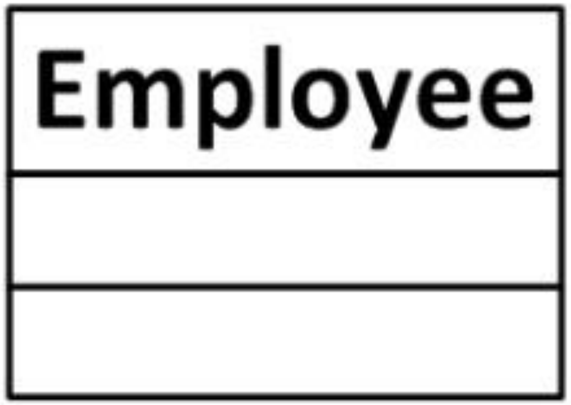
Стандарт UML: діаграма класів

Systems Analysis and Design

Basic OO Concepts (Cont.)

- Abstractions can be represented in OO**P** as **classes**.
- **Class** is an abstraction in that it
  - Emphasis relevant characteristics.
  - Suppresses other characteristics.

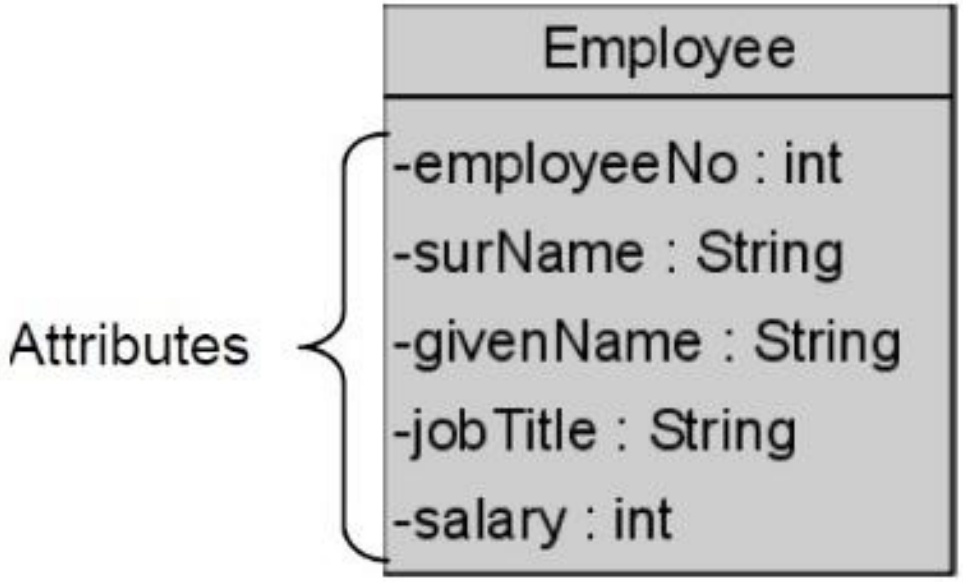


Basic OO Concepts (Cont.)

- Classes are more than representation for abstractions
  - A class is a description of a set of objects that share
    - a common structure
    - a common behavior
  - A class serves as a template for creating objects.
  - The objects created from a class are called the instances of the class.
  - The class is an encapsulation of;
    - A set of attributes (the objects structure description)
    - A set of operations (the objects behavior description)
- The class is the static description; the object is a run time instance of that class.

Basic OO Concepts (Cont.)

- An **Attribute** describes the range of values that instances of a **property** may hold.
- An attribute has a type that defines the type of its instances



Basic OO Concepts (Cont.)

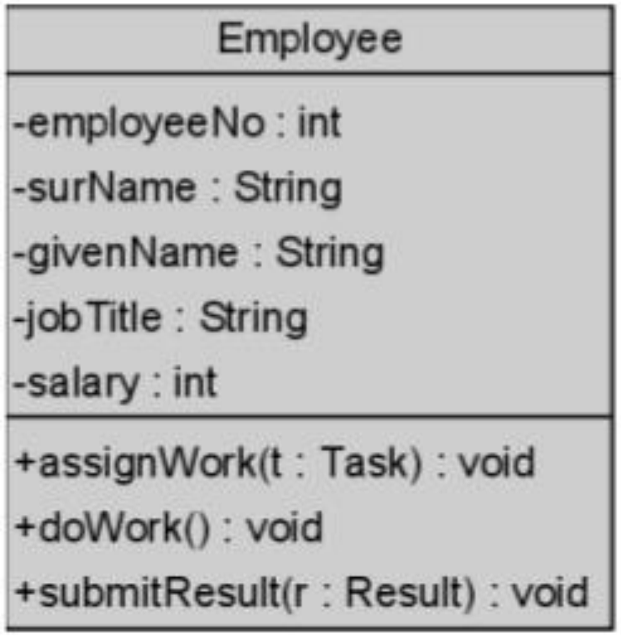
- **Operation** is the description of a service that can be requested from any object of the class to affect **behavior**.
- An operation has s signature, which restrict the possible parameters.

Basic OO Concepts (Cont.)

- An operation can be:
  - **Question/Selector**: accesses the state of an object but does not alter the state.
  - **Modifier/Command:** alters the state of an object.
  - **Iterator**: permits all parts of an object to be accessed in some well-defined order.
  - **Constructor**: creates and/or initializes its state.
  - **Destructor**: frees the object status and/or destroys the object itself.

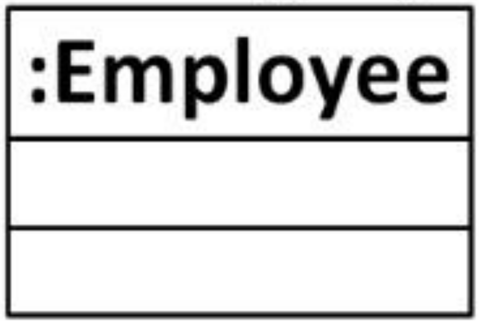
Basic OO Concepts (Cont.)

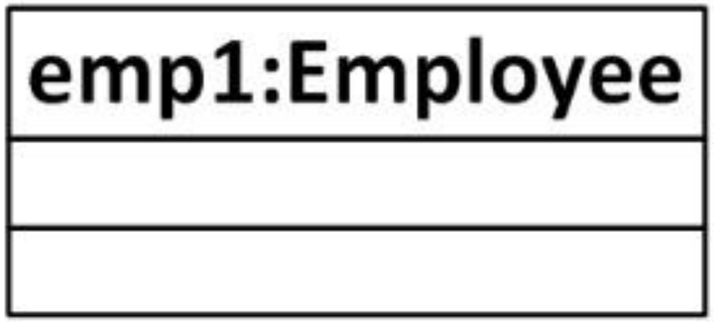
- Operation Example:

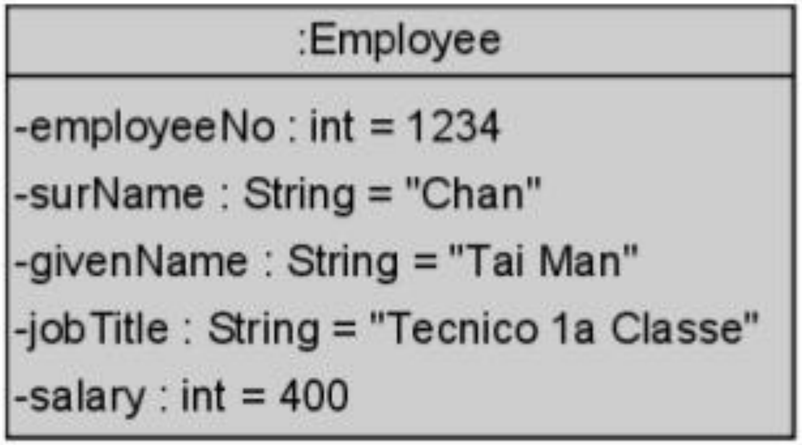


Basic OO Concepts (Cont.)

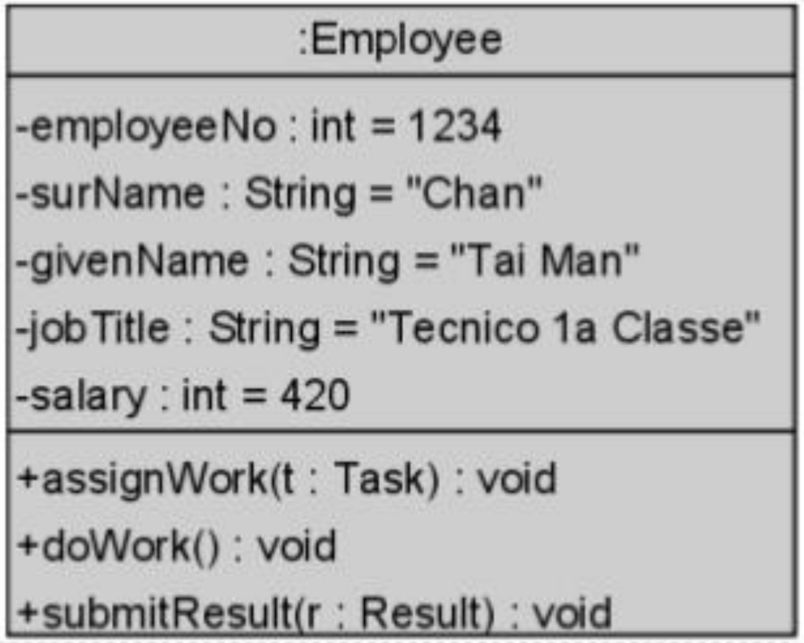
- Modeling Objects in UML






:Employee

-employeeNo : int = 1234
-surName : String = "Chan"
-givenName : String = "Tai Man"
-jobTitle : String = "Tecnico 1a Classe"
-salary : int = 400


:Employee

-employeeNo : int = 1234
-surName : String = "Chan"
-givenName : String = "Tai Man"
-jobTitle : String = "Tecnico 1a Classe"
-salary : int = 420

+assignWork(t : Task) : void
+doWork() : void
+submitResult(r : Result) : void

Basic OO Concepts (Cont.)

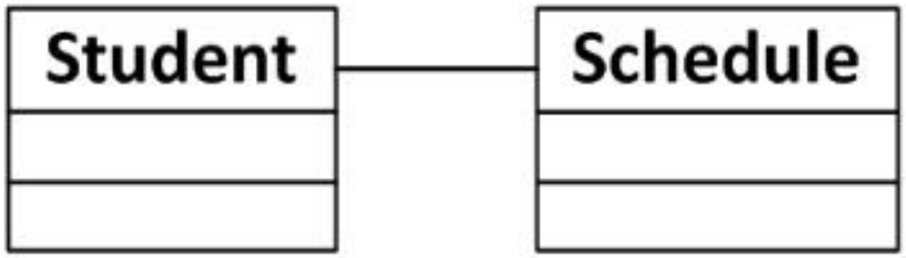
- Relationships between Classes:
  - Association (General relationship)
    - Aggregation (Special Type of Association)
    - Composition (Special Type of Association)
  - Dependency (General relationship)
    - Usage (Special Type of Dependency)
    - Generalization (Special Type of Dependency)
    - Realization (Special Type of Dependency)

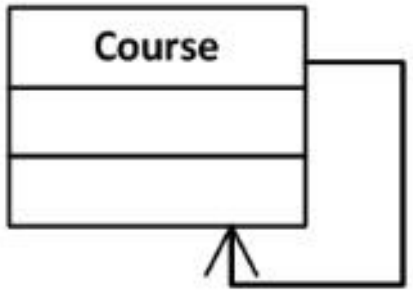
Basic OO Concepts (Cont.)

- **Association**: is a structural relationship, specifying that objects of a class are linked to other objects.
- Structural Relationship means that the relationship is part of the object structure.
- **Link** is an instance of association.
- In Association, the messages may flow in either direction or in both directions across a link.
- Bi-directional associations (between classes) need two links (between the objects) to represent the association.

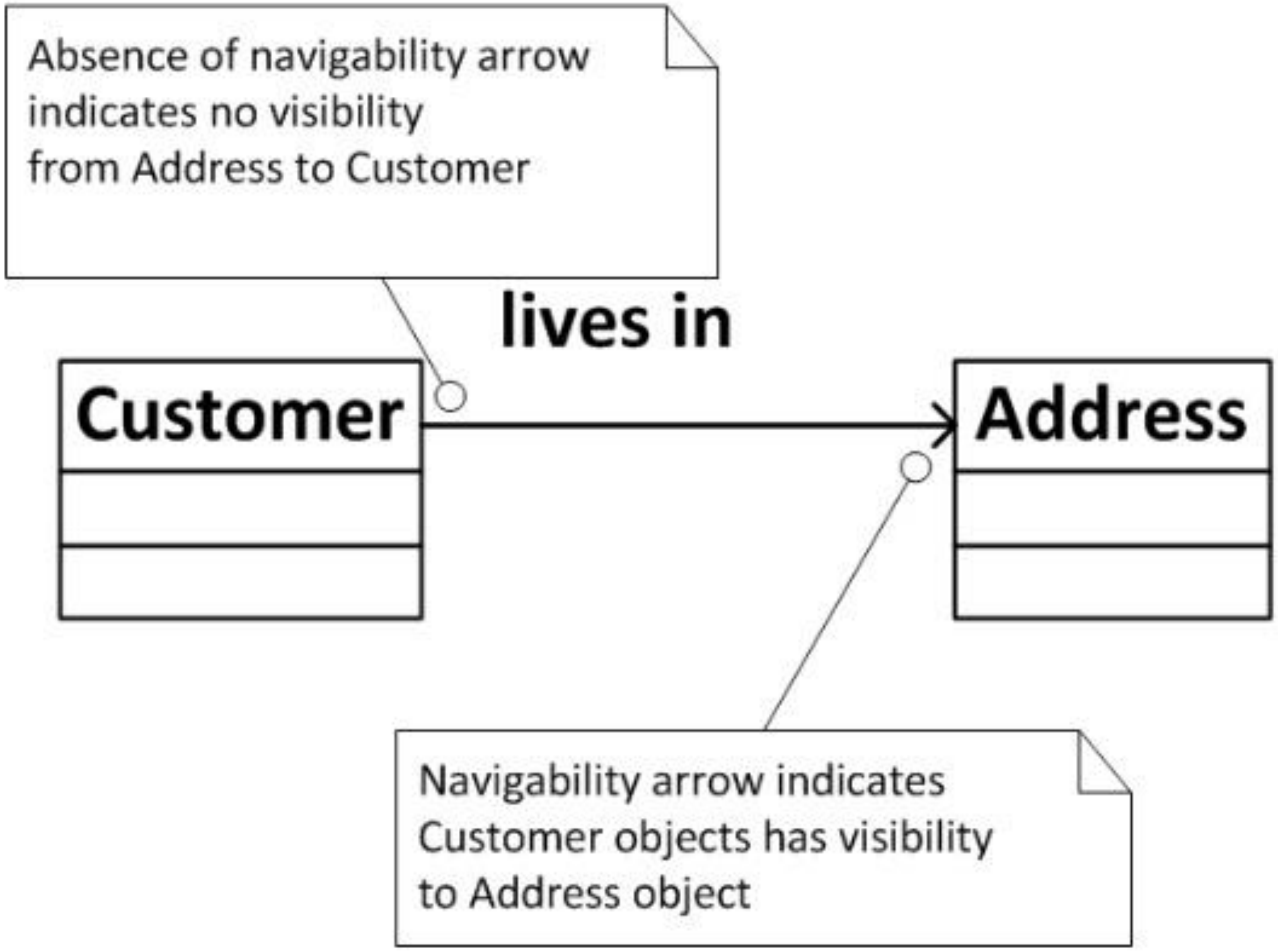
Basic OO Concepts (Cont.)

- Most Associations are <u>binary associations</u> (between two classes)
- <u>Unary association</u>: when a class has an association to itself.
- Unary association means that one instance of a class has <u>links</u> to another instance of the same class.





Basic OO Concepts (Cont.)

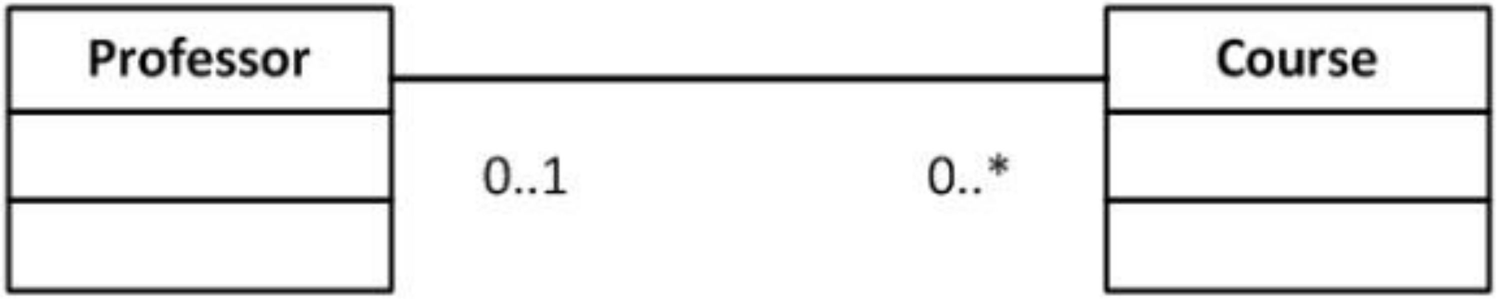

Absence of navigability arrow indicates no visibility from Address to Customer
lives in
Customer
Address
Navigability arrow indicates Customer objects has visibility to Address object

Basic OO Concepts (Cont.)

- **Association Multiplicity**:
  - Multiplicity is the number of instances of a 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.

Basic OO Concepts (Cont.)

- In below:
  - For each instance of professor, many Course instances may be taught.
  - For each instance of Course, there may be either one or zero Professor instance as the instructor.



Basic OO Concepts (Cont.)

- Multiplicity Indicators:

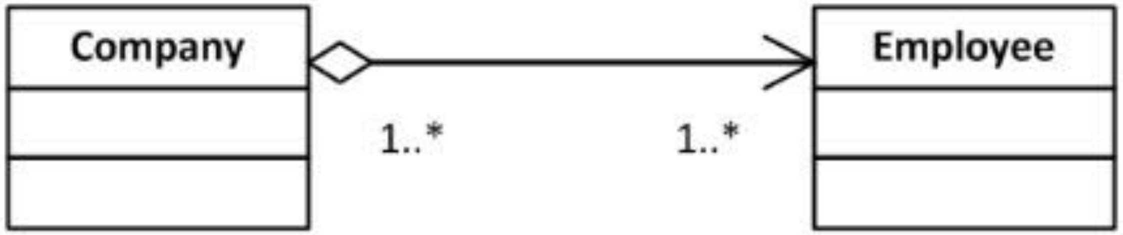
Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or one (optional)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6
Multiple	2, 4, 6

Basic OO Concepts (Cont.)

- **Aggregation**:
  - Is a special type of the association
  - is the same as the association with the exception that the aggregation can't be bi-directional.
  - Aggregation is a typical whole/part relationship.
  - Aggregation is a weak form of association where the part is independent of the whole (aggregate)
    - The same instance of part could be included in several whole instances (shared parts)
    - If the whole instance is destroyed, the part instances may still exist

Basic OO Concepts (Cont.)

- Aggregation Example - In below:
  - One instance of Company can have one or more instances of Employee.
  - One instance of Employee can be working in one or more companies at the same time.

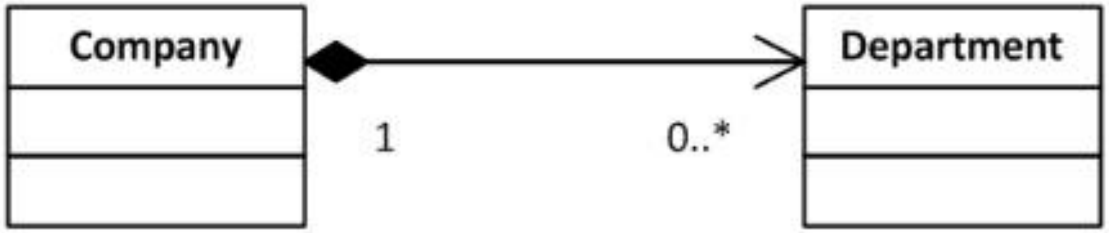


Basic OO Concepts (Cont.)

- **Composition**:
  - Is a special type of the association
  - is exactly the same as the aggregation with the exception that the life time of the part is controlled by the whole.
  - Composition is a strong form of aggregation.
    - The instance of part can be included in one and only one instance of the whole.
    - If the whole instance is deleted, then the part instances must be deleted too.

Basic OO Concepts (Cont.)

- Composition Example - In below:
  - One instance of Company may contain zero or more instances of Departments.
  - One instance of Department can be part of one and only one instance of Company.
  - The Department instance can't live without its company

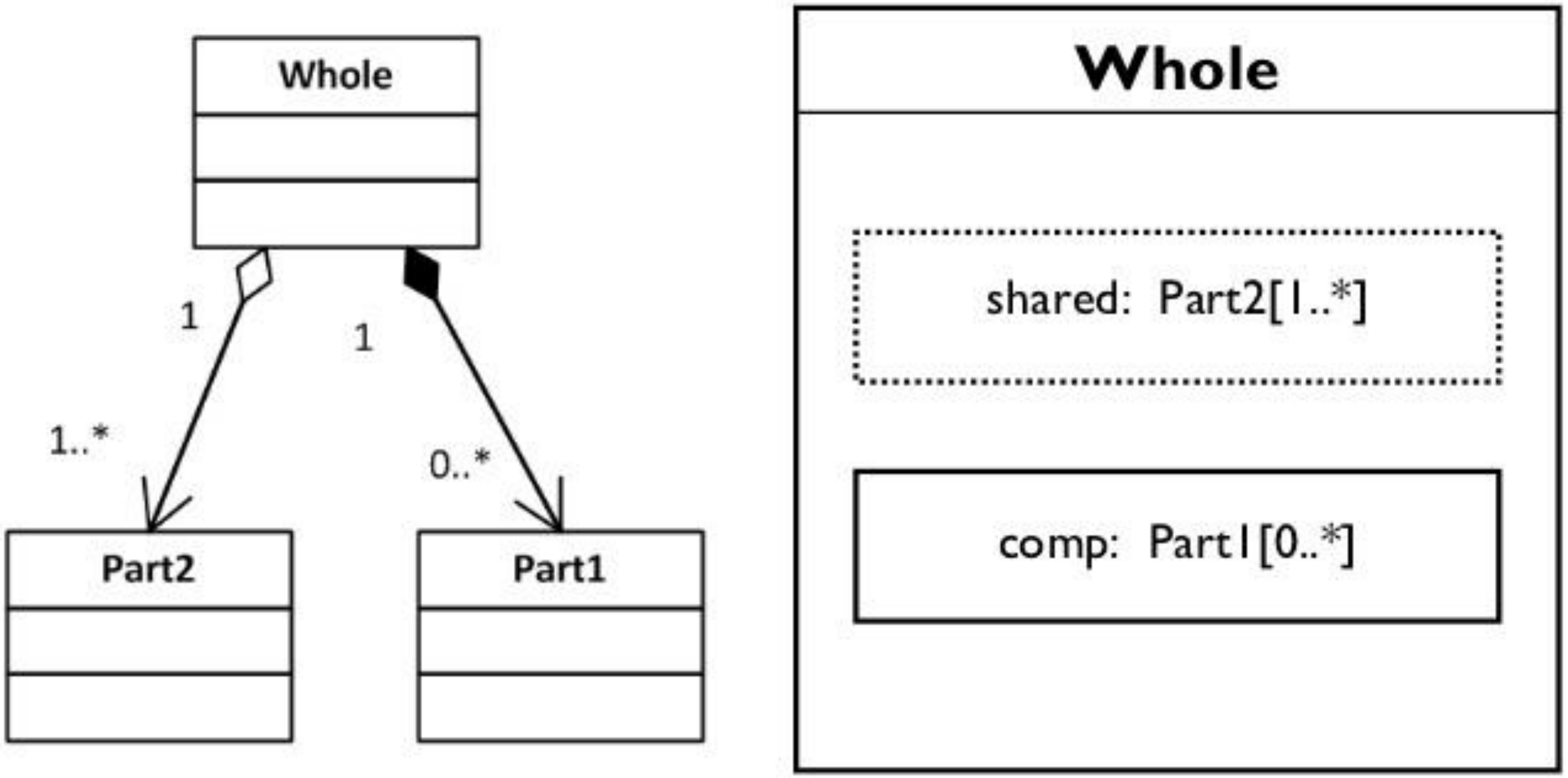


Basic OO Concepts (Cont.)

- UML Structured Class:
  - Because the aggregation and composition are structural relationships we can represent them using UML Structured class.
  - UML Structured class are usually used to describe the internal structure of components and systems.
  - UML Structured class can be used to describe the internal structure of an instance with complex internal structure (complex aggregations and compositions).
  - Structured class contains parts that form the class structure.
  - The parts themselves may also be structured classes.

Basic OO Concepts (Cont.)

Structured Class Example:



Basic OO Concepts (Cont.)

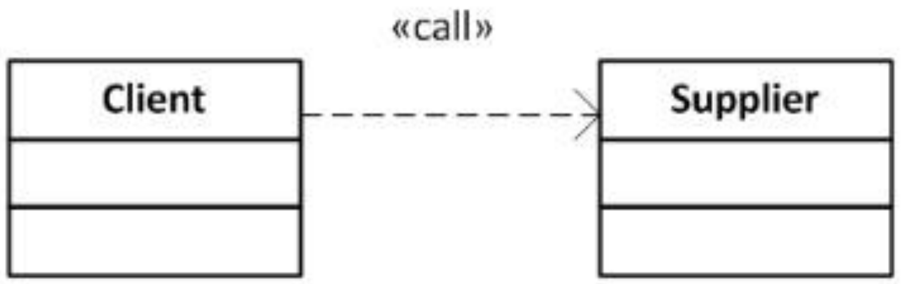
- Dependency Relationship:
  - is a relationship in which changes to one model element impact another model element.
  - Non-structural relationship (that the relationship is not part of the client object structure)
  - Dependency Types:
    - Usage
    - Generalization
    - Realization

Basic OO Concepts (Cont.)

- Usage Dependency:
  - "Usage" usually called dependency which leads to a confusion.
  - "Usage" dependency is a weaker form of relationship.
  - "Usage" dependency shows a relationship between a client/consumer and supplier/provider.
  - The Usage relationship is not part of the client object structure.
  - Indicates that the instance of the client class <u>temporarily</u> uses an instance of the supplier.

Basic OO Concepts (Cont.)

- Usual UML keywords of Usage:
  - «use», «create», «call», «instantiate», «send»

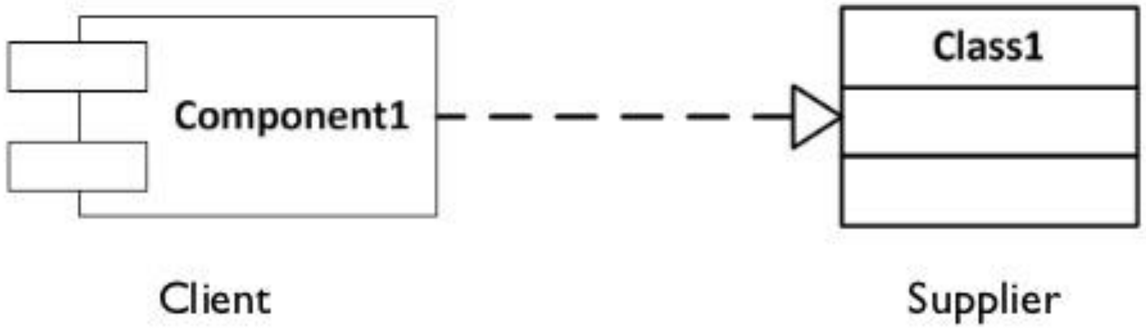


Basic OO Concepts (Cont.)

- Generalization/Specialization
  - Is a relationship among classes where one class shares the structure/behavior of one or more classes.
  - Generalization defines a hierarchy of abstractions in which a subclass inherits from one or more super classes.
  - The objects of the subclasses (specialized classes) are substitutable for objects of the super class (generalized classes).
  - Sometimes called "is a" relationship.
- Inheritance is a mechanism to implement the generalization relationship.

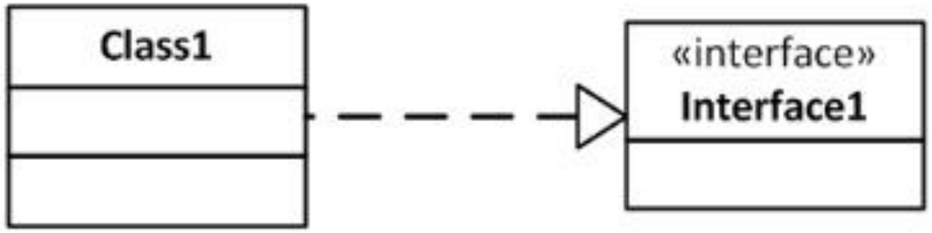
Basic OO Concepts (Cont.)

- Realization
  - One classifier serves as the contract that the other classifier agrees to carry out.
  - One classifier supplies/provides (supplier) the contract and the other realizes it (client).
  - The contract means the set of operations signatures.
  - It is not true generalization, as only the contract is inherited.
  - Several clients can realize the behavior of a single supplier.



Basic OO Concepts (Cont.)

- Implementation (Interface Realization):
  - Is a specialized type of realization relationship between a classifier (client/implementation) and an interface (supplier)
  - The implementation relationship specifies that the realizing classifier must conform to the contract that the interface provides.

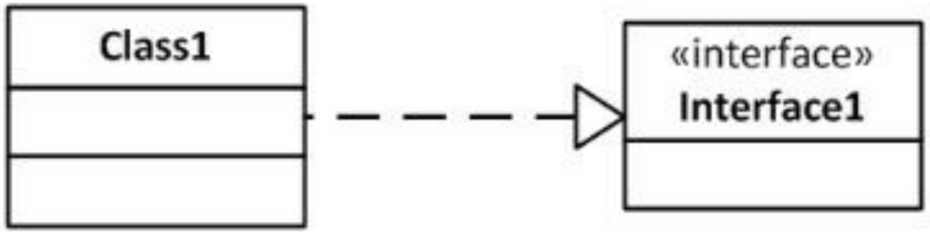


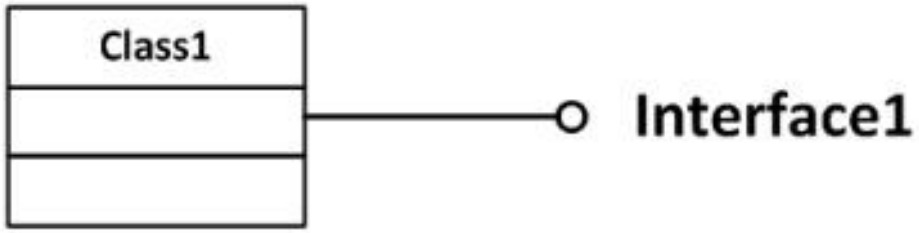
Basic OO Concepts (Cont.)

- Interface:
  - A declaration of a coherent set of public features and obligations
  - An interface specifies a contract between providers and consumers of services.
  - Examples of Interfaces:
    - Provided Interface – describe the set of services the classifier offers to its consumers.
    - Required Interface – describes the set of services the classifier requires from its other classifiers

Basic OO Concepts (Cont.)

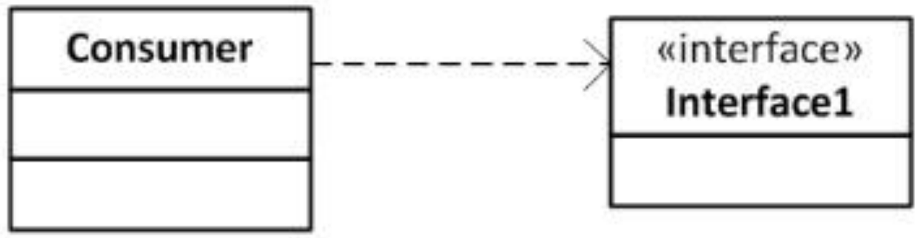
- Provided Interface

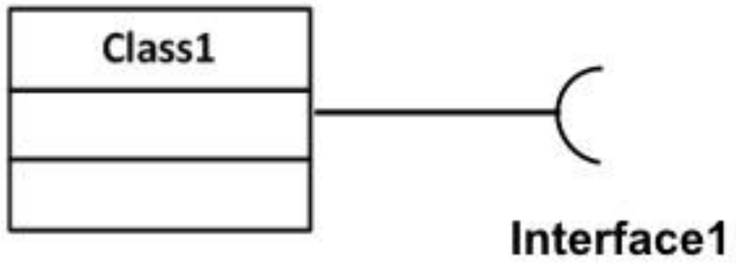




Basic OO Concepts (Cont.)

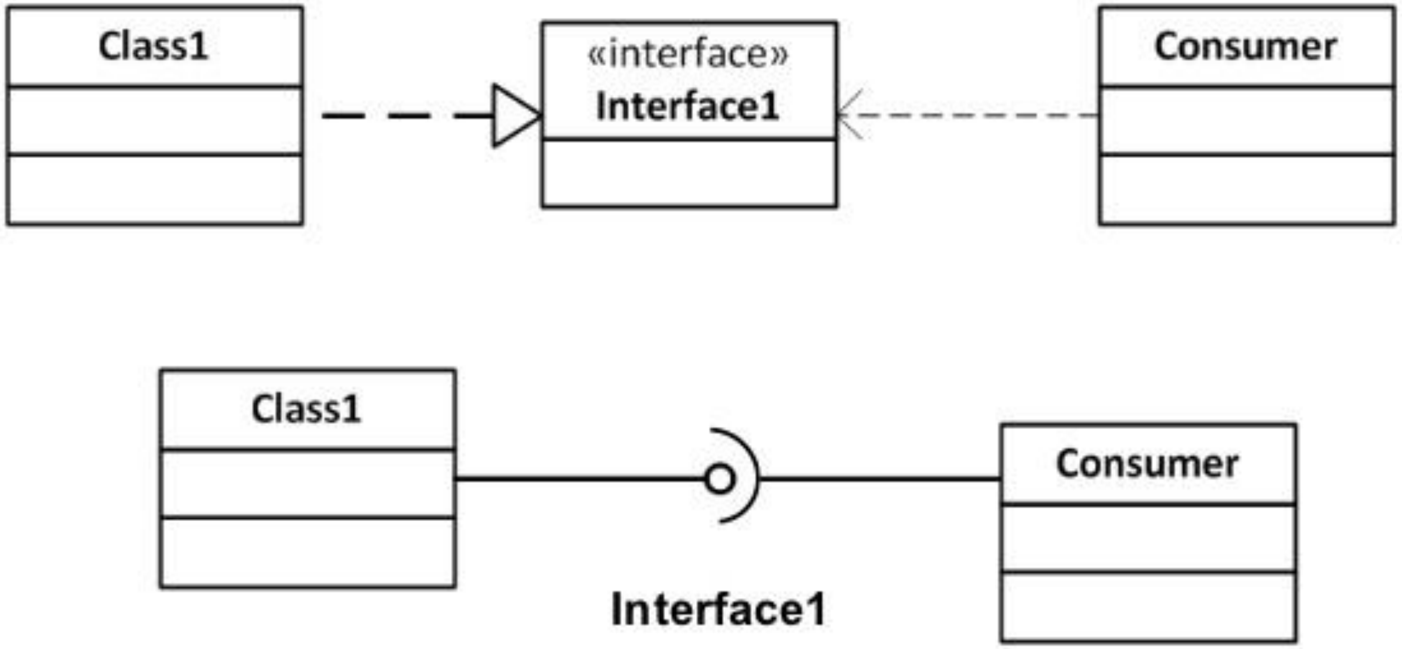
- Required Interface





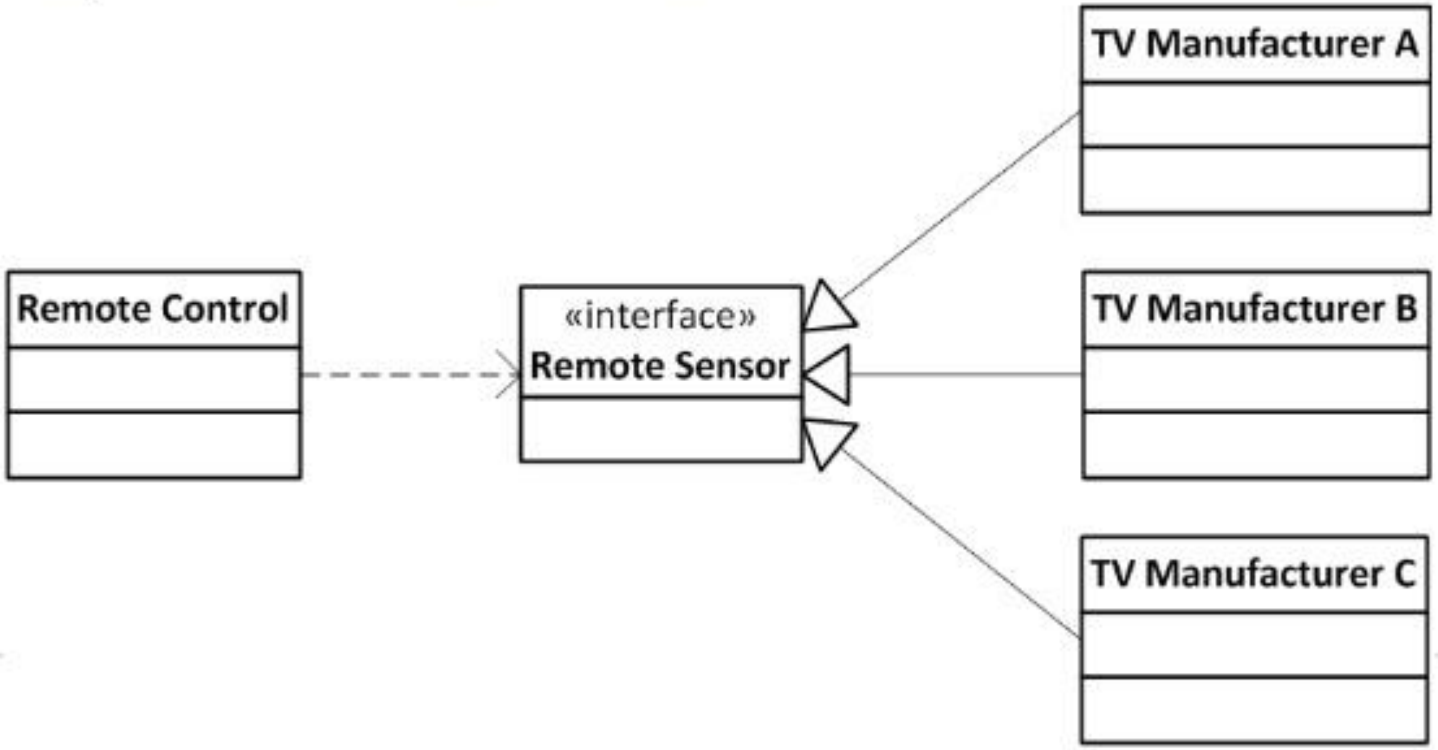
Basic OO Concepts (Cont.)

- Connecting Interfaces



Basic OO Concepts (Cont.)

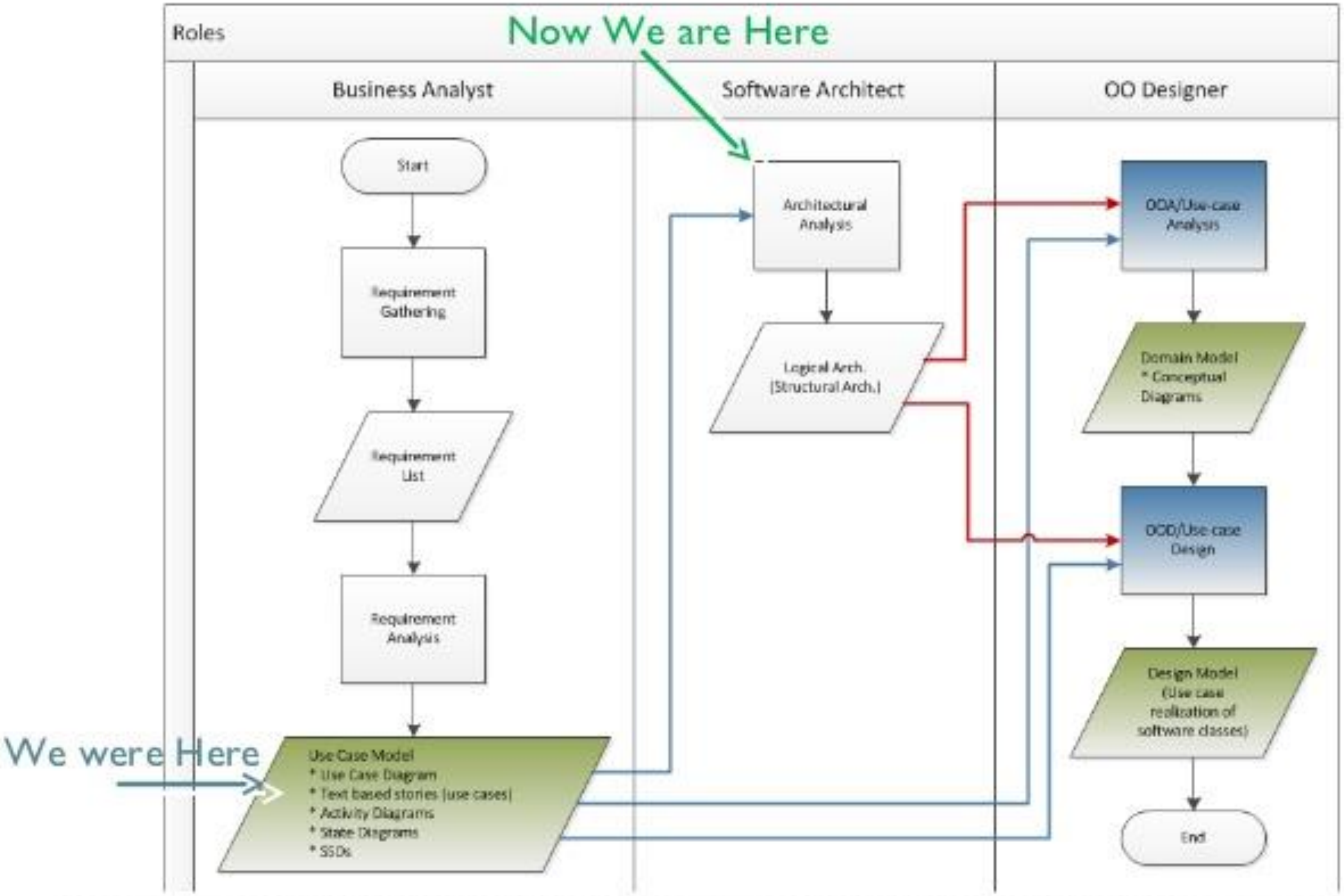
- Polymorphism
  - The ability to hide many different implementations behind a single interface.
  - Every implementation of an interface must fulfill the requirements of that interface.



Basic OO Concepts (Cont.)

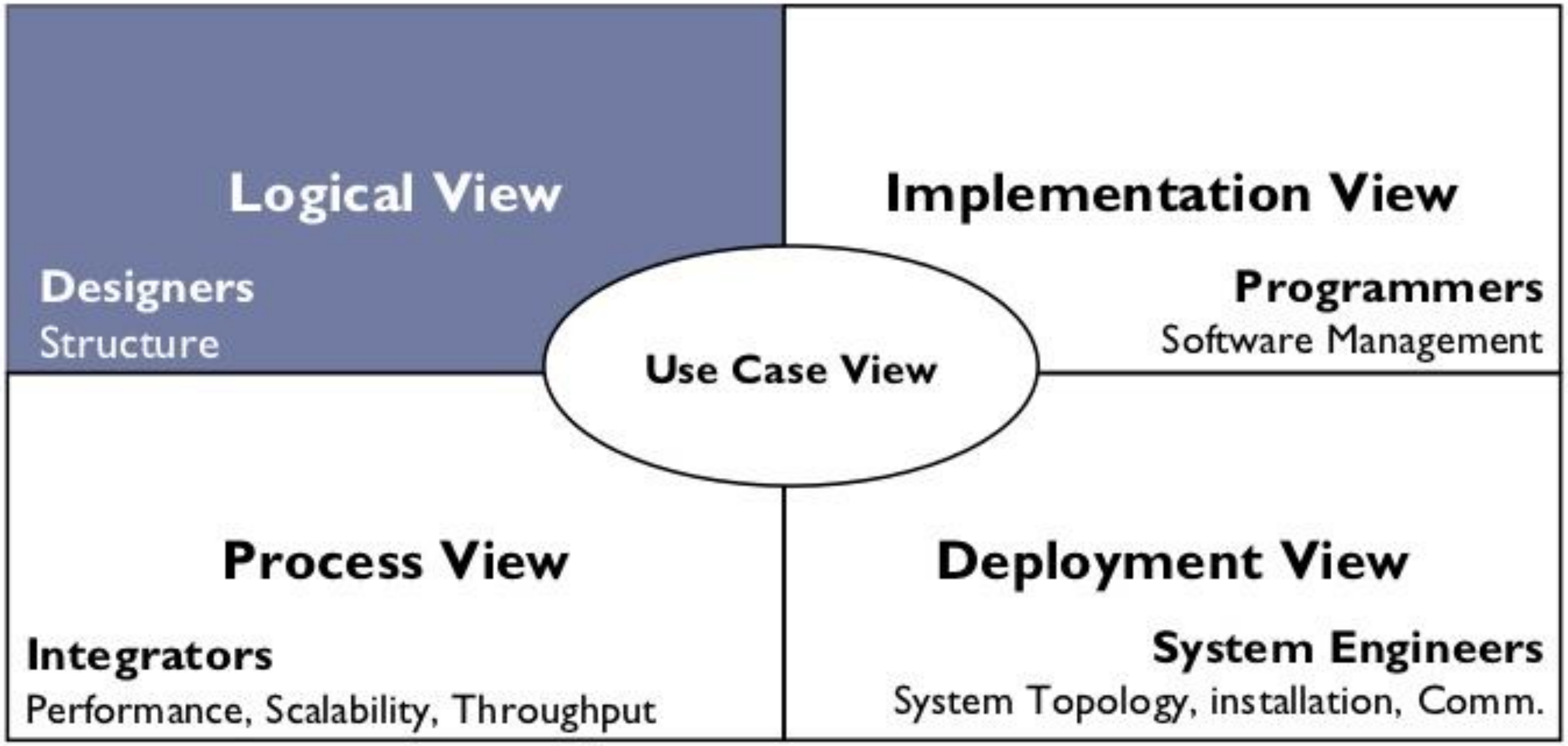
- Relationship Visibility
  - is the ability of one object to see or have reference to another.
  - For an object A to send a message to an object B, B must be visible to A.
  - Object B is visible to object A if B class and A class has any of the following relationships:
    - Association (Aggregation and Composition)
    - Usage Dependency

Architectural Analysis



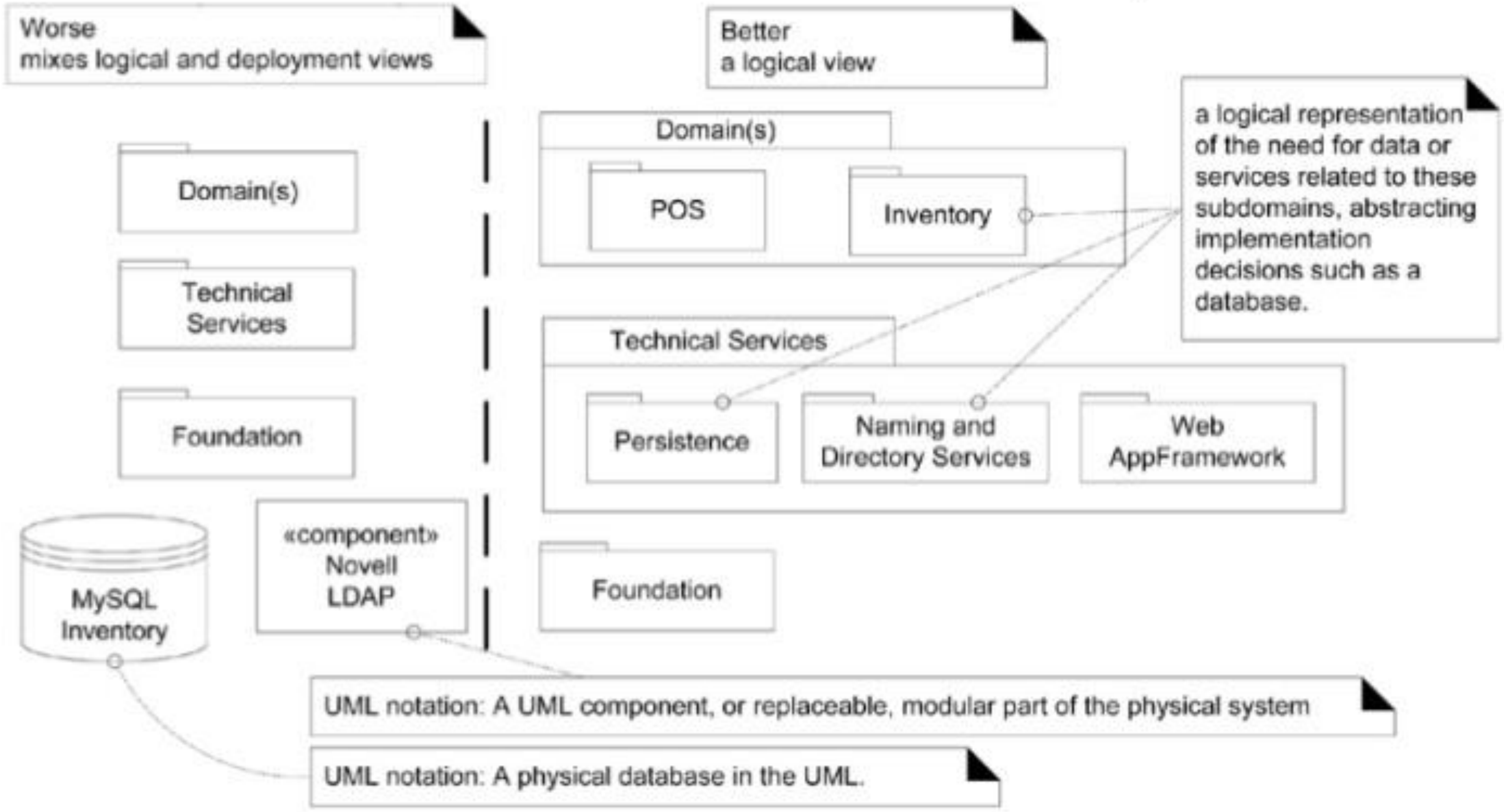
Architectural Analysis (cont.)

- **What is Architecture: The "4+1 View" Model**



Architectural Analysis (cont.)

- Applying UML:
  - Don't show external resources as the bottom layer


Worse
mixes logical and deployment views
Domain(s)
Technical Services
Foundation
MySQL Inventory
«component» Novell LDAP
Better
a logical view
Domain(s)
POS
Inventory
a logical representation of the need for data or services related to these subdomains, abstracting implementation decisions such as a database.
Technical Services
Persistence
Naming and Directory Services
Web AppFramework
Foundation
UML notation: A UML component, or replaceable, modular part of the physical system
UML notation: A physical database in the UML.