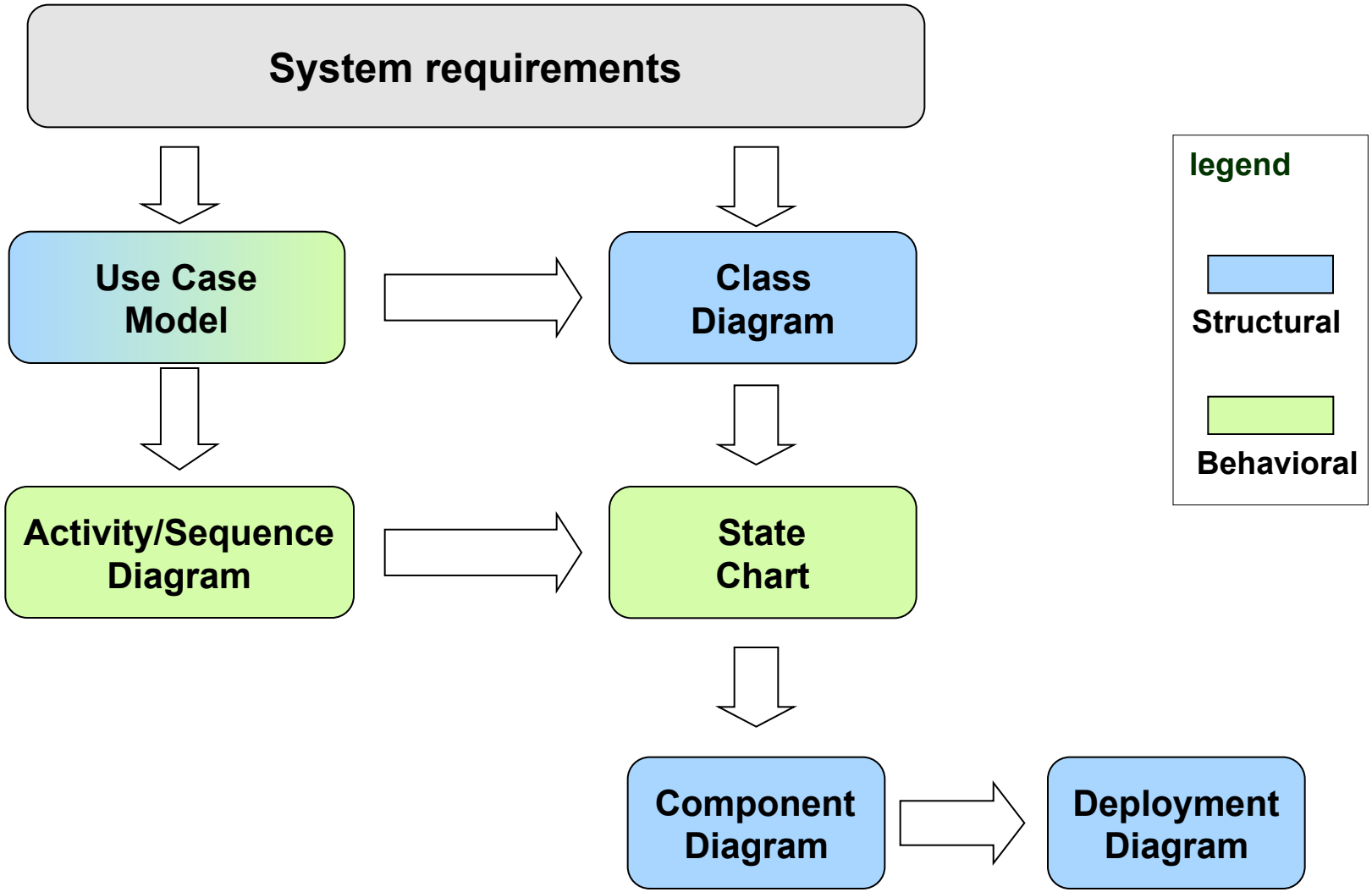
Classes and Objects

Outline

- Introduction
  - Structural modeling
- Classes
  - Attributes and operations
- Relations
  - Associations
  - Dependencies, compositions
- Generalization
  - Inheritance
  - Interfaces
- Object Diagrams

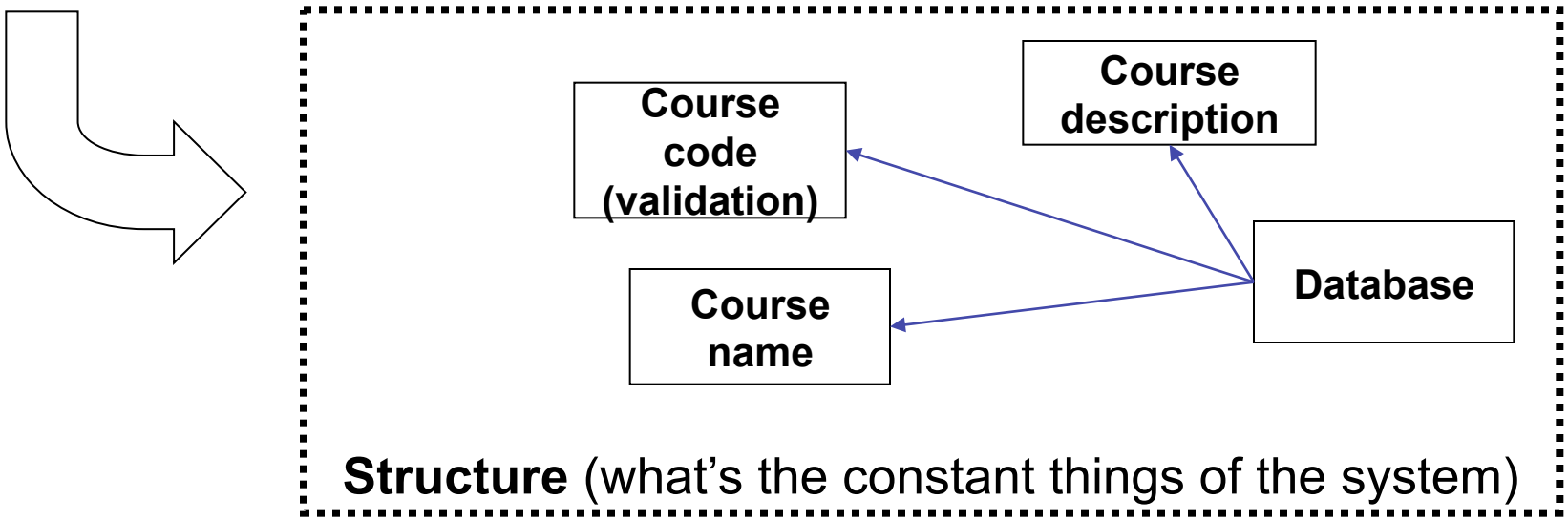
Design Process



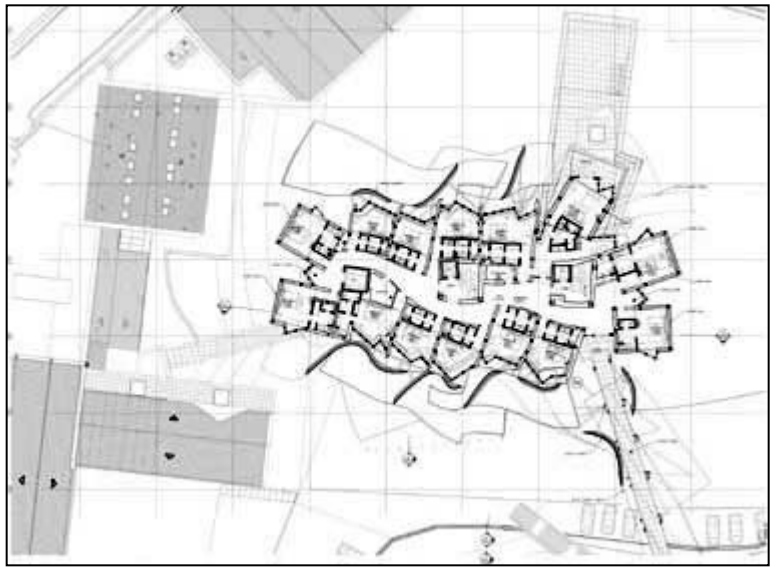
From Requirements to Structure

1. Administrator enters course name, code and description
2. System validates course code
3. System adds the course to the data base and shows a confirmation message

Requirements Document



What is Structural Modeling?



A structural design defines the artifact unchanging characteristics, which do not change over time.

Structural Modeling in Information Systems

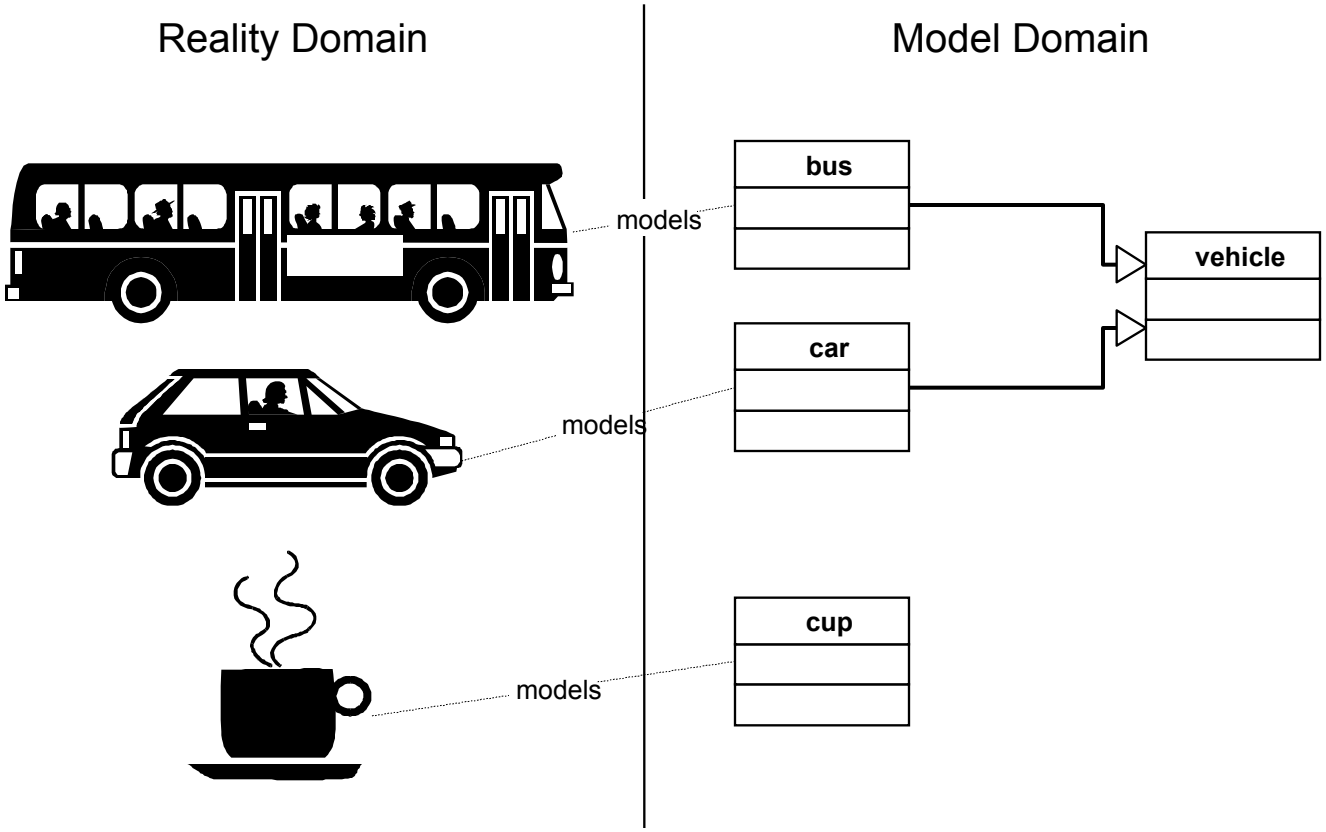
- Static structure of the model
  - the entities that exist (e.g., classes, interfaces, components, nodes)
  - relationship between entities / classes
  - internal structure of the entities /classes

Outline

- Introduction
  - Structural modeling
- Classes
  - Attributes and operations
- Relations
  - Associations
  - Dependencies, compositions
- Generalization
  - Inheritance

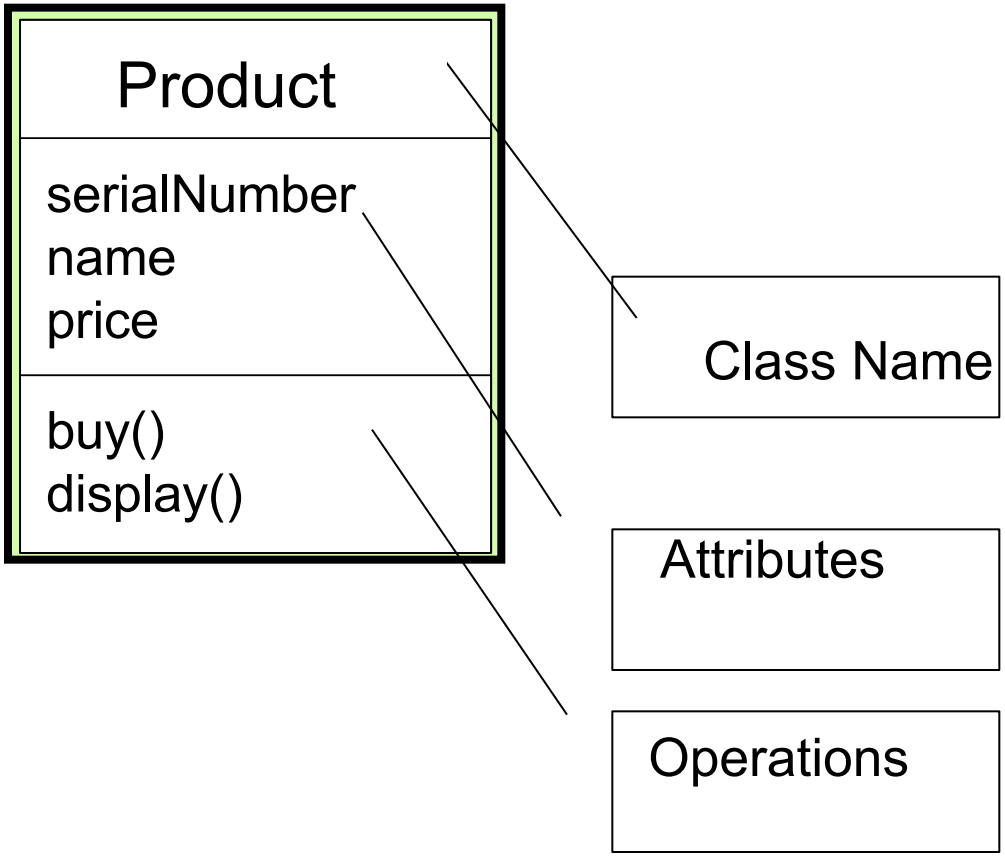
Object-Oriented Approach

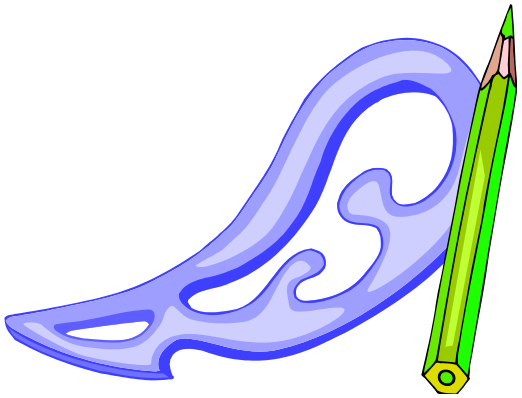
- Objects are abstractions of real-world or system entities



Classes

- A class is a template for actual, in-memory, instances





Attributes - Signature

[visibility] **name** [[multiplicity]] [: type]

- visibility: the access rights to the attribute ?
- multiplicity: how many instances of the attribute are they:
  - middleName [0..1] : String, phoneNumber [1..*]
- Type: the type of the attribute (integer, String, Person, Course)

```
+ isLightOn : boolean = false
- numOfPeople : int
```

Operations - Signature

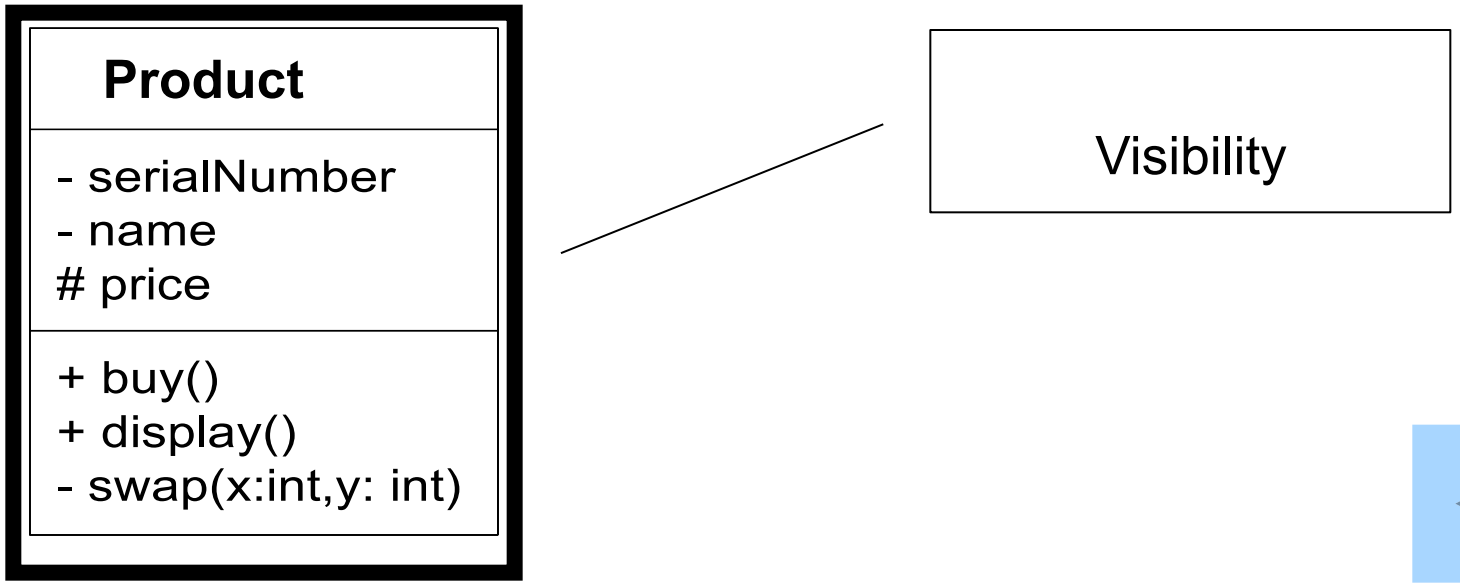
[visibility] **name** [(parameter-list)] [: return-type]

- An operation can have zero or more parameters, each has the syntax:
  - name : type [=default-value]

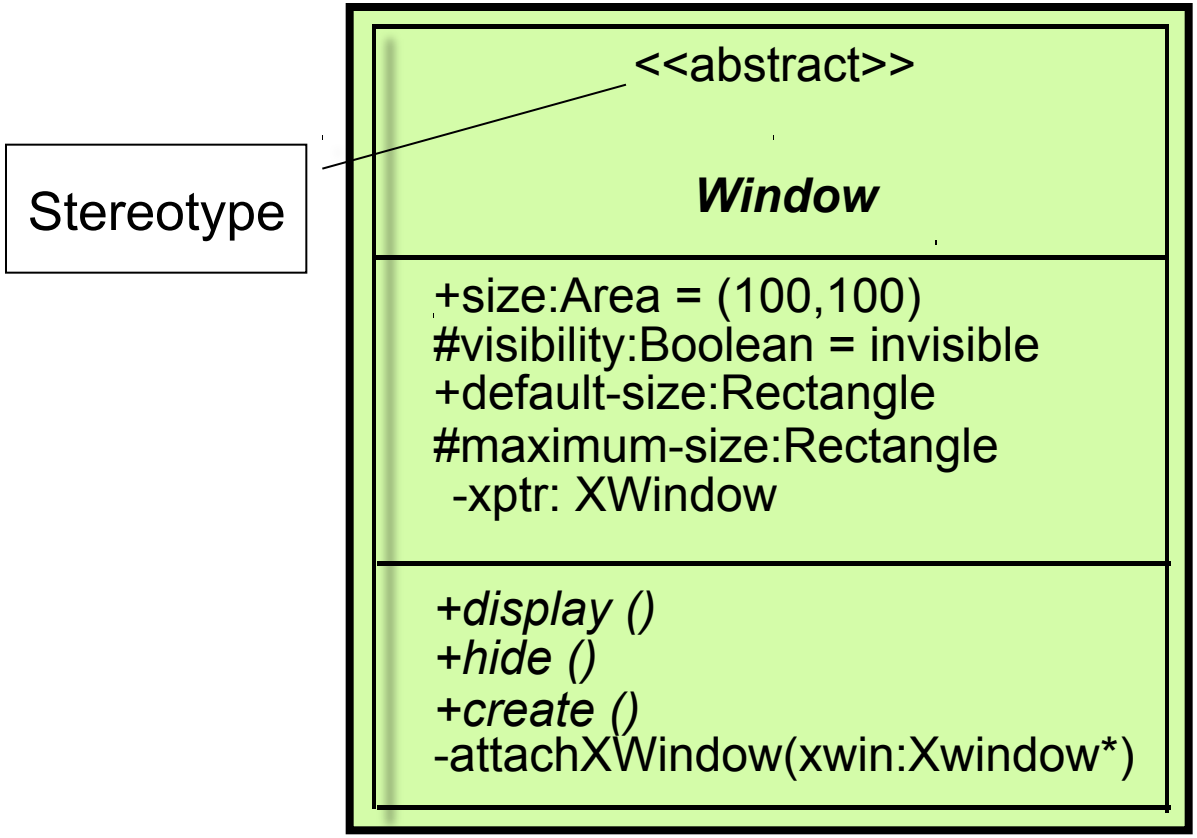
```
+ isLightOn() : boolean
+ addColor(newColor : Color)
+ addColor(newColor : Color) : void
# convertToPoint(x : int, y : int) : Point
- changeItem(key : string) : int
```

Visibility

- public (+) – external objects can access the member
- private (-) – only internal methods can access the member
- protected (#) – only internal methods, or methods of specialized objects can access the member



Full Blown Class

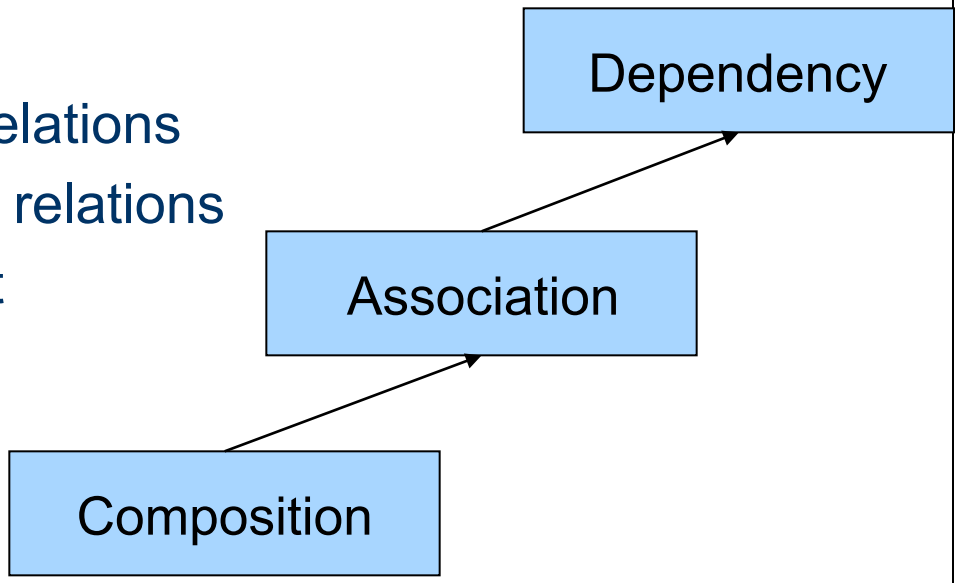


Outline

- Introduction
  - Structural modeling
- Classes
  - Attributes and operations
- Relations
  - Associations
  - Dependencies, compositions
- Generalization
  - Inheritance
  - Interfaces
- Object Diagrams

Relations

- A relation is a template for a connection between two instances.
- Relations are organized in a Hierarchy:
  - Dependency: dynamic relations
  - Associations: consistent relations
  - Composition: whole-part relations



How do you find the classes / objects?

Look for NOUNs in a sentence.

•A <u>noun</u> is a <u>person, place or thing</u>

1.A campaign may be conducted using one or more advertisements.

2.A grade may be allocated to one or more staff members.

3.There are between 1 and 7 cards in a hand.

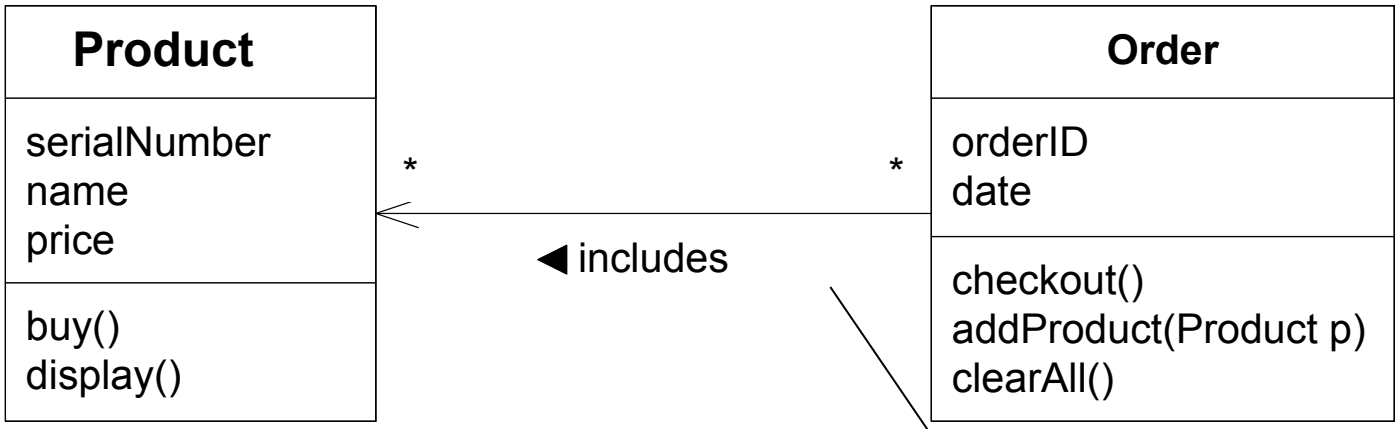
•What are the NOUNs in the sentences above?

How do you find an relation/association?

- A relation is a connection between objects / classes.
- Look for the verbs or action words in the sentence.
- What are the verbs in the following sentences?

1. A campaign may be conducted using one or more advertisements.
2. A grade may be allocated to one or more staff members.
3. There are between 1 and 7 cards in a hand.

Associations



- Objects on both sides of the association can find each other

Multiplicity
Indicates cardinality
- – **1:1default**
- **3 – exactly 3 object**
- *** (or n) - unbounded**
- **1..* - 1 to eternity**
- **3..9 – 3 to 9**

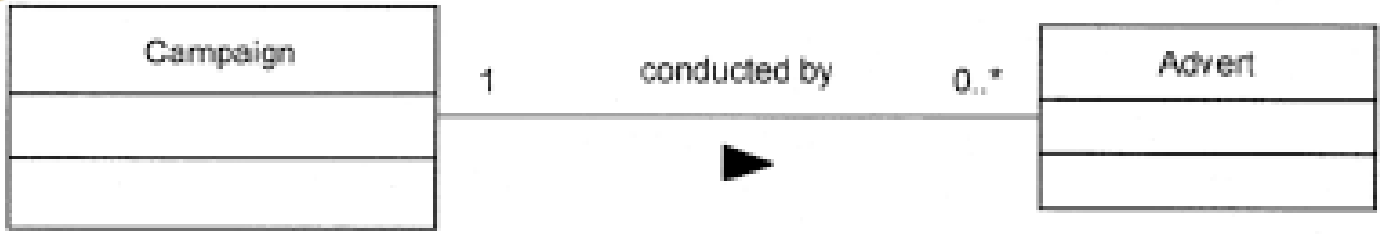
An order includes many products. (one or more)
A product is included in many orders. (one or more)

Multiplicity

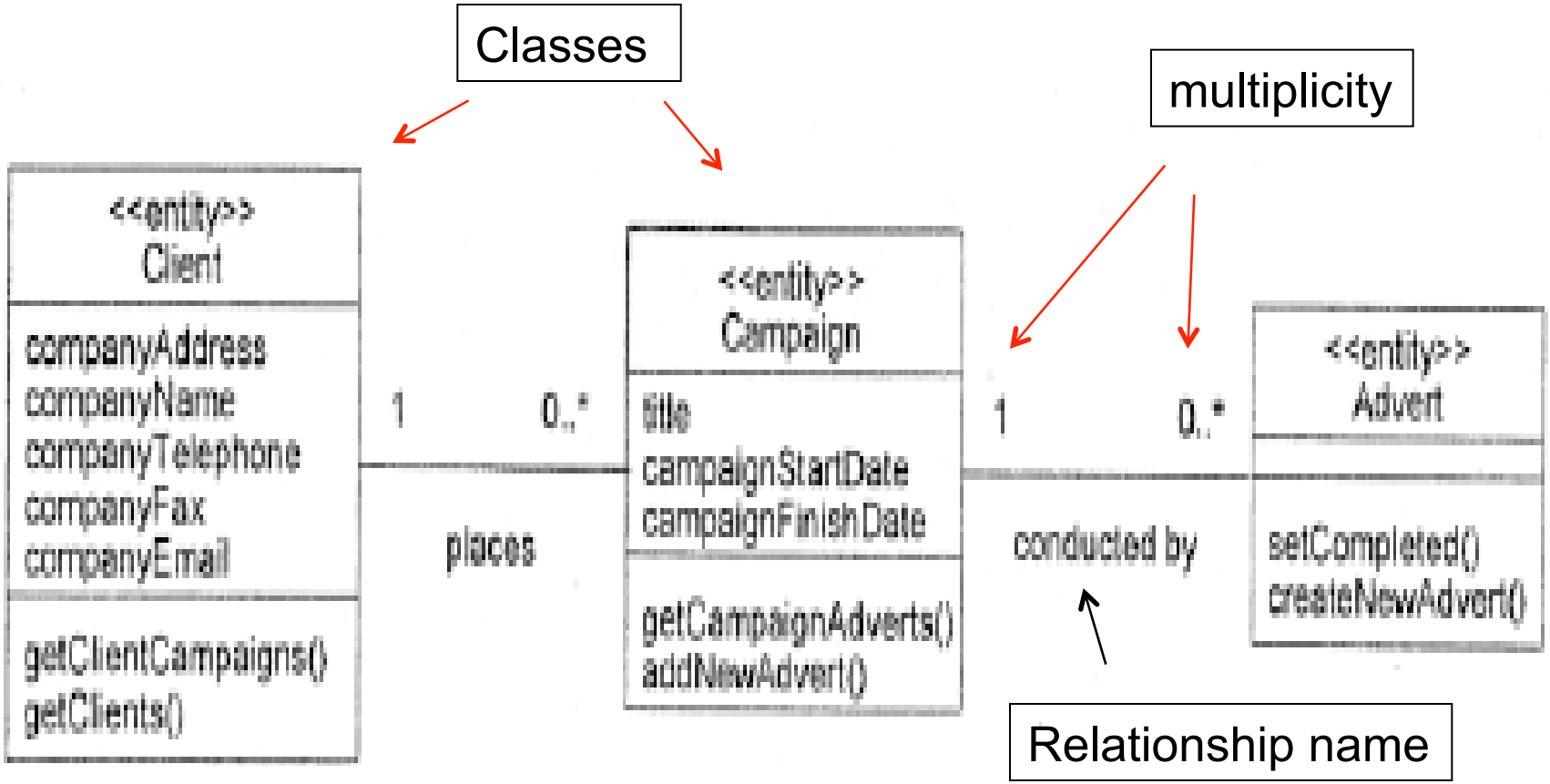
- Some examples of specifying multiplicity:

– 0…1	Optional (0) or 1
– 1	Exactly 1
- 0…*	Zero or more
- 1…*	One or more
- 2…6	Range of values between 2…6

Associations – Examples

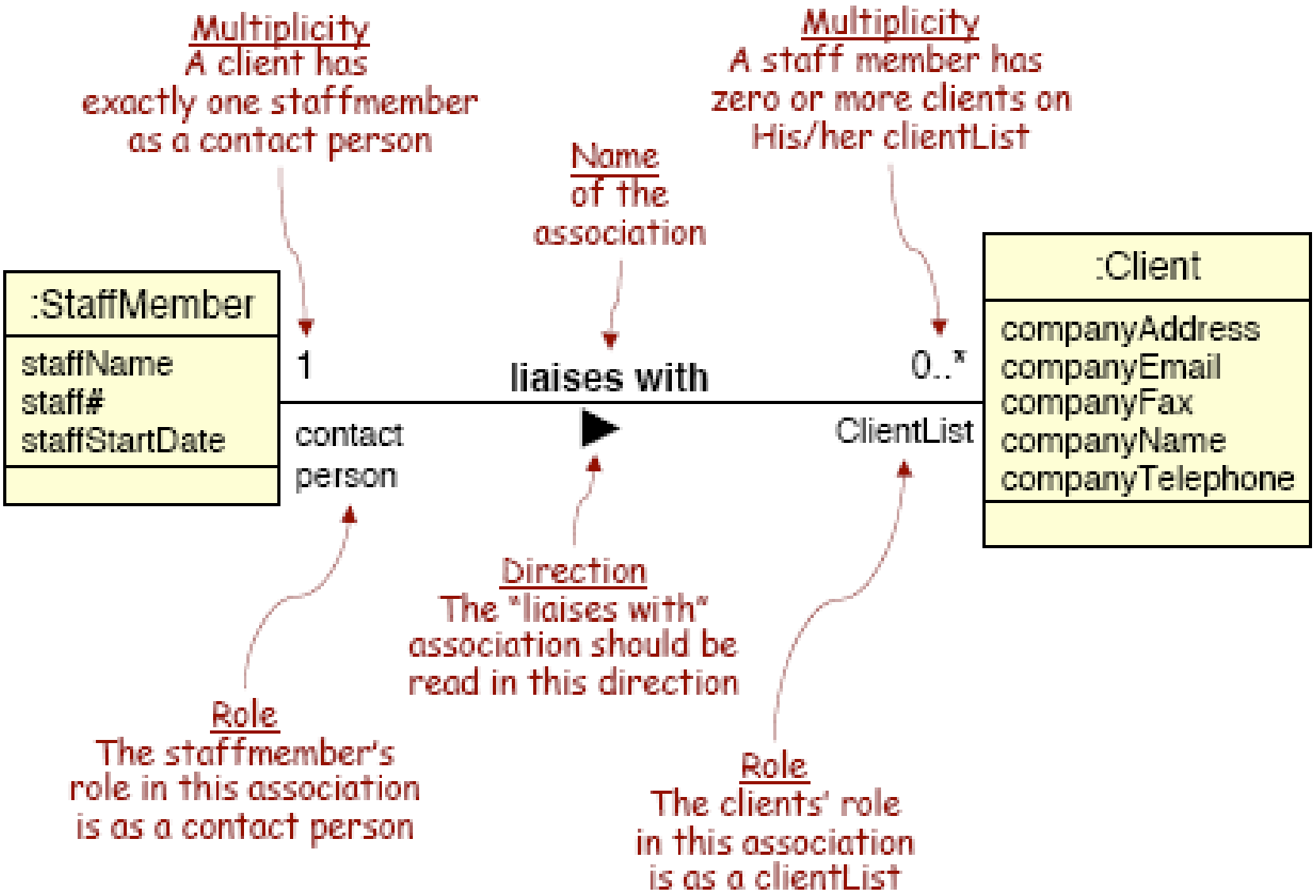


Associations



A client places 0 or more campaigns.
Each campaign is placed by a client.
A campaign is conducted using 0 or more advertisements.
Each advertisement is conducted for a particular campaign.

Associations

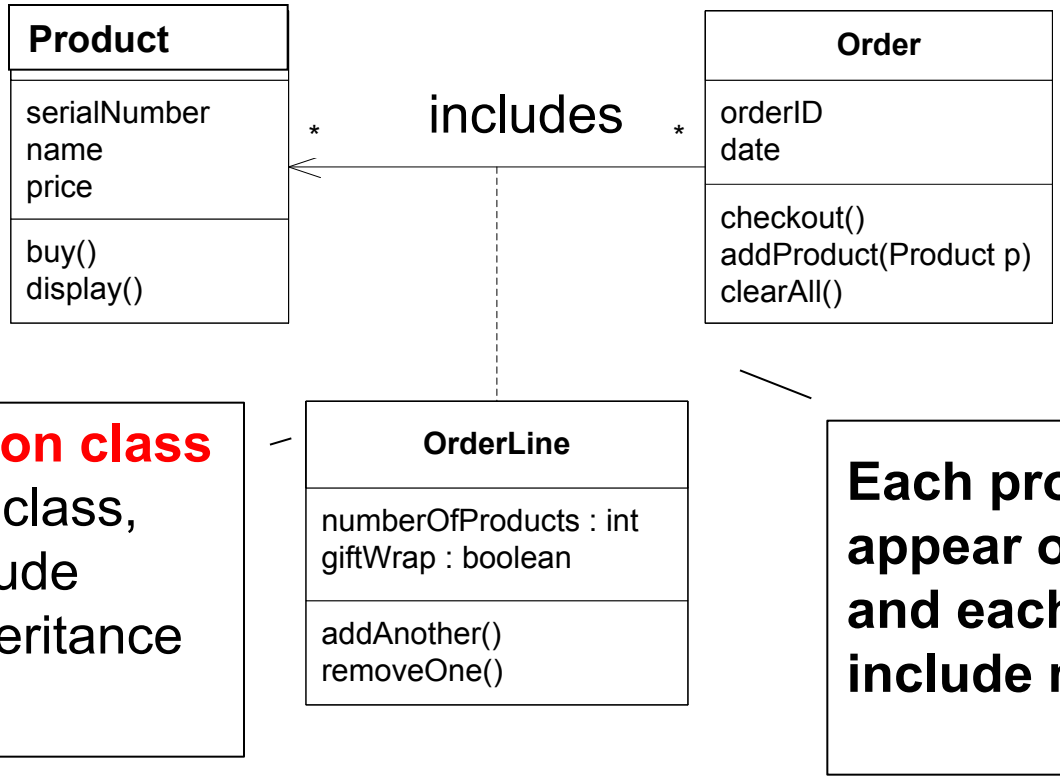


Questions

- What does multiplicity of 1..5, 7, 9..* indicate?
- Identify the classes and multiplicity in the relationships
  - A developer has one or more computers that play the role of their build machine.
  - Draw diagrams
    - A teacher teaches an unlimited number of classes and a student takes 4 to 6 classes.
    - A class has 10 to 30 students in it.
    - A course has 6 modules.
    - Student study 1 to 6 modules.

Association Classes

Denoted as a class attached to the association, and specify properties of the association

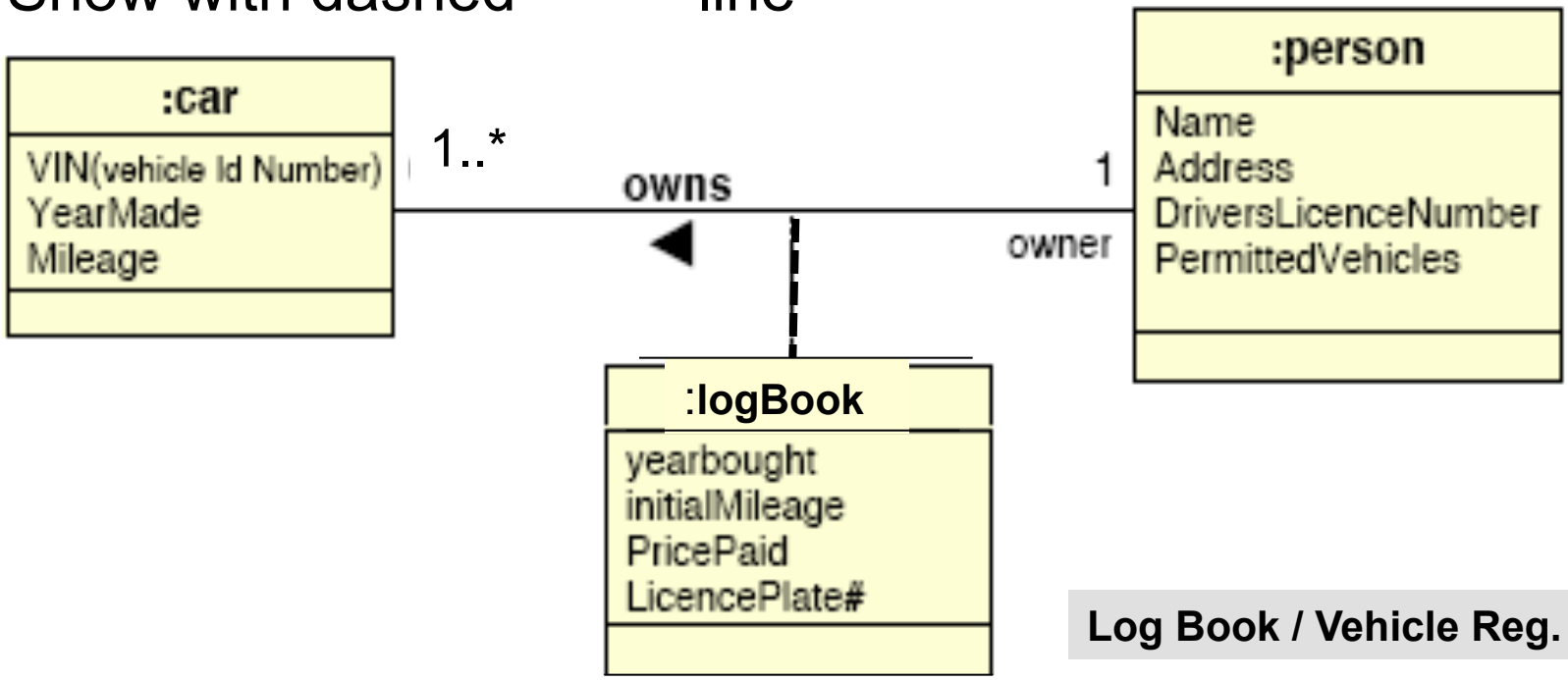


An **association class** is a “normal” class, and may include relations, inheritance etc.

Each product can appear on many orders and each order may include many products

Association Classes

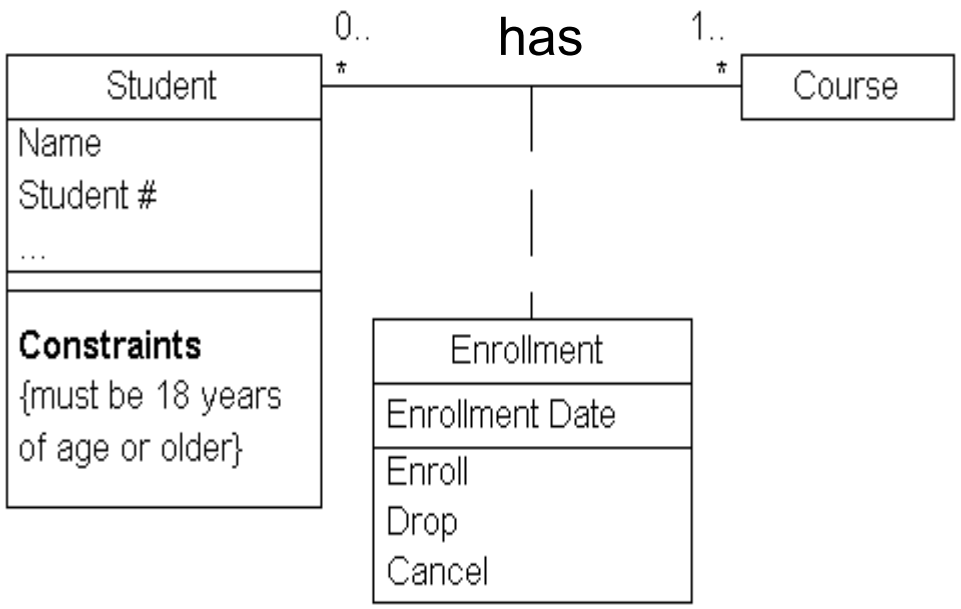
- A person has 1 or more cars. Each car is associated with one person.
- Hidden class /Association class is Log Book / Vehicle Reg Cert.
- Show with dashed ------- line



Log Book / Vehicle Reg. Cert

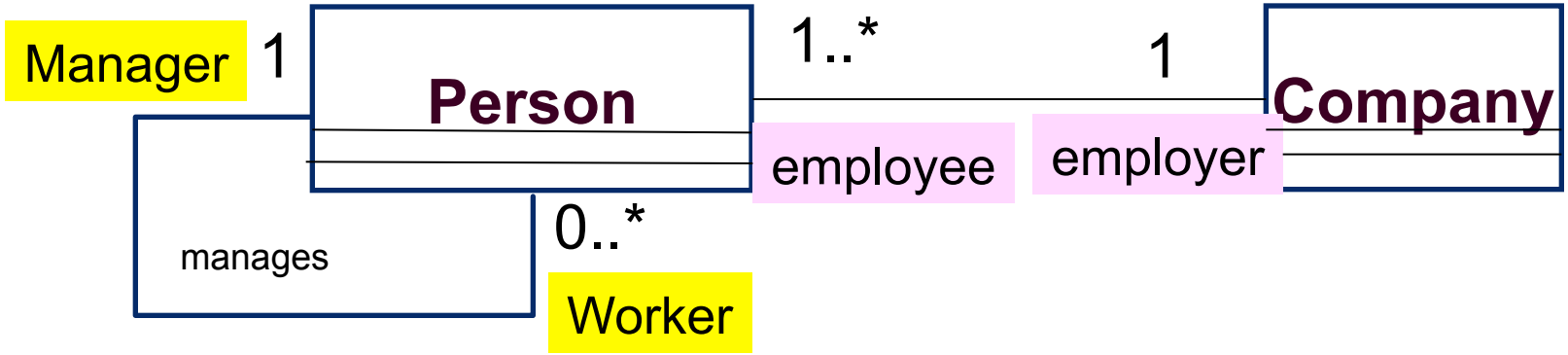
Association Class

A student enrols on one or more courses.
A course may have many students on it or may have none.
Hidden class: Enrolment



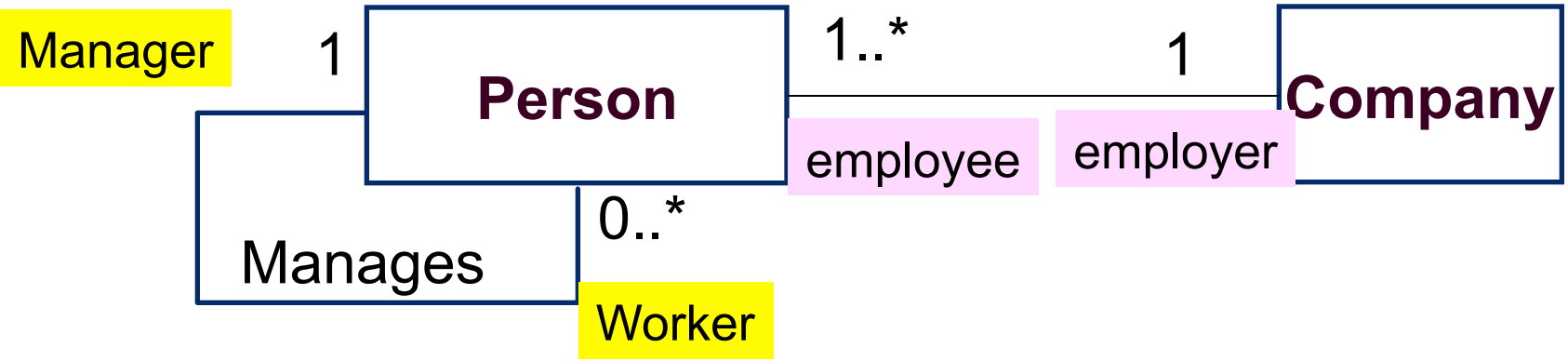
Role Names

- Names may be added at each end of the association
- Provide better understanding of the association meaning
- Especially helpful in self-associated classes



Role Names

- A person manages zero or more people.
- Each person (worker) is managed by one manager.
- A company has one or more employees
- A person is employed by one company.



Ternary Associations

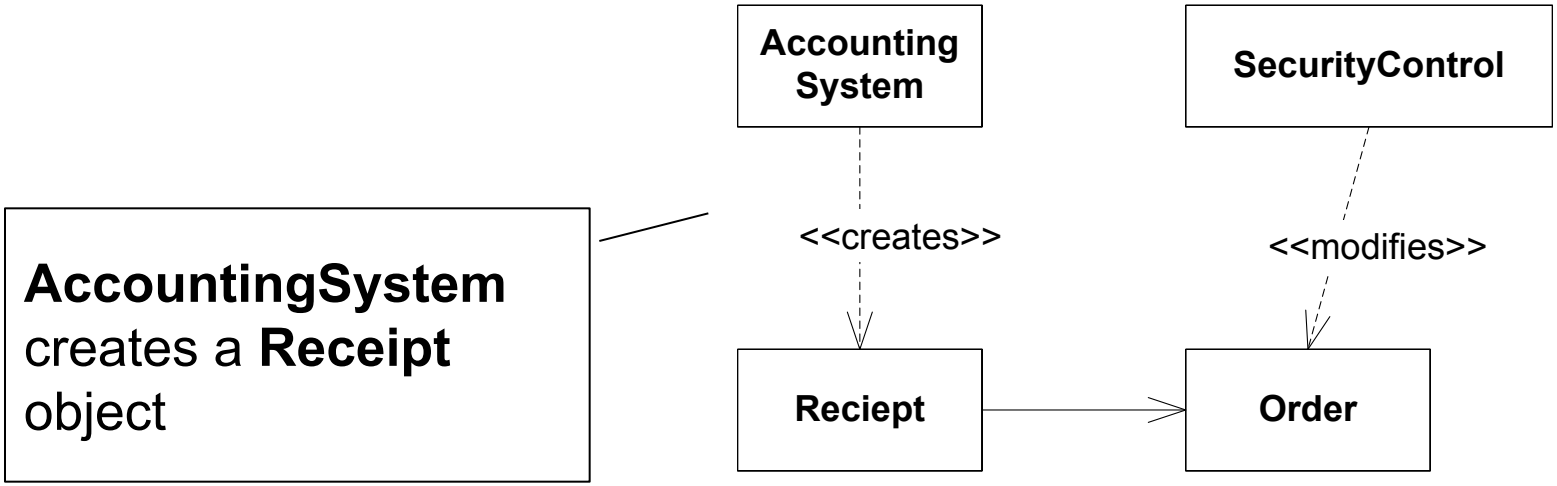
An association exists between three classes. Create a LINK CLASS that links all three classes

Outline

- Introduction
  - Structural modeling
- Classes
  - Attributes and operations
- Relations
  - Associations
  - Dependencies, compositions
- Generalization
  - Inheritance

Dependency

- Notated by a dotted line - - - - - - - - - - - - - - →
- The most general relation between classes
- Indicates that an object affects another object



Dependency – cont'd

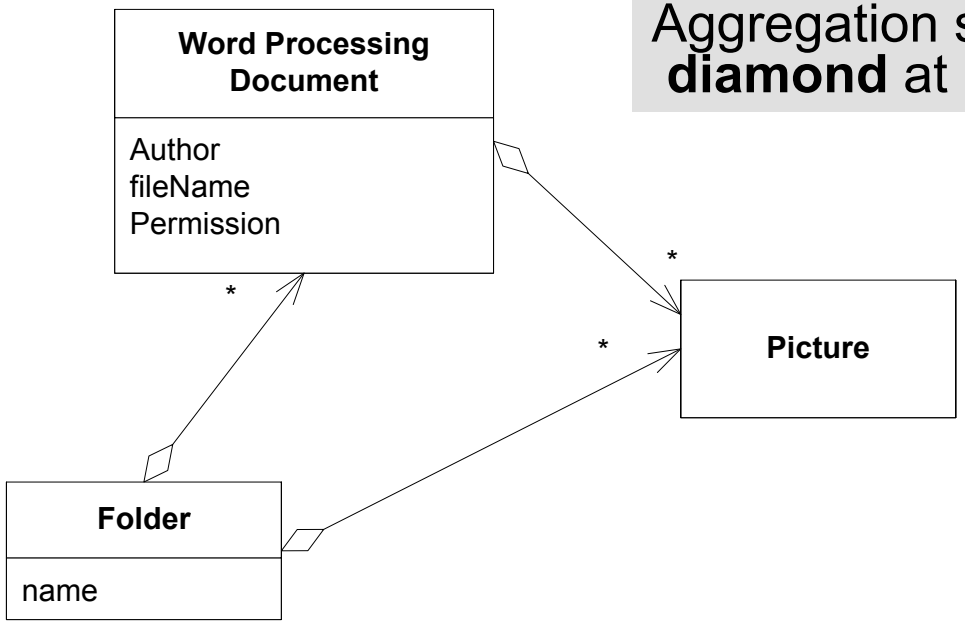
- Dependencies are the most abstract type of relations.
- Properties:
  - Dependencies are always directed (If a given class depends on another, it does not mean the other way around).
  - Dependencies do not have cardinality.

- Types:
  - «call»
  - «create»

Aggregation

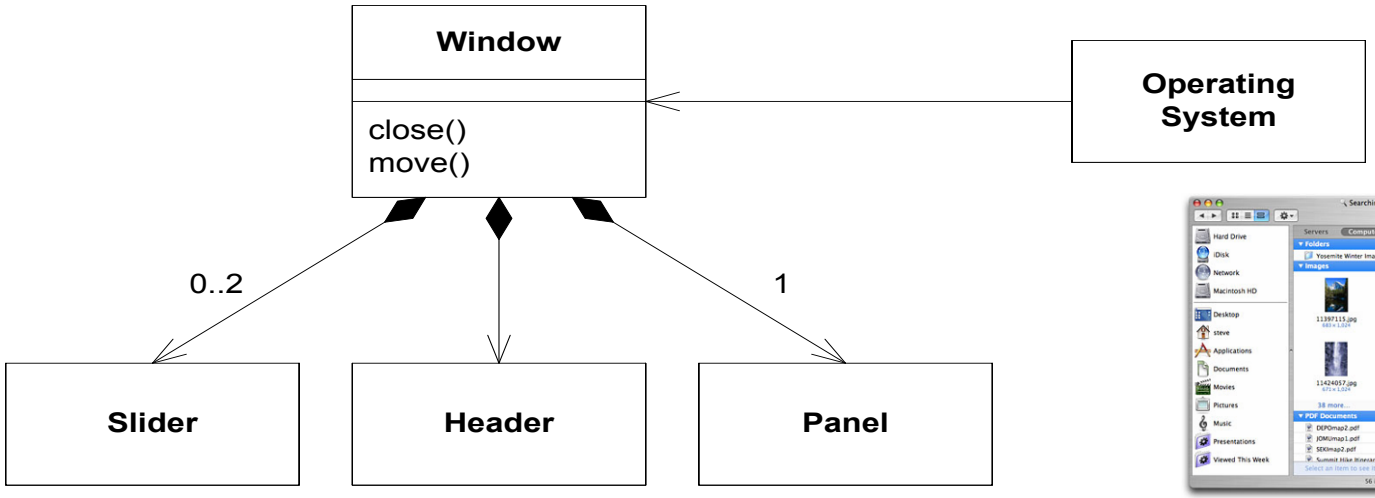
- "Whole-part" relationship between classes
- "Has-a" relationship between classes
- Assemble a class from other classes
  - Combined with "many" - assemble a class from a couple of instances of that class

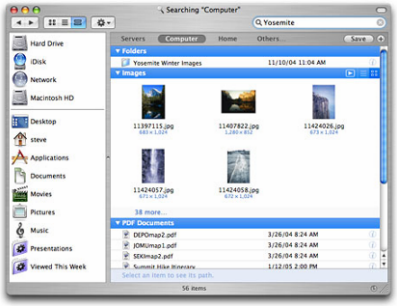
Aggregation shown by **hollow diamond** at larger end



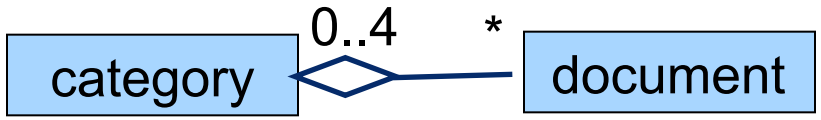
Composition

- Composition is a stronger form of aggregation
- **Larger object itself cannot exist without the smaller object(s)**
- If the whole object is deleted, then the parts deleted with it
- Multiplicity at whole end must be 1.
- Signified by **black diamond** at larger end



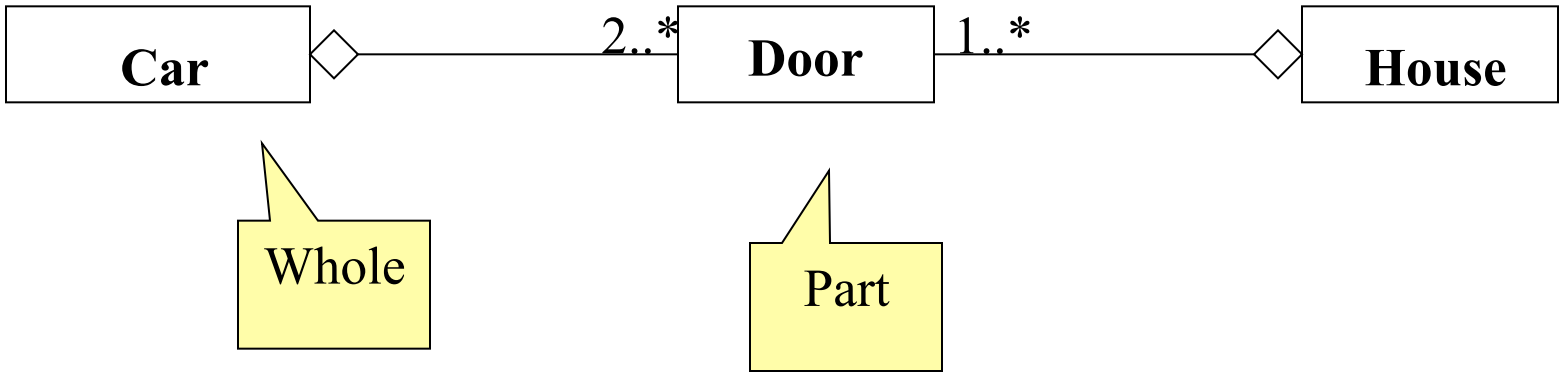


Composition vs. Aggregation

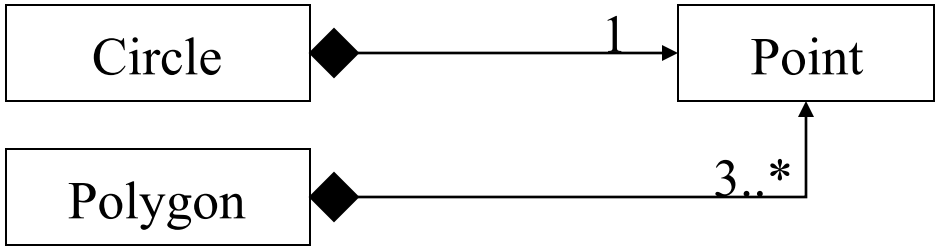
Aggregation	Composition
Part can be shared by several wholes 	Part is always a part of a single whole 
Parts can live independently (i.e., whole cardinality can be 0..*)	Parts exist only as part of the whole. When the wall is destroyed, they are destroyed
Whole is not solely responsible for the object	Whole is responsible and should create/destroy the objects

Aggregation & Composition

- **Aggregation**: “is-part-of” relationship



- **Composition:** The whole is the sole owner of its part. The part object may belong to only one whole.

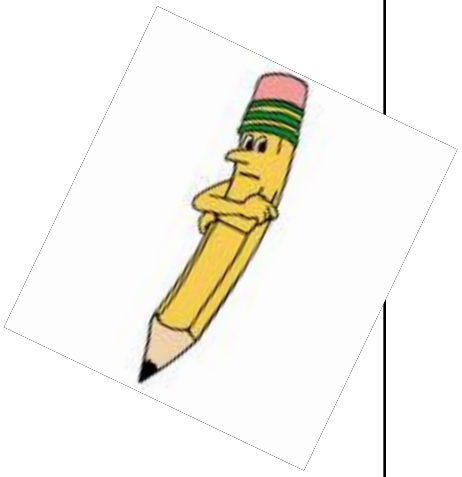


Exercise – Aggregation/Composition?

- Draw class diagrams to represent

(A) PC which consists of:

- CPU (1 to 4)
- Hard Disk (1 or more)
- Monitor (1)
- Keyboard (1)



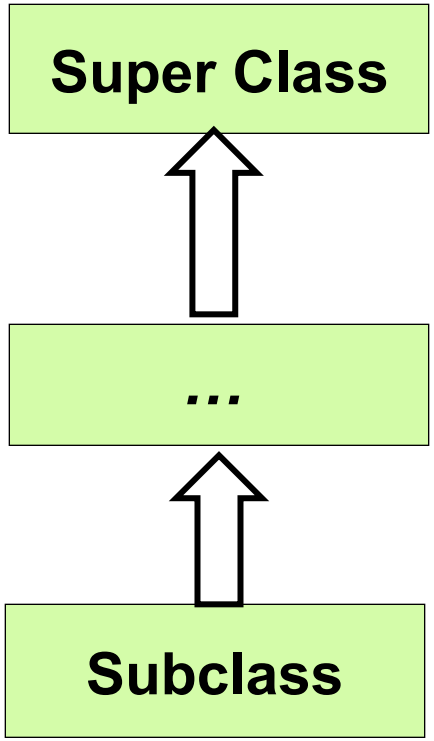
(B) A house has 2 or more bedrooms

Outline

- Introduction
  - Structural modeling
- Classes
  - Attributes and operations
- Relations
  - Associations
  - Dependencies, compositions
- Generalization
  - Inheritance

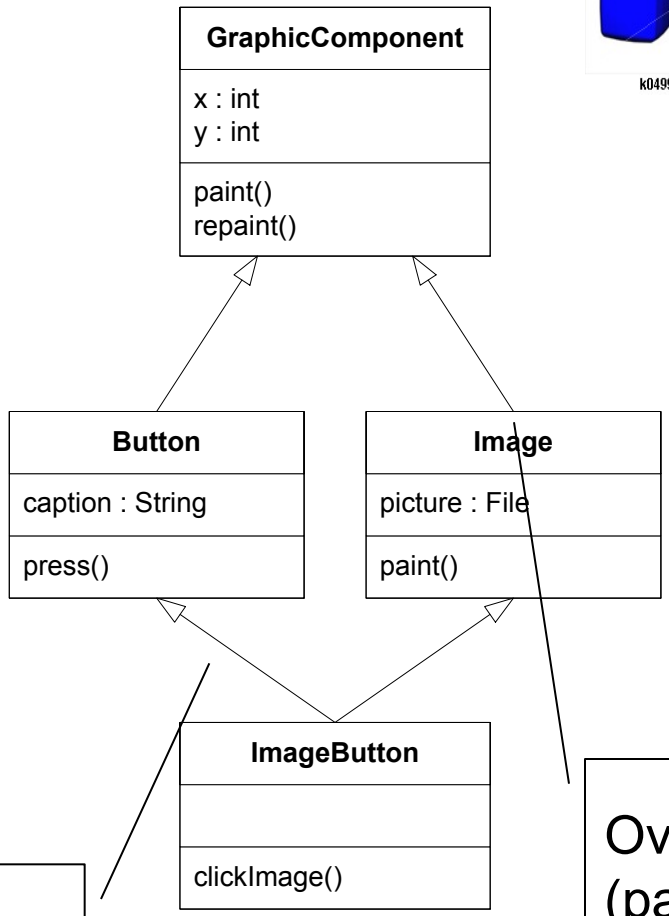
Generalization – Definitions

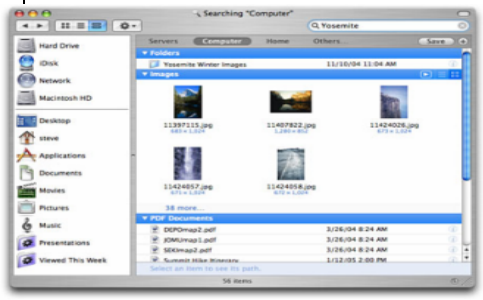
- Super Class (Base class)
  - Provides common functionality and data members
- Subclass (Derived class)
  - Inherits public and protected members from the *super class*
  - Can extend or change behavior of super class by *overriding* methods
- Overriding
  - Subclass may override the behavior of its super class



Generalization – advantages

- Modularity:
  - Eliminate the details
  - Find common characteristics among classes
  - Define hierarchies
- Reuse:
  - Allow state and behavior to be specialized

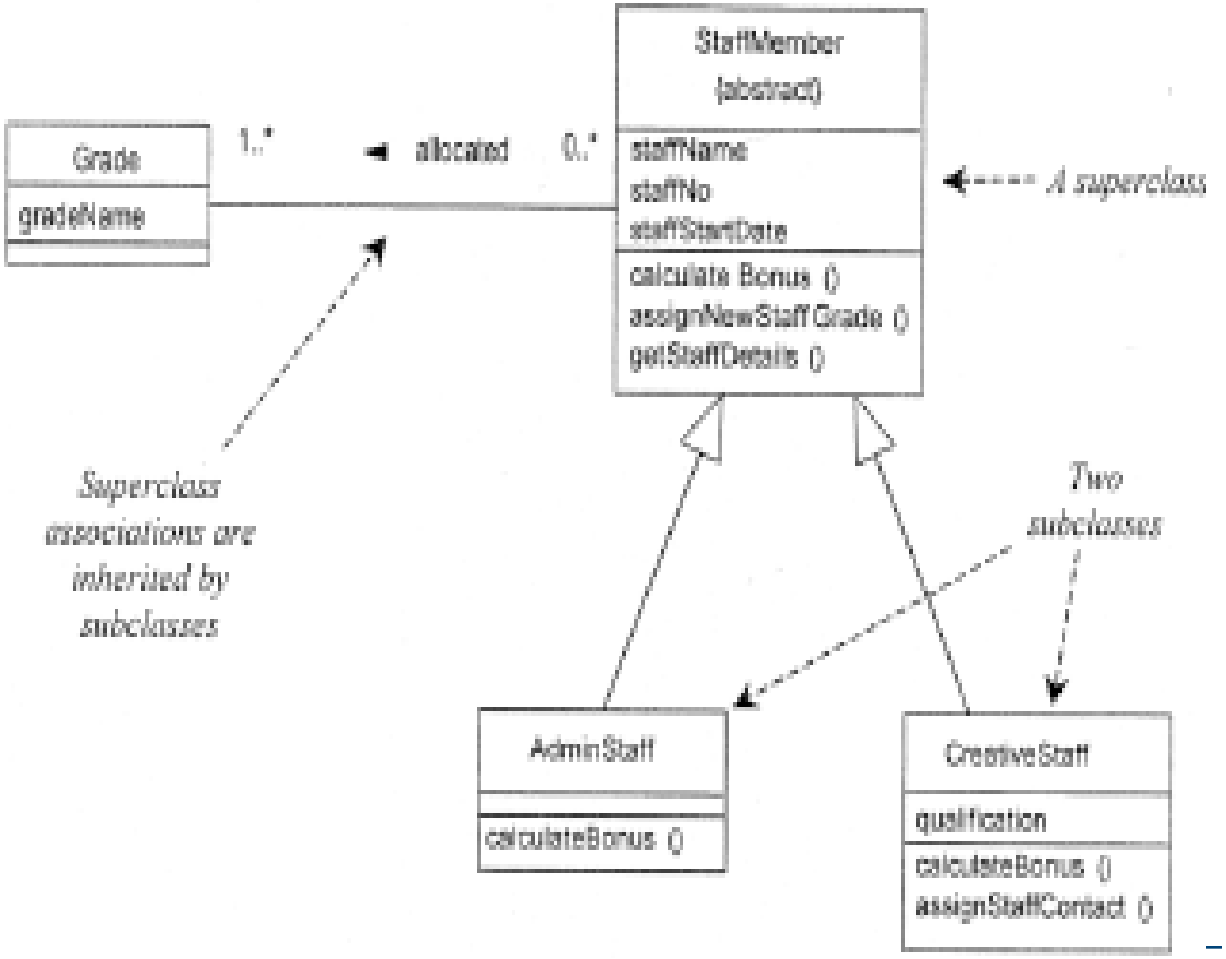




Multiple Inheritance

Overriding (paint())

Generalisation



→ Notes:

Generalisation

- Notes on previous slide:
- Subclasses inherit attributes, associations and operations from the superclass
- A subclass may override an inherited aspect
  - AdminStaff & CreativeStaff have different methods for calculating bonuses
- Superclasses may be declared (abstract), meaning they have no instances
  - Implies that the subclasses cover all possibilities
  - E.g. there are no staff other than AdminStaff and CreativeStaff

Exercise

- Name the Superclass.
- What type of class is the Superclass?
- Name the two subclasses.
- What method is overrridden in the subclasses?
- Complete the following:

A staff member is allocated _____ or more grades.

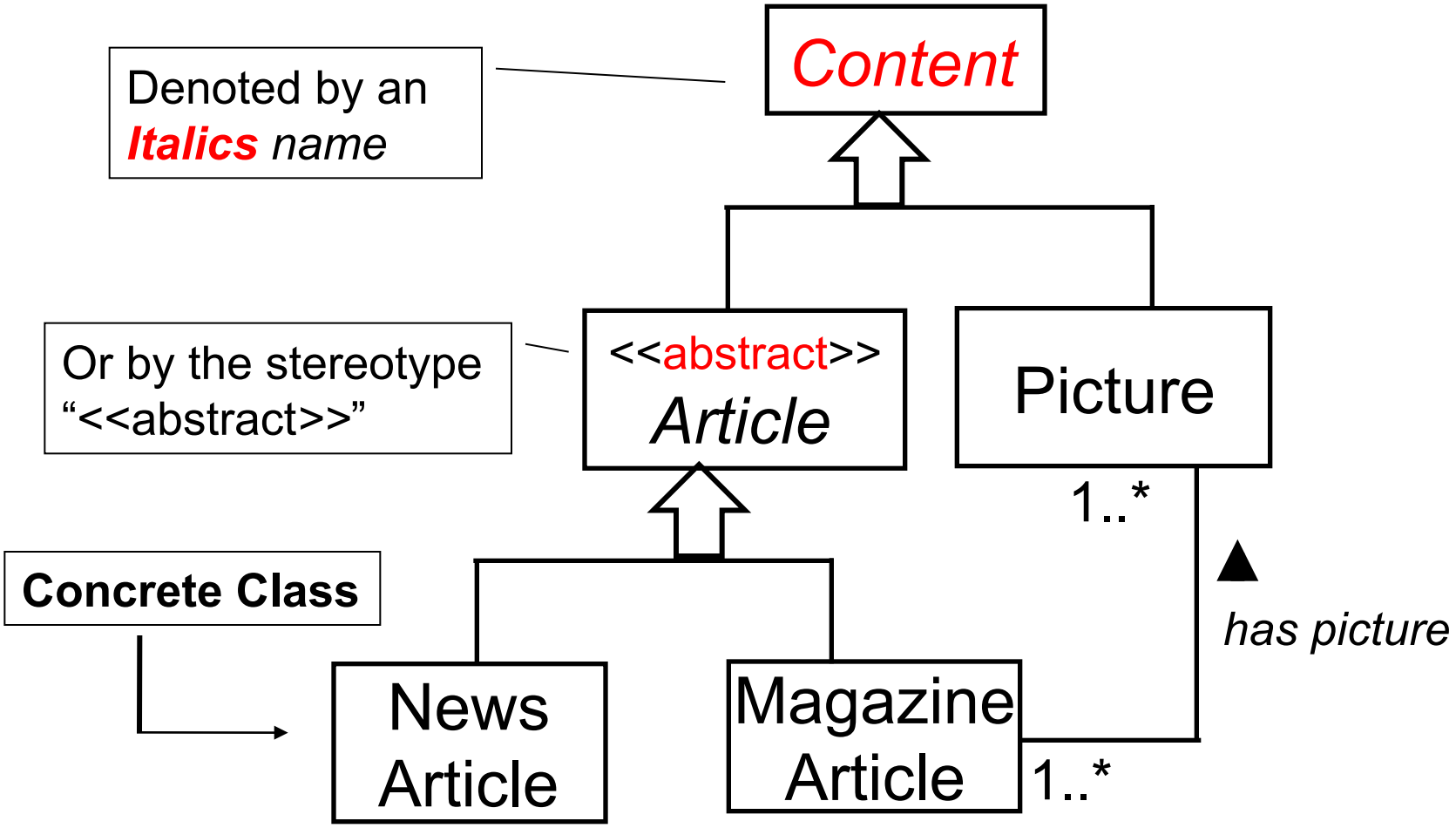
A grade has _____ or _____ staff allocated to it.

There are two types of staff in the organisation

_____ and _____

Abstract Class

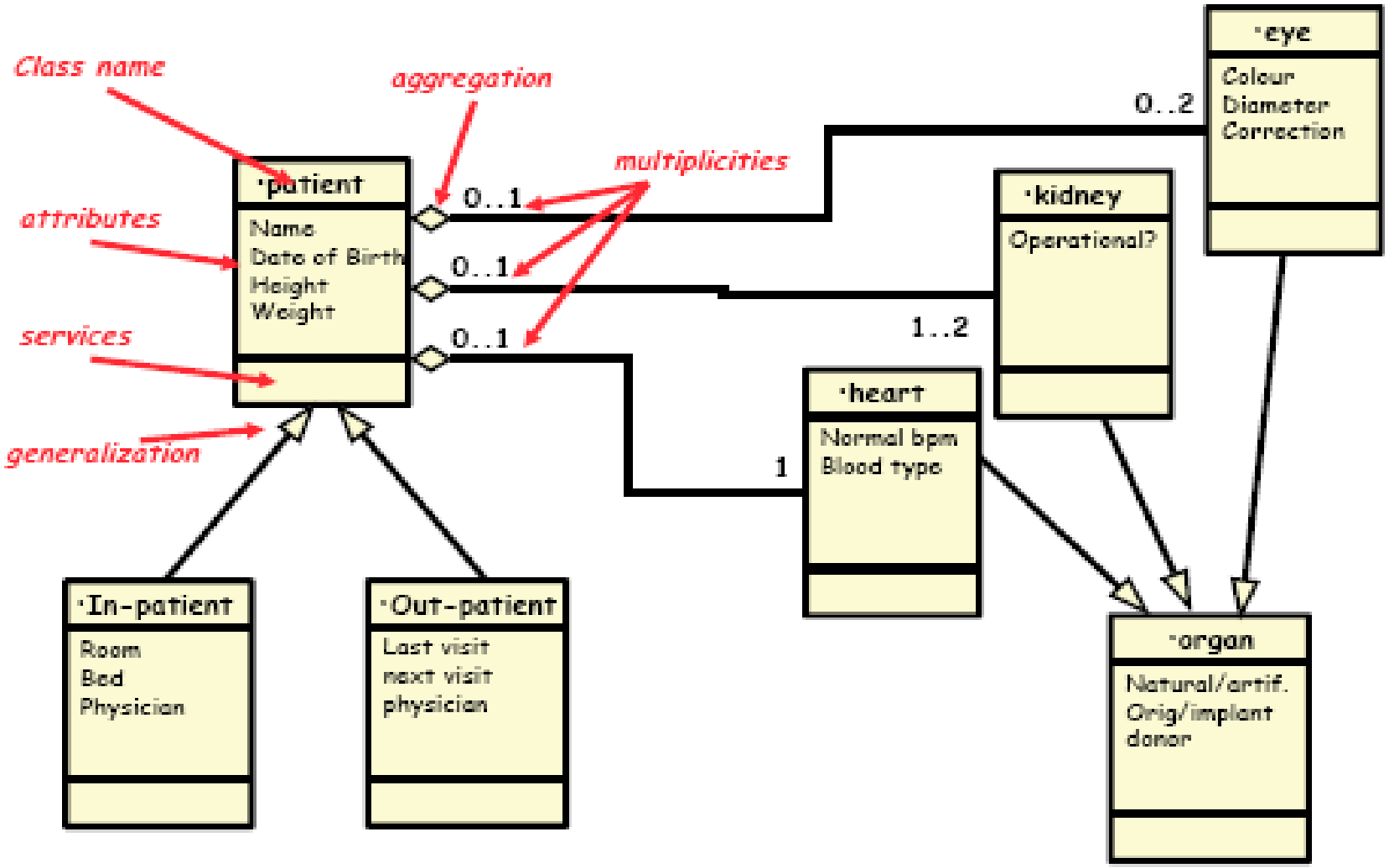
- Abstract Class -- A class that has no direct instances



Class Diagram: Example

- Draw a **class diagram** for the following specification for a hospital system. Include classes, associations, multiplicity symbols etc., as appropriate, in your diagram.
- A patient may be an in-patient or out-patient.
- Details are kept on the following organs that a patient has: heart, kidney and eyes as they may natural or artificial, original or implanted from a donor
- Add some appropriate attributes to each class

Class Diagram - Example

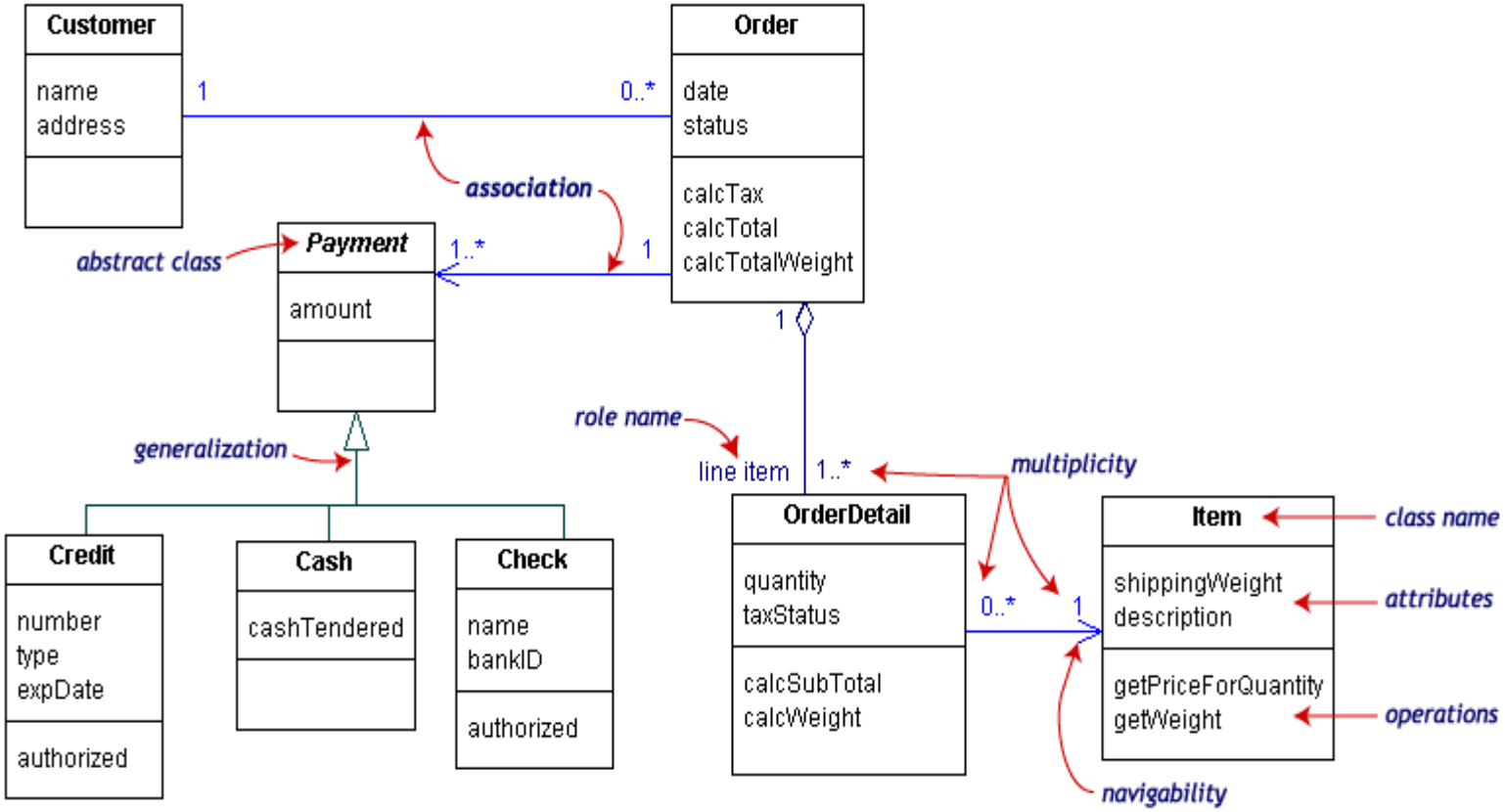


Class Diagram - Example

Draw a **class diagram**, showing the relevant classes, attributes, operations and relationships, for each of the following situations. Include classes, associations, multiplicity symbols etc., as appropriate, in your diagram.

- A customer may place orders which consist of one or more orderlines.
- Each orderline is for exactly one item and an item may appear on one or more orderlines.
- Payment for an order is made either by cash, cheque or credit card.
- The system needs to record a customer's name and address, an order's date and status, an item's description and price, and the quantity ordered of each item.
- The system must also record the amount paid; for credit cards – the credit card type, and expiry date; for cash – the amount tendered; and for cheques – the account number and the bank details.

Class Diagram – Example

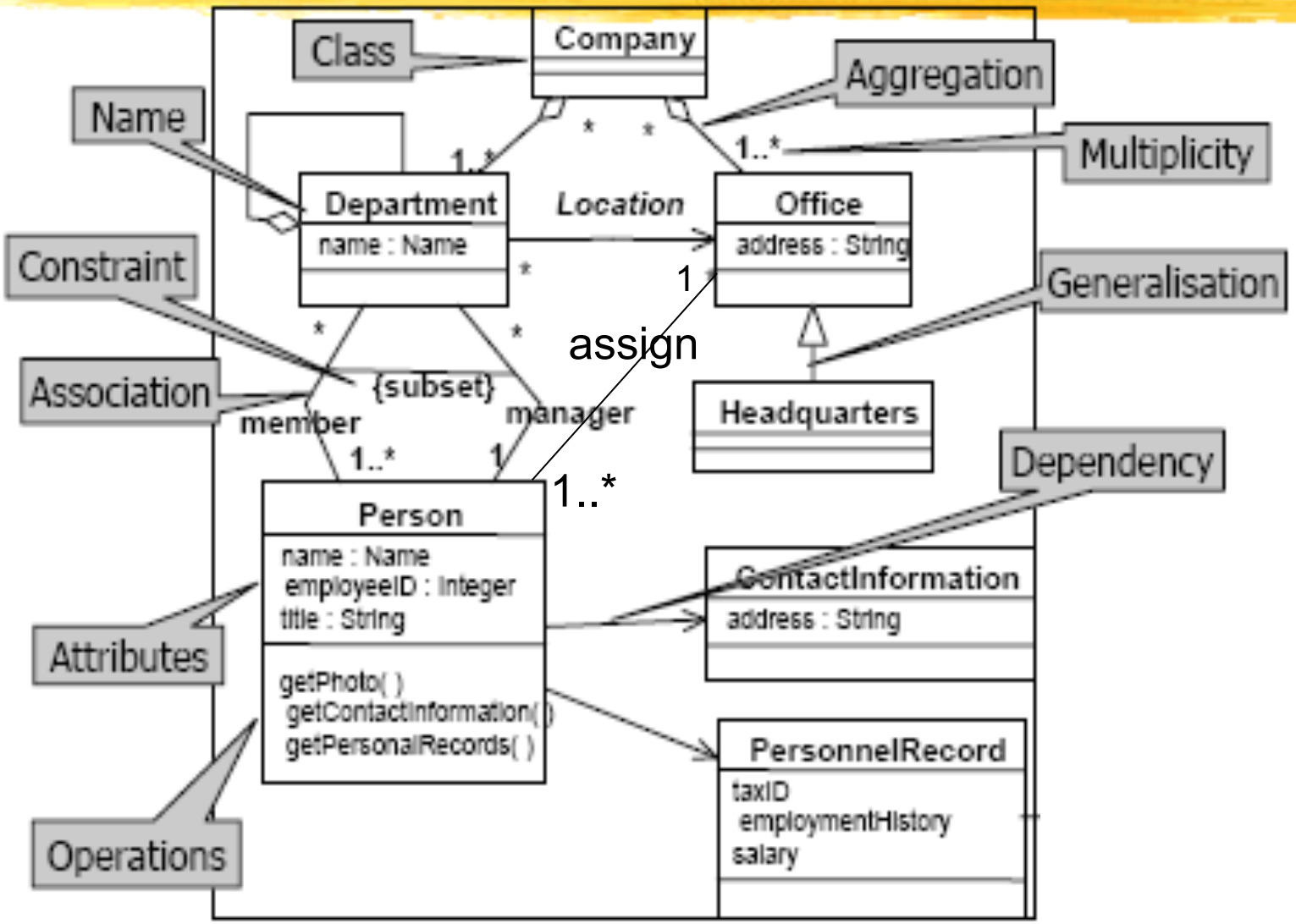


Class Diagram – Example

Draw a **class diagram** for the following specification. Include classes, associations, multiplicity symbols etc., as appropriate, in your diagram.

- A company has many departments. It also has many offices. One office is designated as the headquarters.
- A department has many employees and each is assigned to a particular office. Each department has one manager and each manager manages many departments.
- Each person has a personnel record associated with them and a record of their contact details.
- Add some appropriate attributes to each class.

Class Diagram - Example



Summary

- ✓ Introduction
  - Structural modeling
- ✓ Classes
  - Attributes and operations
- ✓ Relations
  - Associations
  - Dependencies, compositions
- ✓ Generalization
  - Inheritance