MEDI-CAPS
UNIVERSITY

Object Oriented Programming
UNIT – 3

Arpit Deo
Assistant Professor,
Computer Science & Engineering Department,
Medi-Caps University, Indore

Defining Class

- *A CLASS is a template (specification, blueprint)* for a collection of objects that share a common set of attributes and operations.

- *Class* →

Health Club Member
attributes
operations

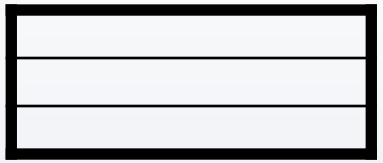
Objects →

What is a Class Diagram?

- A Class Diagram is a diagram describing the structure of a system shows the system's
  - Classes
  - Attributes
  - Operations (or methods),
  - Relationships among the classes.

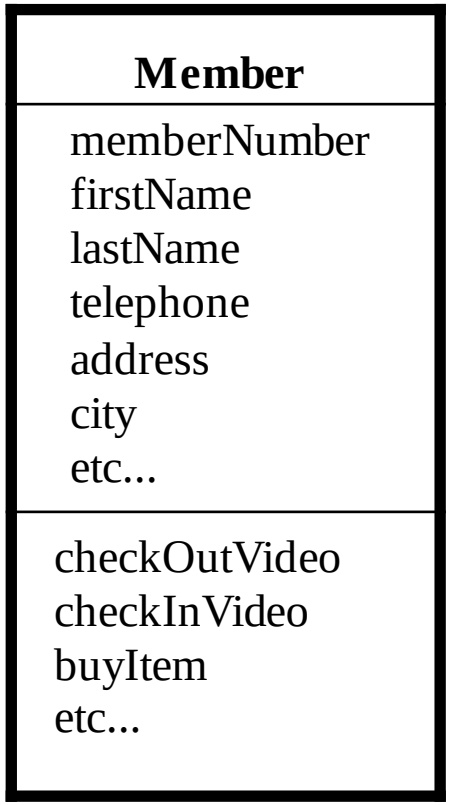
UML Class Diagram Notation

Class



attributes

operations



Expanded view of a Class into its three sections:

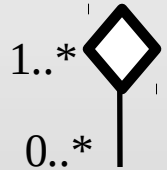
- **Top: Class Name**
- **Middle: attributes**
- **Bottom: operations**

UML Class Diagram Notation

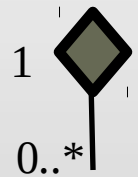
Class
Generalization
Relationship

Object Association

Object Aggregation Association



Object Composition Association



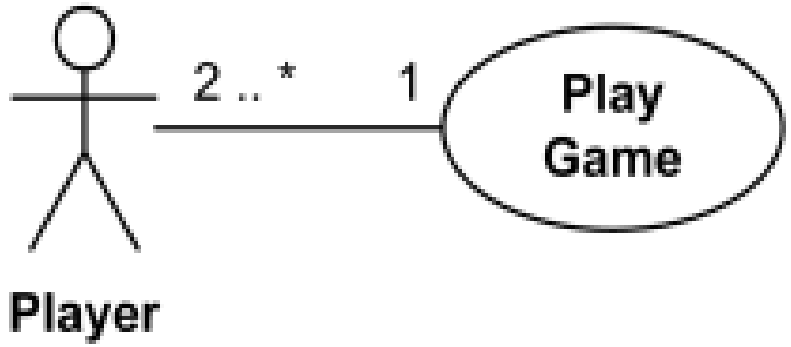
Will always be “1”

Multiplicity

- The number of objects involved in both sides of relationships is called multiplicity of a relationship.
- Multiplicity is the number of objects of a class that participate in the association.

Multiplicity	Option	Cardinality
0..0	**0**	Collection must be empty
0..1		No instances or one instance
1..1	**1**	Exactly one instance
0..*	*****	Zero or more instances
1..*		At least one instance
5..5	**5**	Exactly 5 instances
m..n		At least m but no more than n instances

- Example



There are four types of multiplicities in association:

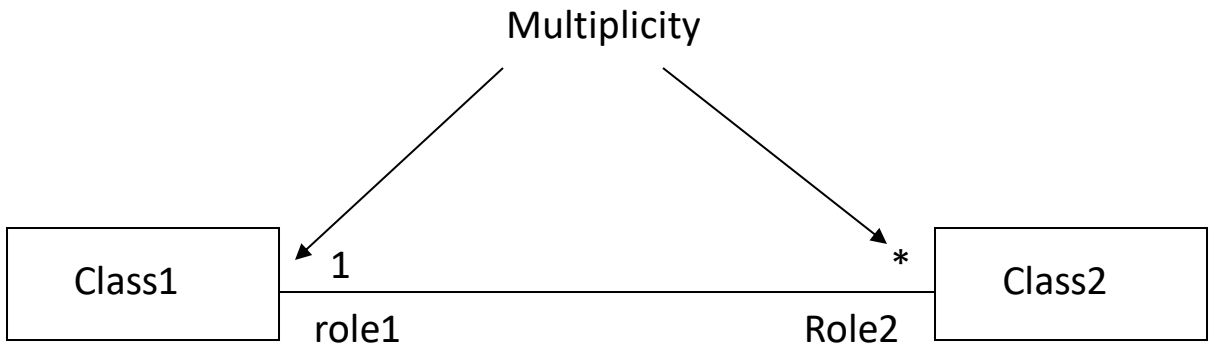
One-to-one: Each class instance is related to a single instance of another class.

One-to-many: A class instance can be related to multiple instances of the other class.

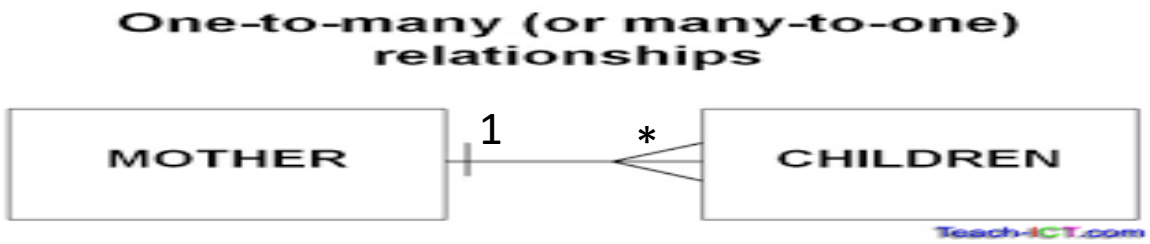
Many-to-one: Multiple instances of a class can be related to a single instance of the other class. This multiplicity is the opposite of a one-to-many relationship.

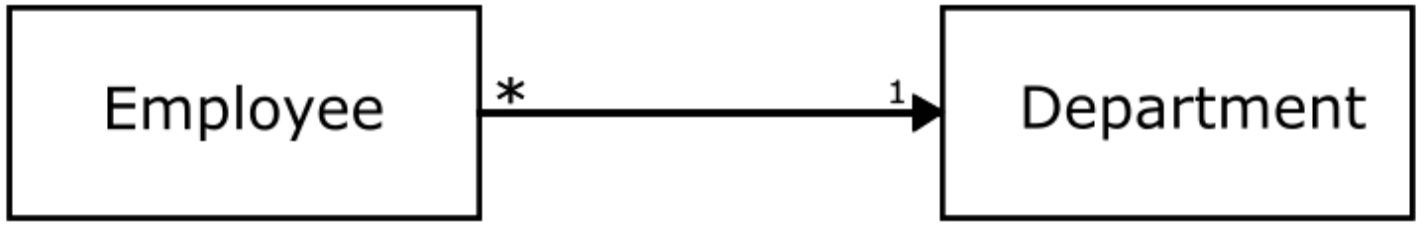
Many-to-many: The class instances can be related to multiple instances of each other.

Generalized Example of Multiplicity:

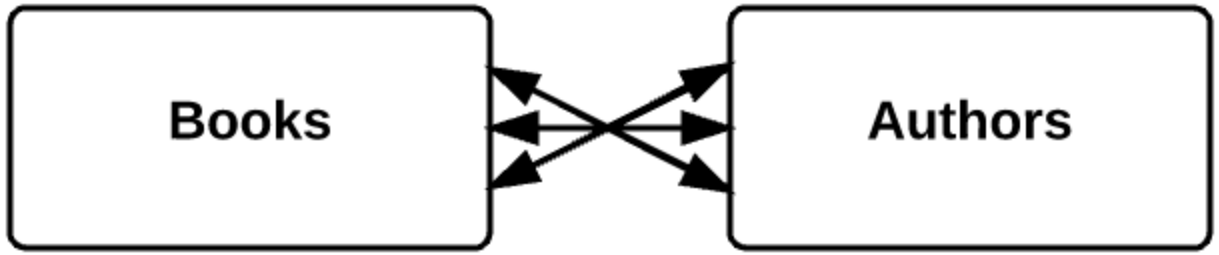


One-to-one:

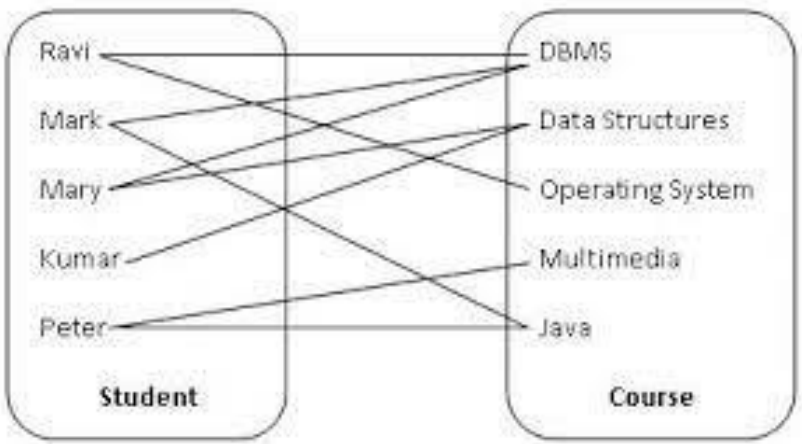




Domain model diagram of Many-To-One association


Books
Authors

Many to Many

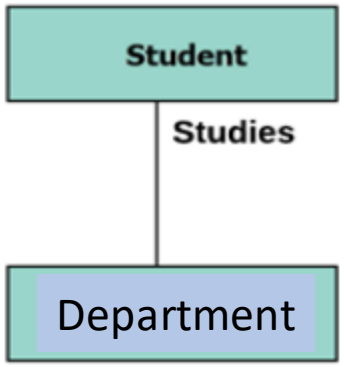

Ravi
Mark
Mary
Kumar
Peter
Student
DBMS
Data Structures
Operating System
Multimedia
Java
Course

Association

- Association is a simple structural connection or channel between classes and is a relationship where all objects have their own lifecycle and there is no owner.
- Association is relation between two separate classes which establishes through their Objects.
- In Object-Oriented programming, an object communicates to other object to use functionality and services provided by that object.
- Association can be one-to-one, one-to-many, many-to-one, many-to-many.

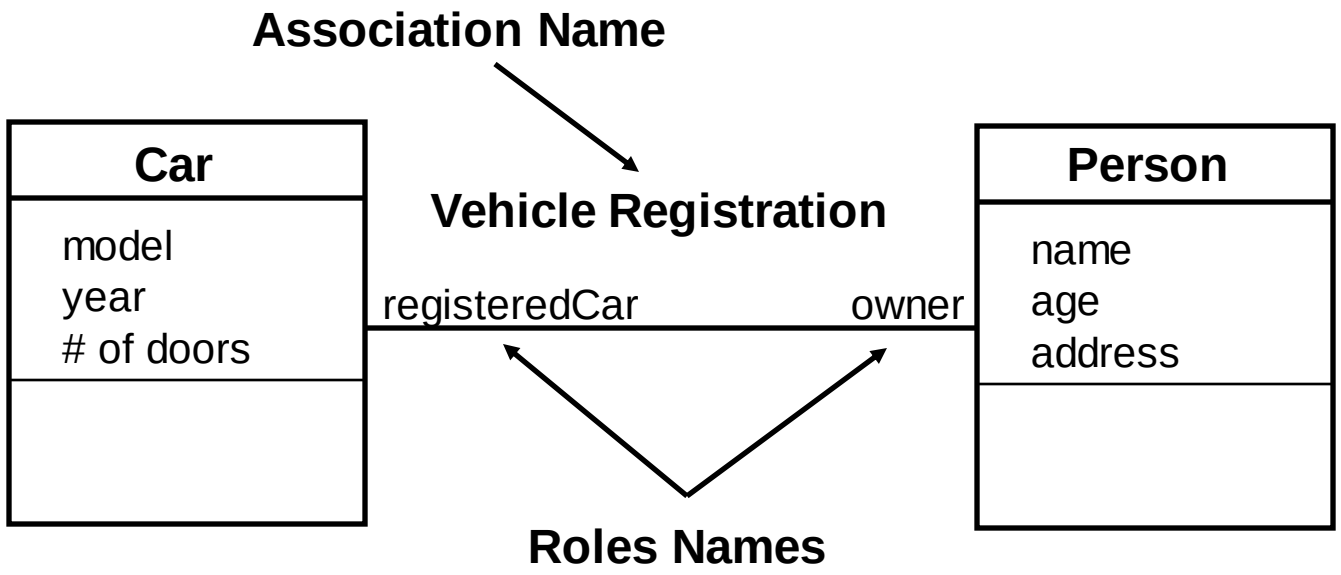
Association

- Composition and Aggregation are the two forms of association.
- Lets take an example of Department and Student. Multiple students can associate with a single Department and single student can associate with multiple Departments, but there is no ownership between the objects and both have their own lifecycle.
- Both can create and delete independently.



Named Association

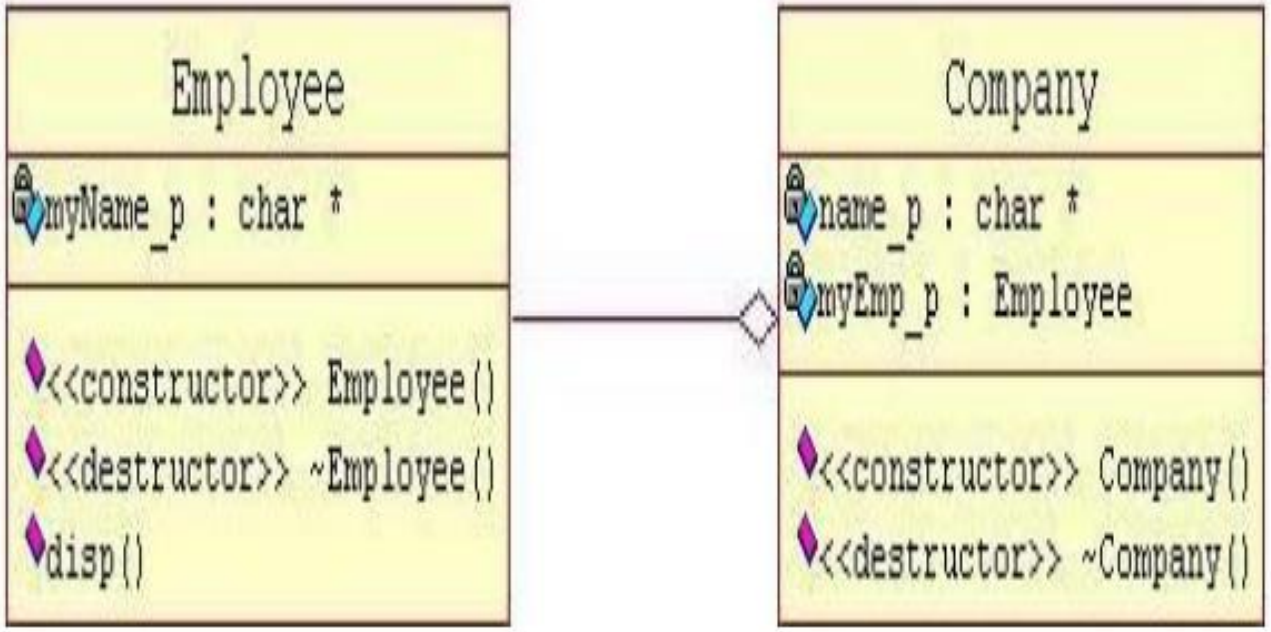
An association has a name (relationship name) on association line:



Aggregation

- Aggregation is a specialize form of Association where all object have their own lifecycle but there is a ownership like parent and child.
- Child object can not belong to another parent object at the same time.
- We can think of it as "has-a" relationship.
- Lets take an example of Employee and Company. A single Employee can not belong to multiple Companies (legally!!), but if we delete the Company, Employee object will not destroy. Here is respective Model (class diagram) for the above example

Aggregation



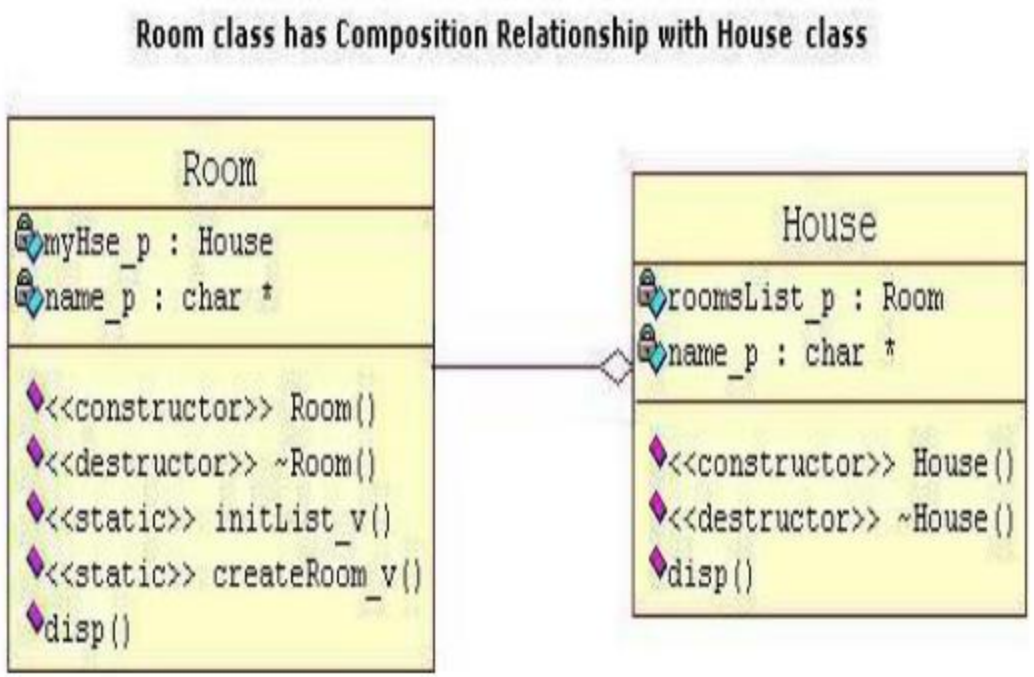
Aggregation

- It is a special form of Association where: It represents Has-A relationship.
- It is a unidirectional association i.e. a one way relationship.
- For example, department can have students but vice versa is not possible and thus unidirectional in nature. In Aggregation, both the entries can survive individually which means ending one entity will not effect the other entity

Composition

- Composition is again specialize form of Aggregation.
- It is a strong type of Aggregation.
- Here the Parent and Child objects have coincident lifetimes.
- Child object dose not have it's own lifecycle and if parent object gets deleted, then all of it's child objects will also be deleted.
- Lets take an example of a relationship between House and it's Rooms. House can contain multiple rooms there is no independent life for room and any room can not belong to two different house. If we delete the house, room will also be automatically deleted. Here is respective Model (class diagram) for the above example

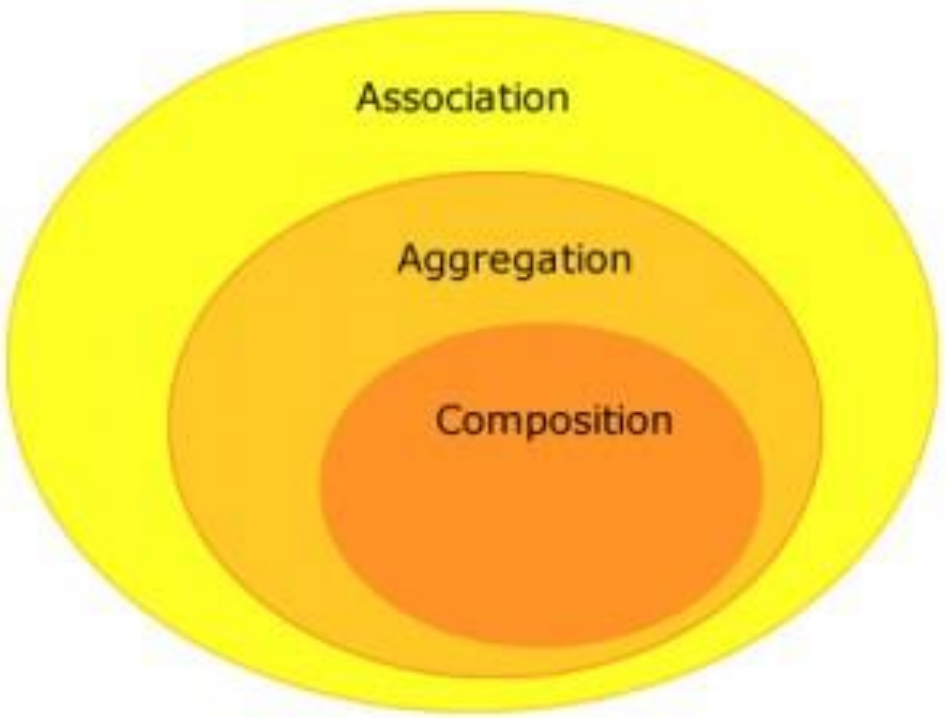
Composition



Composition

- Composition is a restricted form of Aggregation in which two entities are highly dependent on each other.
- It represents part-of relationships.
- In composition, both the entities are dependent on each other. When there is a composition between two entities, the composed object cannot exist without the other entity.
- Lets take another example of Library.
- A library can have no. of books on same or different subjects. So, If Library gets destroyed then All books within that particular library will be destroyed. i.e. book can not exist without library. That’s why it is composition.

Association, Aggregation and Composition



Aggregation vs. Composition

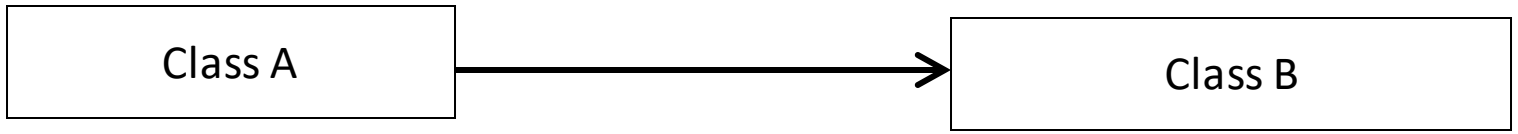
- Aggregation:- Aggregation indicates a relationship where the child can exist separately from their parent class. Example: Automobile (Parent) and Car (Child). So, If you delete the Automobile, the child Car still exist.
- Composition:- Composition display relationship where the child will never exist independent of the parent. Example: House (parent) and Room (child). Rooms will never separate into a House

Aggregation vs. Composition

- Dependency:- Aggregation implies a relationship where the child can exist independently of the parent. For example, Bank and Employee, delete the Bank and the Employee still exist. whereas Composition implies a relationship where the child cannot exist independent of the parent. Example: Human and heart, heart don’t exist separate to a Human
- Type of Relationship:- Aggregation relation is “has-a” and composition is “part-of” relation.
- Type of association:- Composition is a strong Association whereas Aggregation is a weak Association

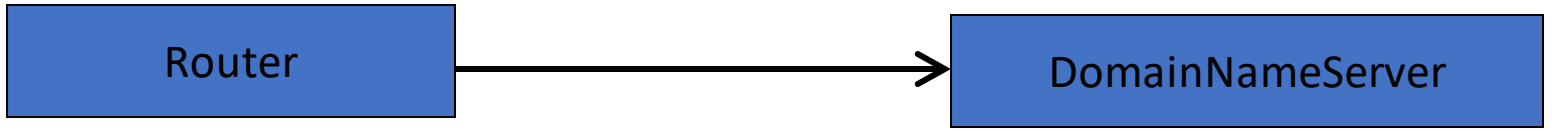
Navigability (Association Relationship)

- Association can go both ways at once, with both classes being able to make requests of the other.
- More often they go only one way, with class A asking B to help it out, but class B is not asking class A for anything.
- “The direction of an association is called navigability, and is represented on UML class diagram by a small open arrowhead pointing toward the collaborator.”
- If each class asks the other to do something, then no arrows are used, just an unadorned line.



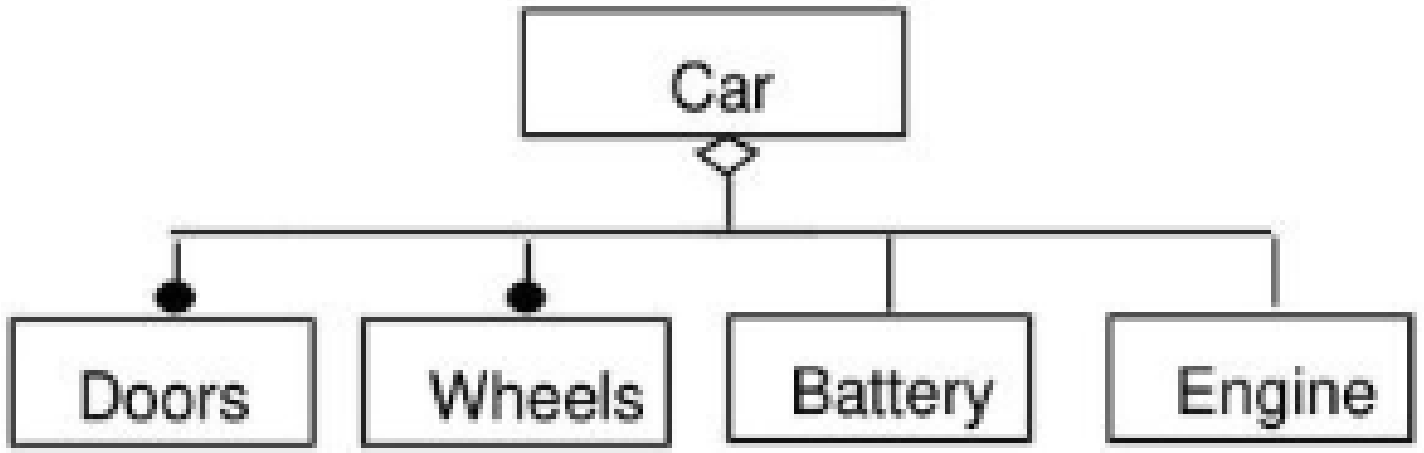
Example: Navigability

- We can constrain the association relationship by defining the navigability of the association. Here, a Router object requests services from a DNS object by sending messages to (invoking the operations of) the server. The direction of the association indicates that the server has no knowledge of the Router.



FIXED AGGREGATION

- It has a fixed structure
- Numbers and types of the parts/components are pre-defined/fixed.



VARIABLE AGGREGATION

- The number of levels of aggregation is fixed, but the number of parts/components are variable.
- Ex.1

- Ex.2

RECURSIVE AGGREGATION

- The Number of levels are un-limited.
- An aggregate contains the components of its own types.
- Contains directly or indirectly an instance of the same kind of aggregate