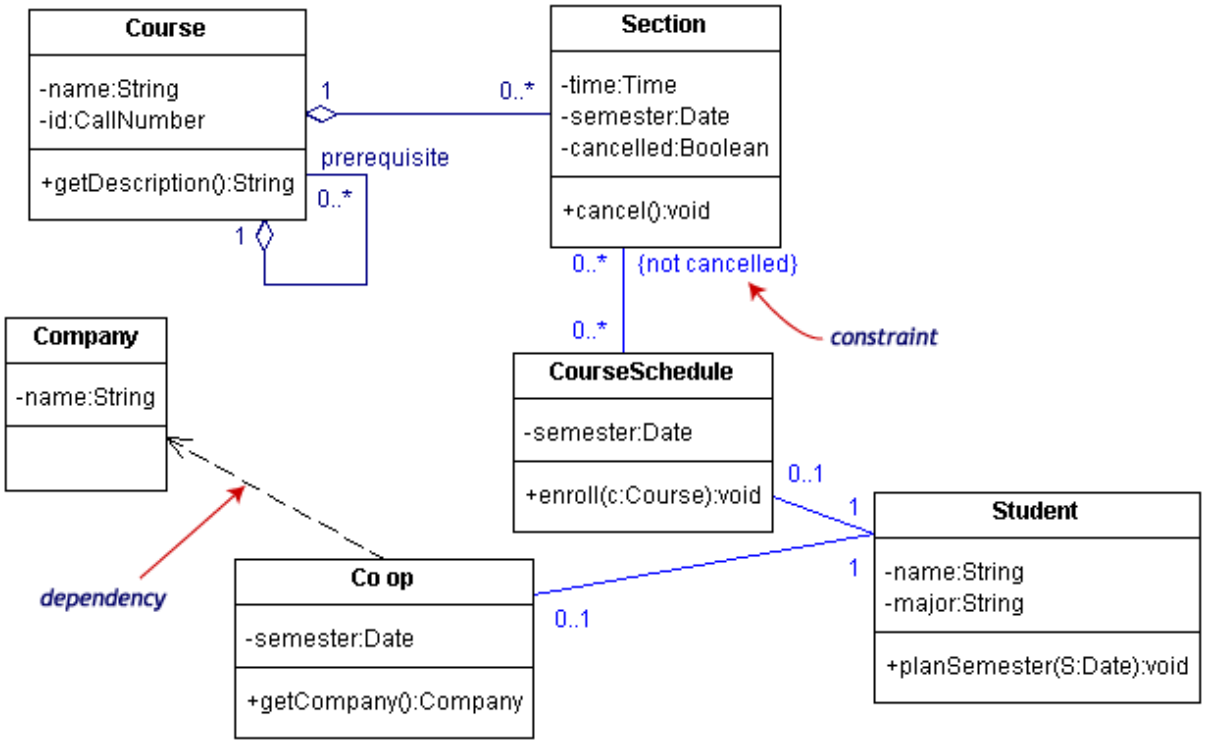


# Dependency

- Weaker form of relationship
- Indicates that one class depends on another
- "using" relationship
  - B is used for a method parameter of A
  - B is used for a local parameter of A

# Constraint

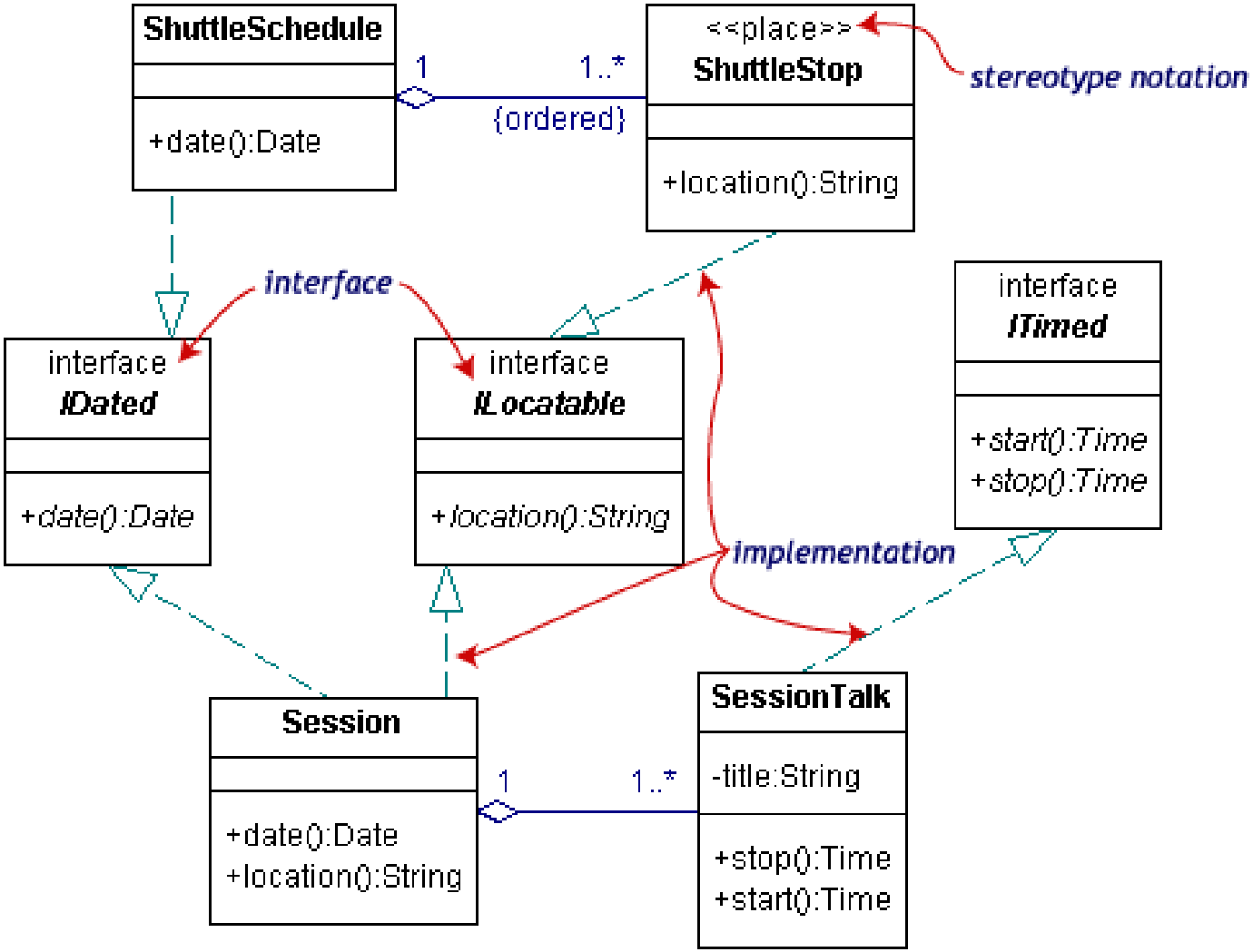
- Indicate the implementation condition
- Used with " { } "



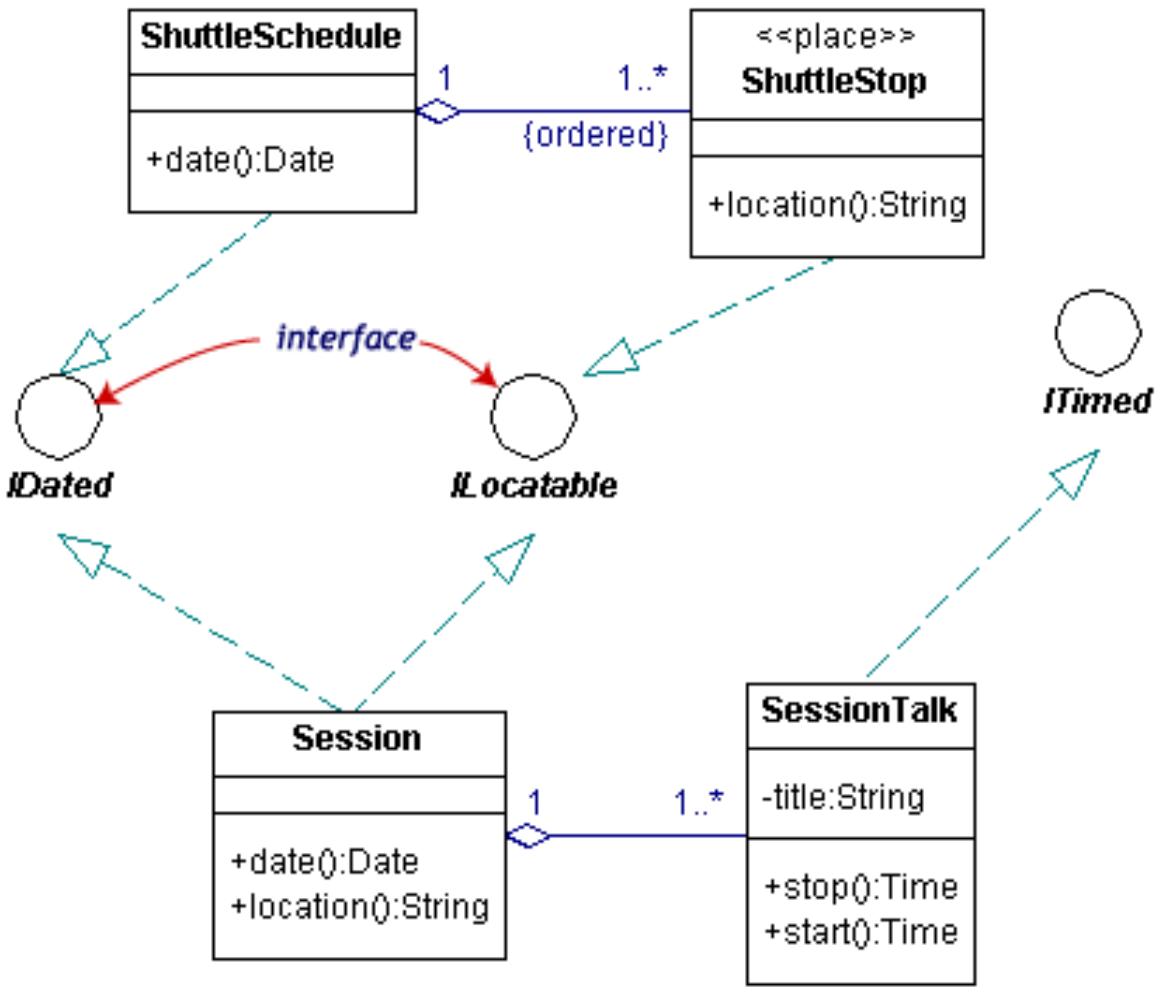
# Realization

- Implement relationship
- Two model elements
  - Client : realizes (implements or executes) the behavior of a model element
  - Supplier : specifies the behavior of a model element
- Interface in Object Oriented Language
  - Allow loose coupling between components
  - Provide better flexibility to softwares

# Drawing of Realization

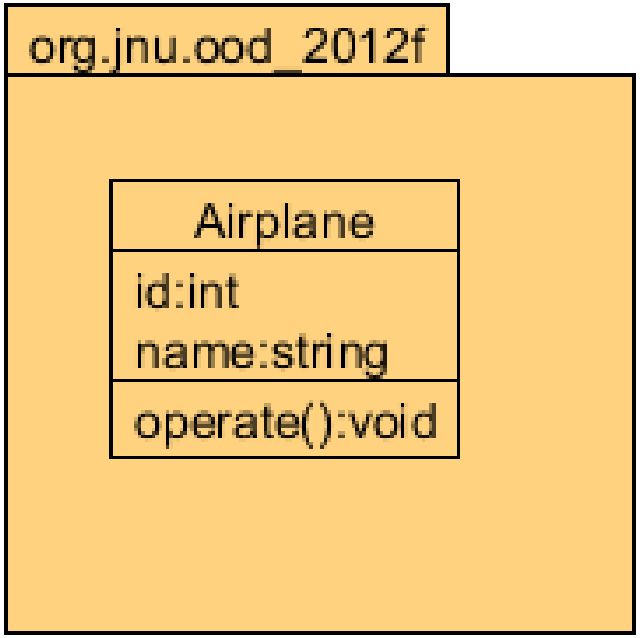


# Circle Representation of Realization

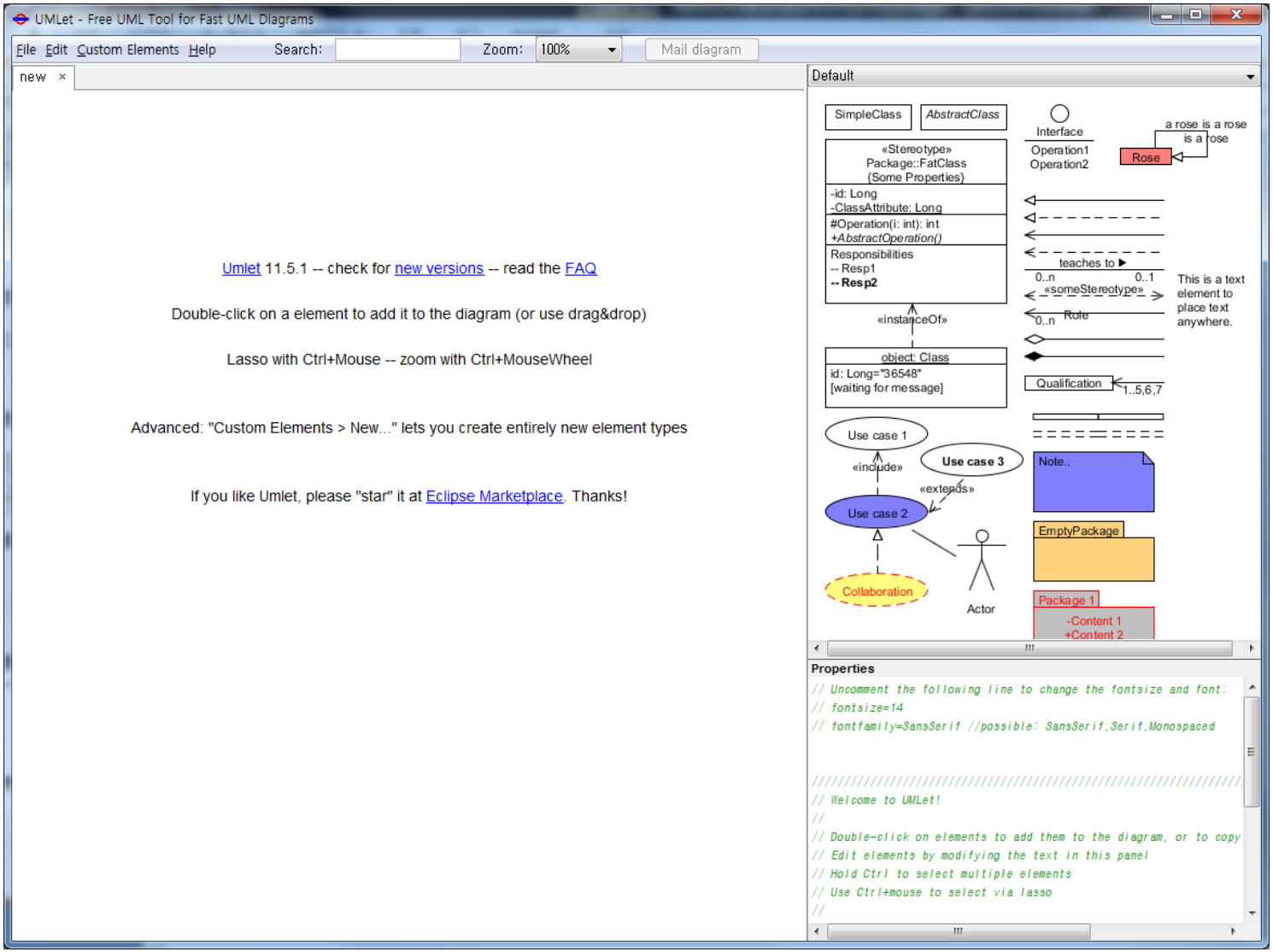


# packages

- Class diagram may include the packages
- Each package has the distinct name space.



# UMLet


UMLet - Free UML Tool for Fast UML Diagrams
File Edit Custom Elements Help
Search:
Zoom: 100%
Mail diagram
new
Umlet 11.5.1 -- check for new versions -- read the FAQ
Double-click on a element to add it to the diagram (or use drag&drop)
Lasso with Ctrl+Mouse -- zoom with Ctrl+MouseWheel
Advanced: "Custom Elements > New..." lets you create entirely new element types
If you like Umlet, please "star" it at Eclipse Marketplace. Thanks!
Default
SimpleClass
AbstractClass
Interface
Operation1
Operation2
a rose is a rose is a rose
Rose
«Stereotype»
Package::FatClass
{Some Properties}
-id: Long
-ClassAttribute: Long
#Operation(i: int): int
+AbstractOperation()
Responsibilities
-- Resp1
-- Resp2
teaches to
0..n
0..1
«someStereotype»
Role
This is a text element to place text anywhere.
«instanceOf»
object: Class
id: Long="36548"
[waiting for message]
Qualification
1..5,6,7
Use case 1
«include»
Use case 3
«extends»
Use case 2
Note..
EmptyPackage
Collaboration
Actor
Package 1
-Content 1
+Content 2
Properties
// Uncomment the following line to change the fontsize and font:
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospaced
//////////////////////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or to copy
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//