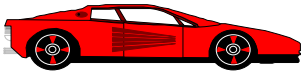
CHAPTER 4: UML CLASS DIAGRAMS

Software Engineering

Computer Science School

Introduction



Real World

House

Person

Car

Semantic Gap

(House)

Lives in

(Person)

drives

(Car)

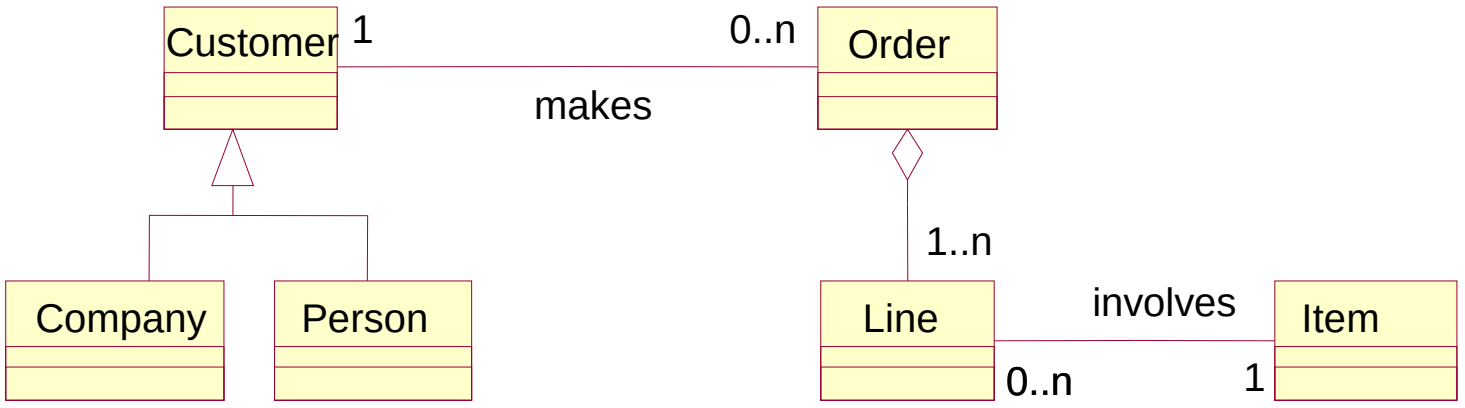
(Equation)

(Binary Tree)

Model

Introduction

- ***Class diagrams*** reflect the **static structure** of the system.
- It is the **main tool** of most OO methods.
- They contain **interrelated classes** , by means of associations , organized as aggregation and generalization and specialization hierarchies.



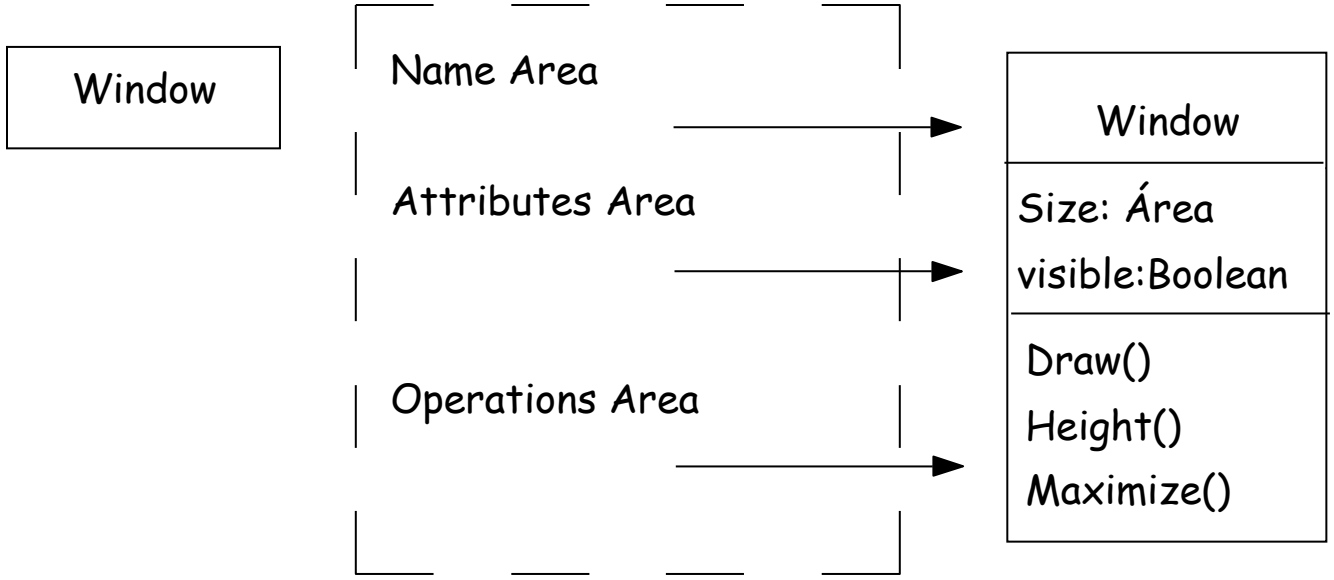
Introduction

- An objects is a concept, abstraction, or thing that makes sense within the context of an application. It is an encapsulation of data and operations.

- Objects appear as names within the description of the problem or in discussions with users.

Classes

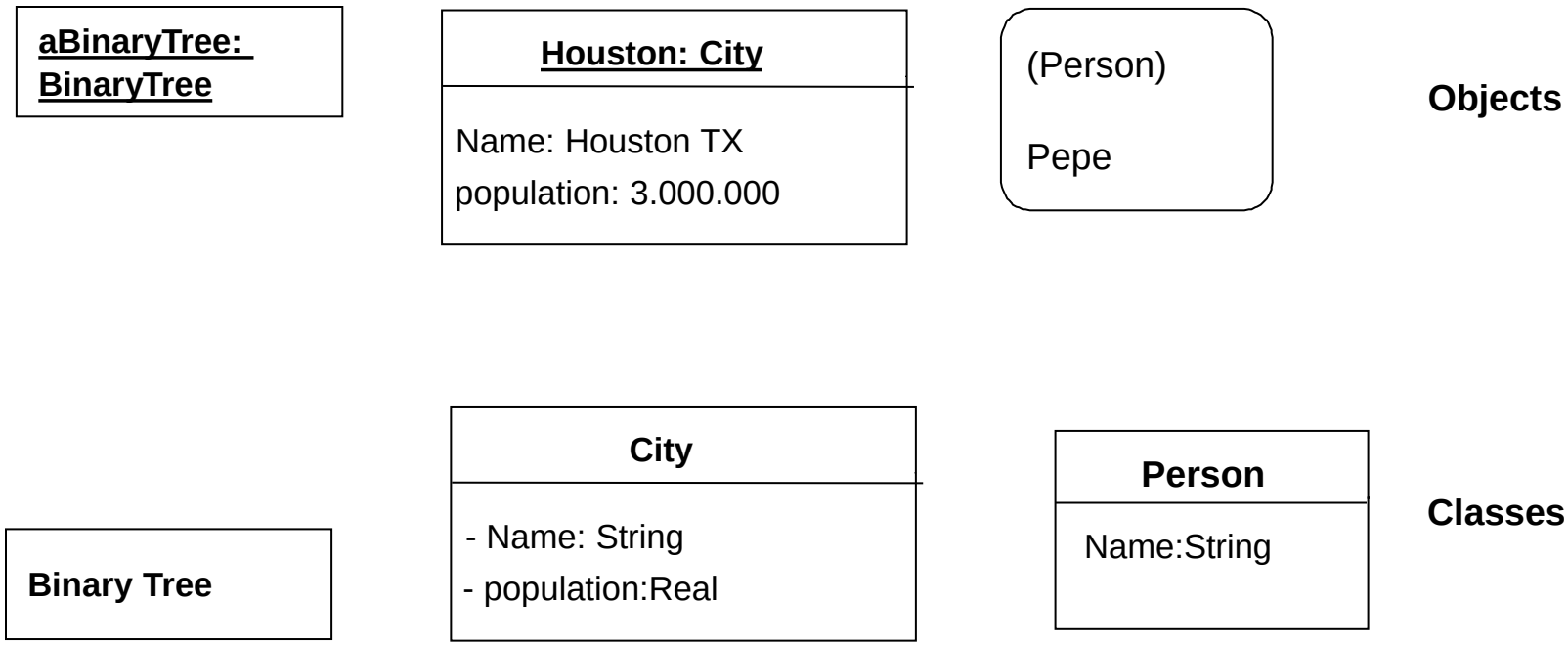
- A class is the description of a group of objects with similar structure, behavior and relationships.



- An additional area may be added to define responsibilities, description, etc.

Classes

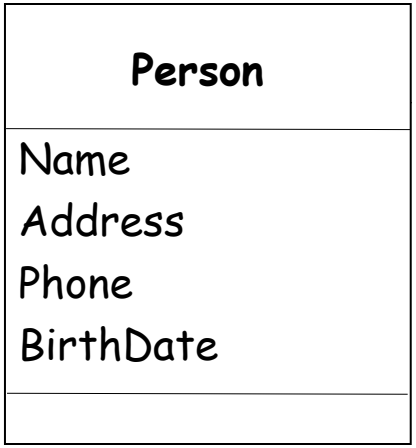
- A class is an abstraction; an object is a concrete realization of this abstraction.



Attributes

- An attribute is a property of a class identified with a name and describes a range of values.

- Attributes may be represented showing only their names.



- The general definition schema is:

  [visibility] Name [: Type] [=initial value]

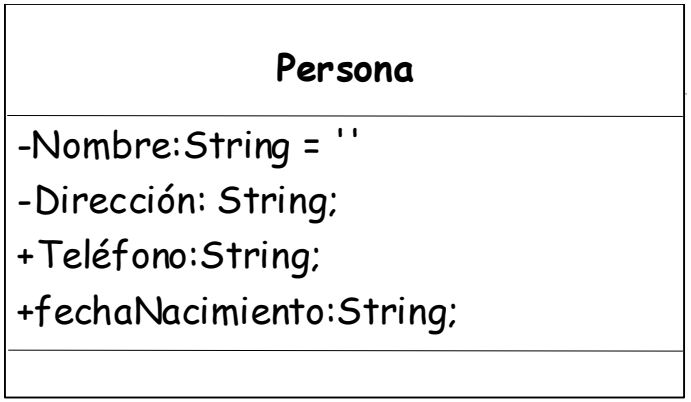
  where visibility may be:

  + = Public

  # = protected

  - = Private (default)

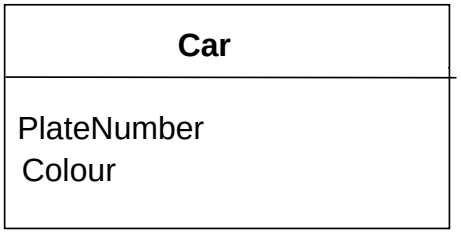
  = implementation or package



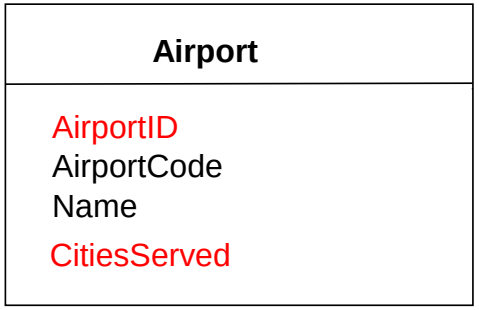
- Allowed types are (integer, real, char, string, etc.), no object types.

- Attributes do not include references to other objects, these references are represented as links.

- In the objects model attributes acting as objects identifiers should not be present



Good for analysis and design

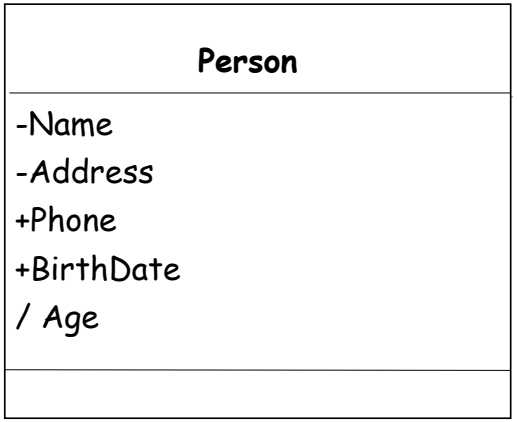


Bad analysis, good for design

- An attribute is derived if its value is obtained in terms of the values of other attributes.

- The notation for derived attributes is:

/Attribute_Name: Type



Operations

- An operation corresponds to a service that may be required to any object of the class.
- An operation is a function or transformation that may be applied to objects.
- A method is the implementation of an operation.

- Operations are defined as follows:

[visibility] Name([parameters]) [: return_type]

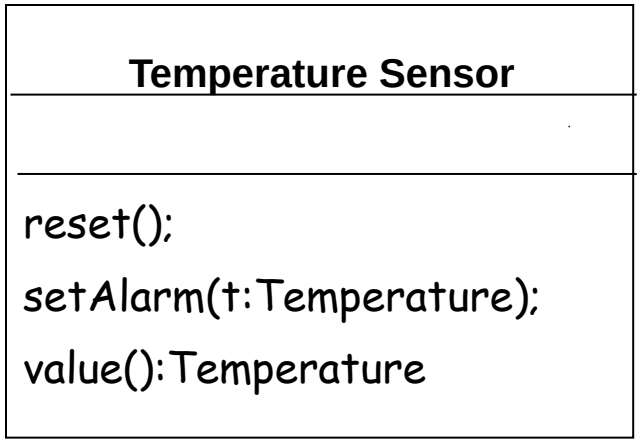
Where visibility is:

+ = public (default)

#= protected

- = private

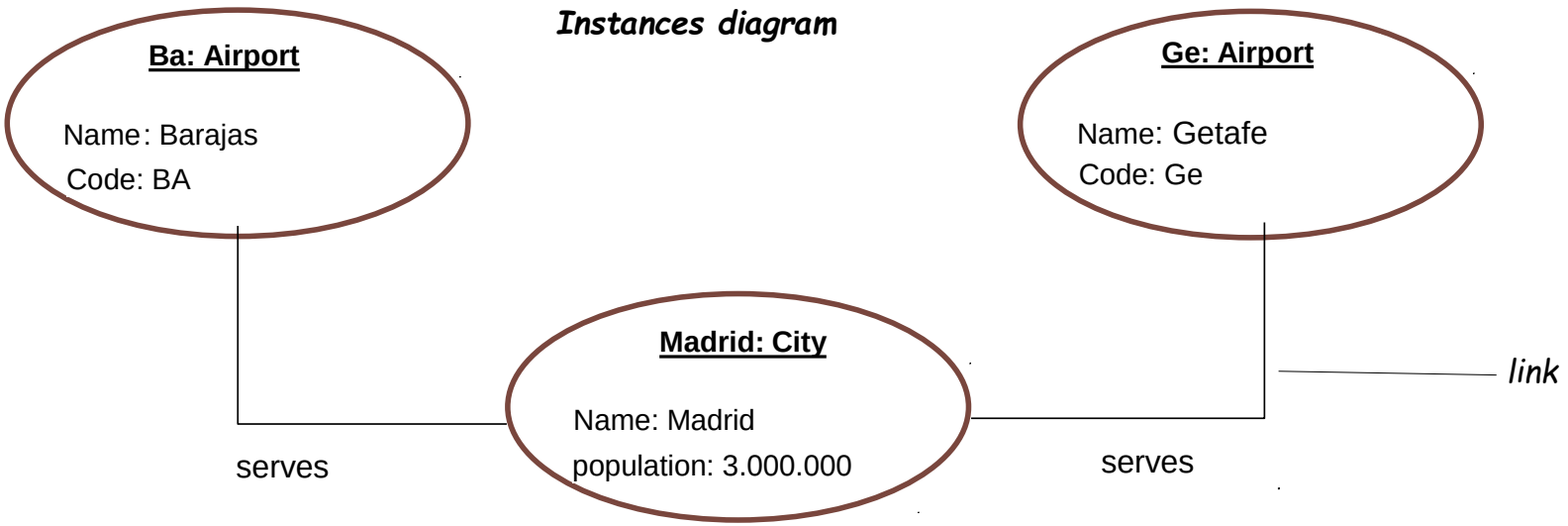
= package

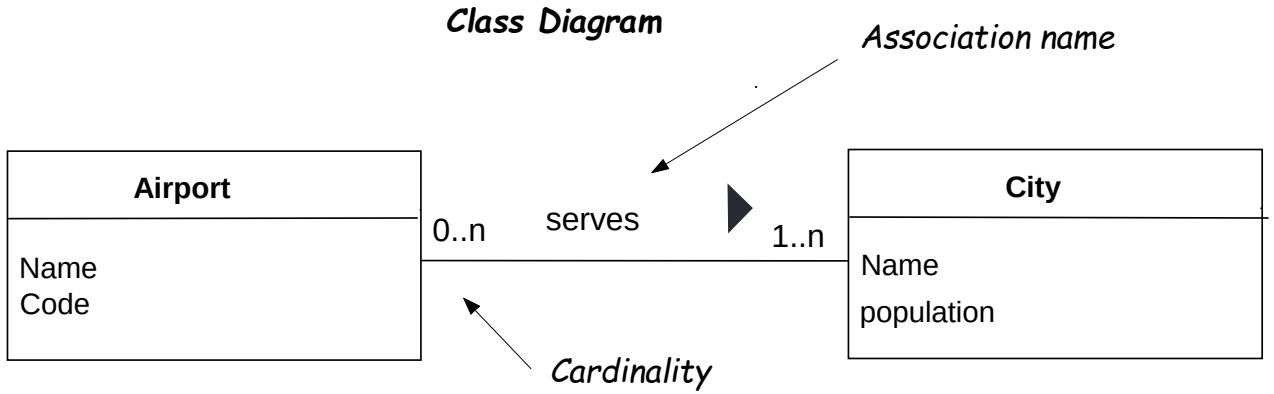


- Operations that change the state of objects are defined as having side effects.
- Operations that do not have side effects and just calculate a functional value are called queries.
- Queries return the values of attributes.

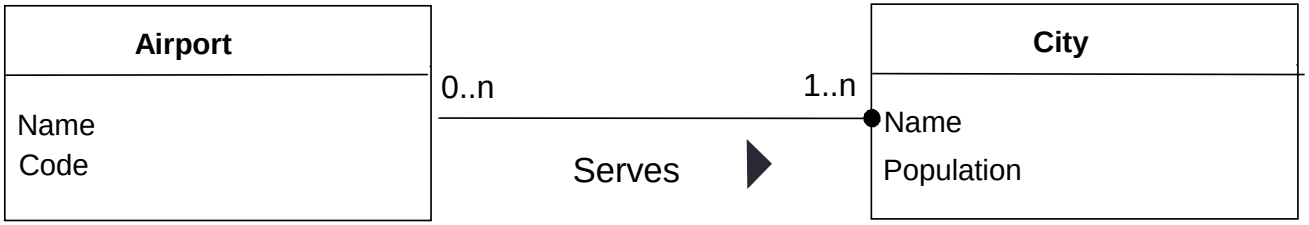
Associations & links

- A link is a physical or conceptual connection between objects.
- An association is a structural relationship to show that the objects of an element are linked to the objects of another element.
- Associations are represented in class diagrams whereas links appear in instances diagrams.

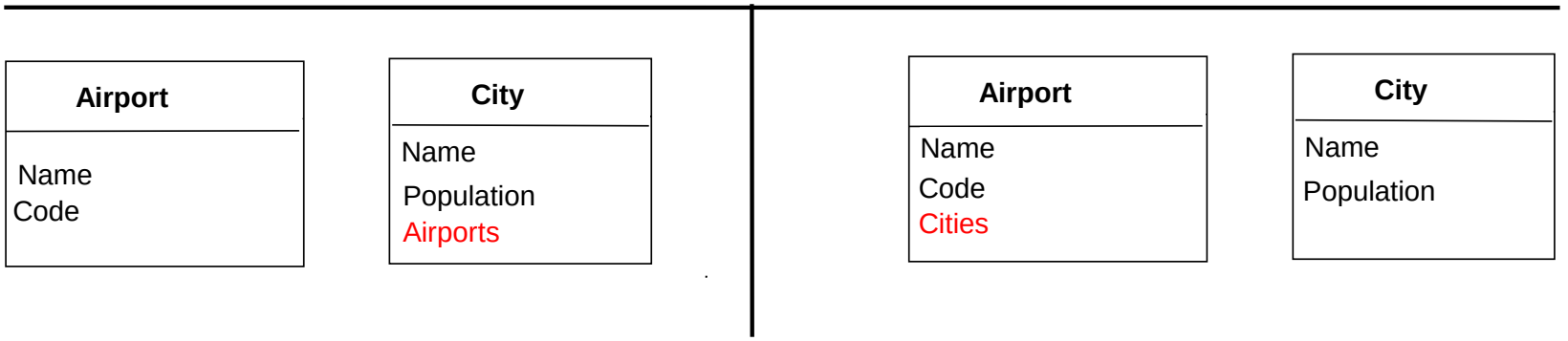

Instances diagram
Ba: Airport
Name: Barajas
Code: BA
Ge: Airport
Name: Getafe
Code: Ge
Madrid: City
Name: Madrid
population: 3.000.000
link
serves
serves


Class Diagram
Association name
Airport
Name
Code
0..n
serves
1..n
City
Name
population
Cardinality

- Each association in a class diagram corresponds to a collection of links in the instances diagram.
- Given an association between two classes, it may be navigated in both directions.
- Binary associations are those connecting two classes.

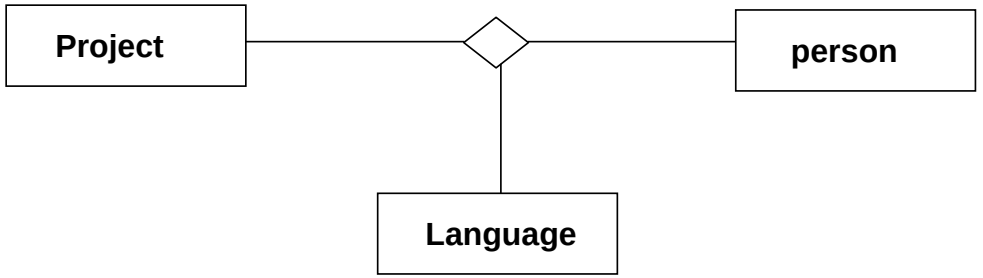


Analysis Model

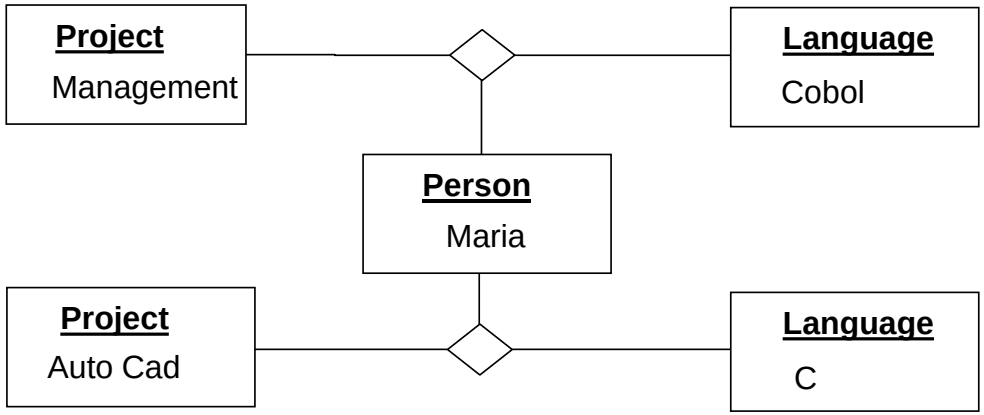


Design Models

- Associations may be of any order (2, 3, …).



Associations

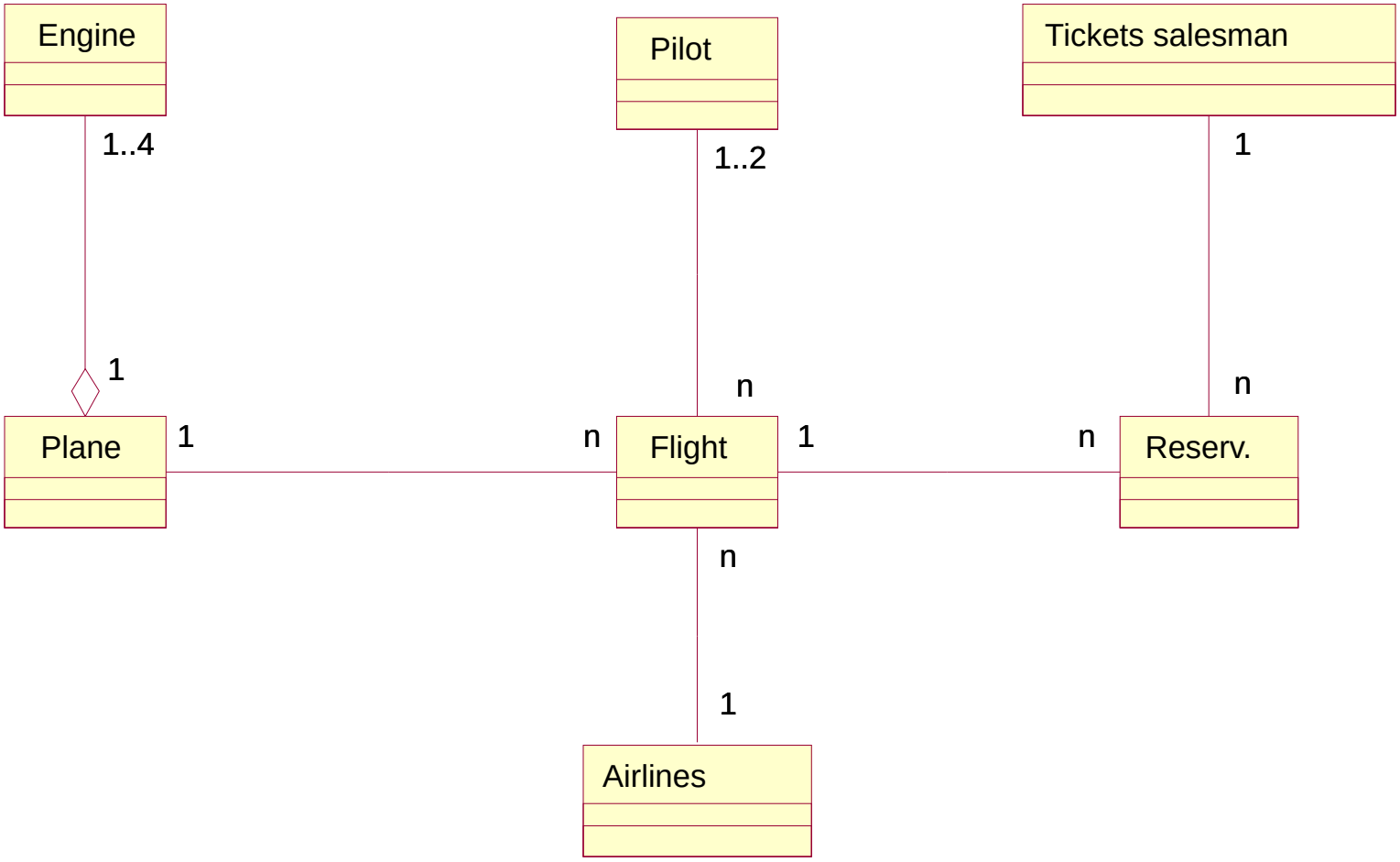


links

Cardinality

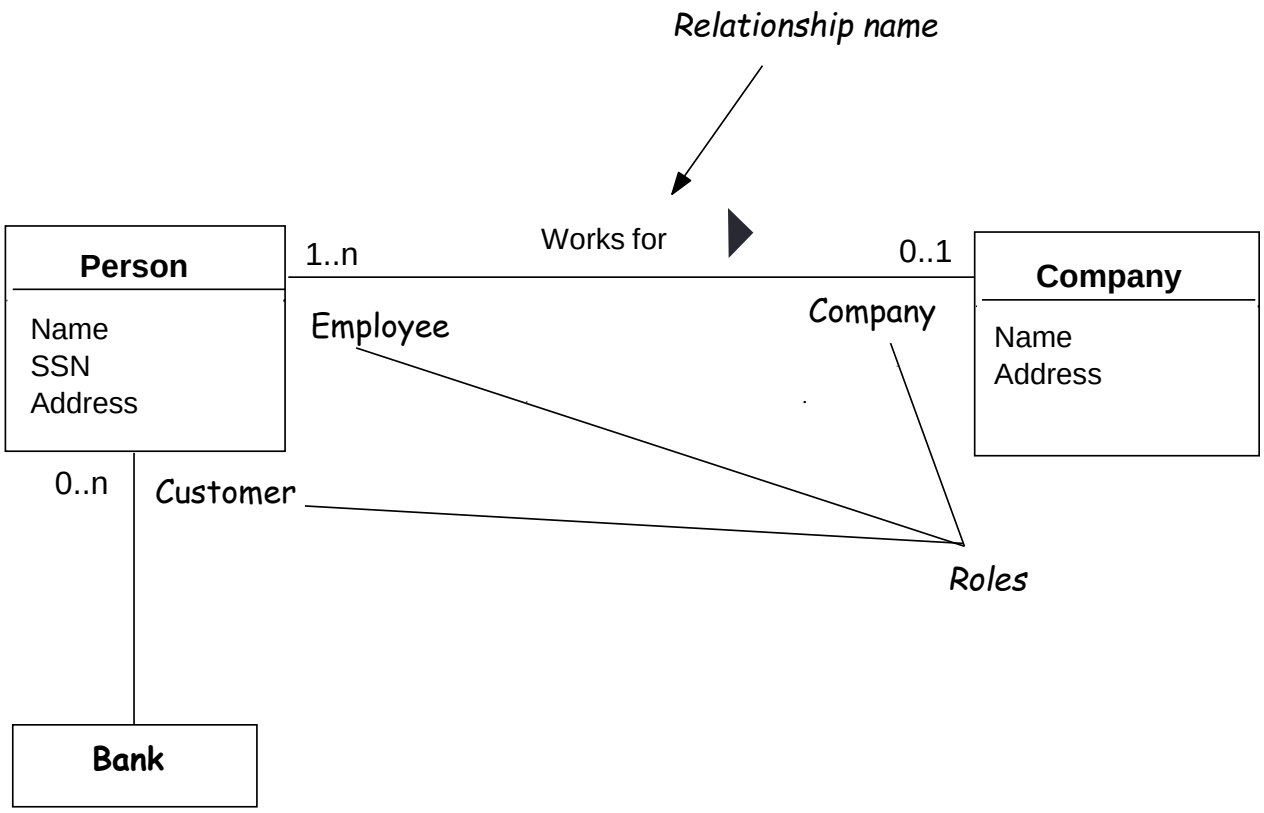
- The cardinality defines the number of instances of another class that may be related to an instance of a given class.

1	One and only one
5	Exactly five
0..1	zero or one
M..N	from M to N
*	from 0 to many
n	from 0 to many
0..*, 0..n	from 0 to many
1..*, 1..n	from 1 to many


Engine
Pilot
Tickets salesman
1..4
1..2
1
1
n
n
Plane
1
n
Flight
1
n
Reserv.
n
1
Airlines

Roles

- Roles are names that define the role played by a class in a relationship.

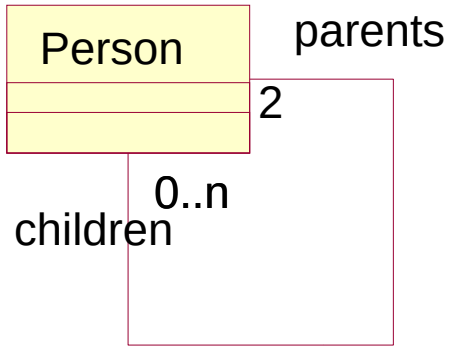


- Roles are used to navigate associations.

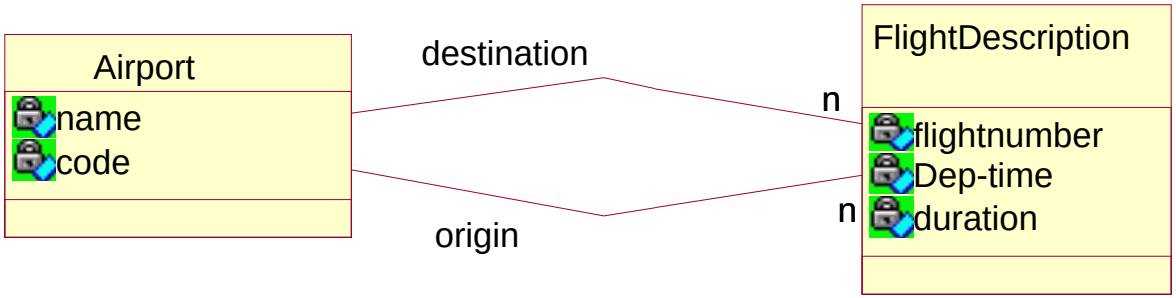
  `aCompany.employee`

  `aperson.company`

- They are mandatory to distinguish reflexive associations.

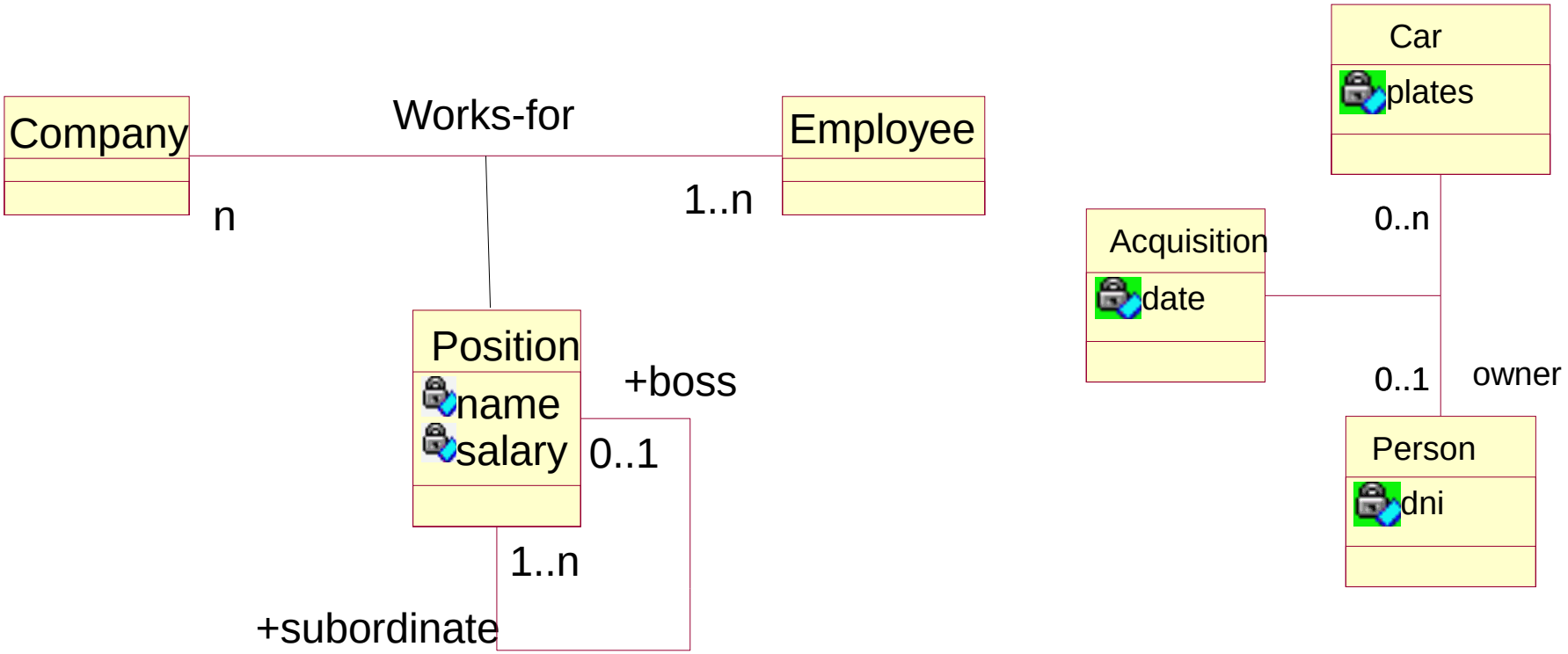


- They are mandatory to distinguish associations between same pair of classes.



Associations as classes/ Link Classes

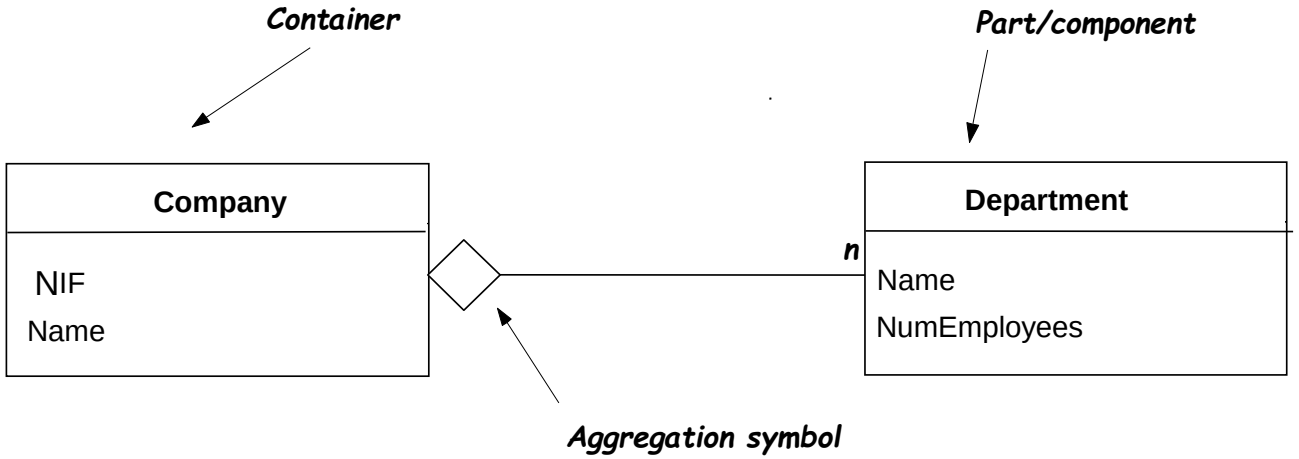
- In an association between two classes the relationship itself may have attributes. These are represented as link classes. If there is no associated class name then it is a Link Attribute



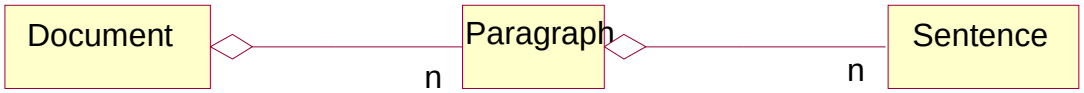
Aggregation

- An aggregation is a type of relationship with additional semantics.
- This relationship is used to model the semantincs “ **part_of consists_of** ”.

"A company consists of departments"

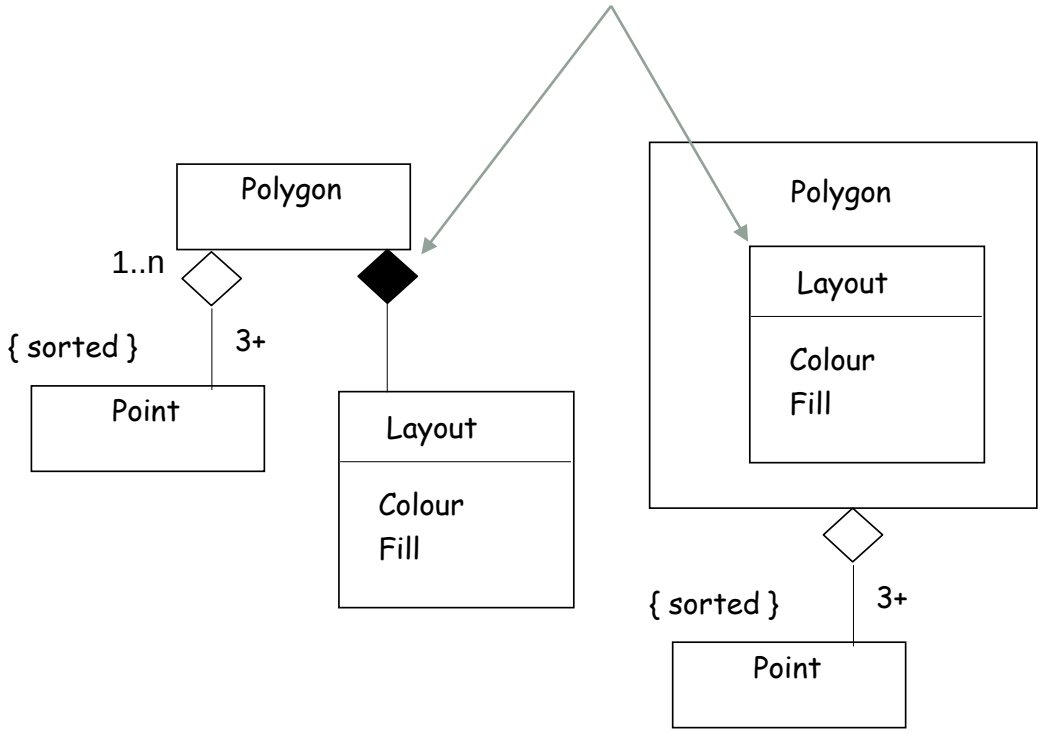


- The properties of the aggregation are:
  - **transitive** (if A is part of B and B is part of C then A is part of C)
  - **antisymmetric** (if A is part of B then B may not be part of A).

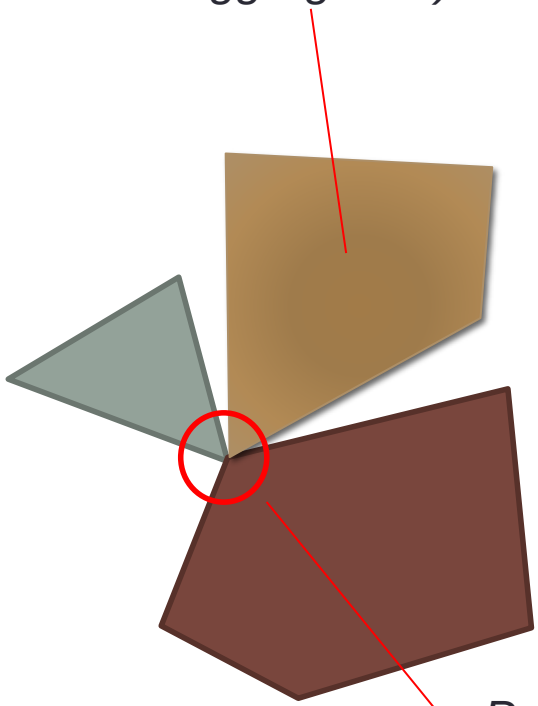


- There are two types of aggregations.
  - ***Inclusive or physical:*** each component may belong at most to one container. The destruction of the container implies the destruction of its parts.
  - ***Referential or catalogue:*** components may belong to several containers. Their lifetimes are not correlated.

Composition (Inclusive)

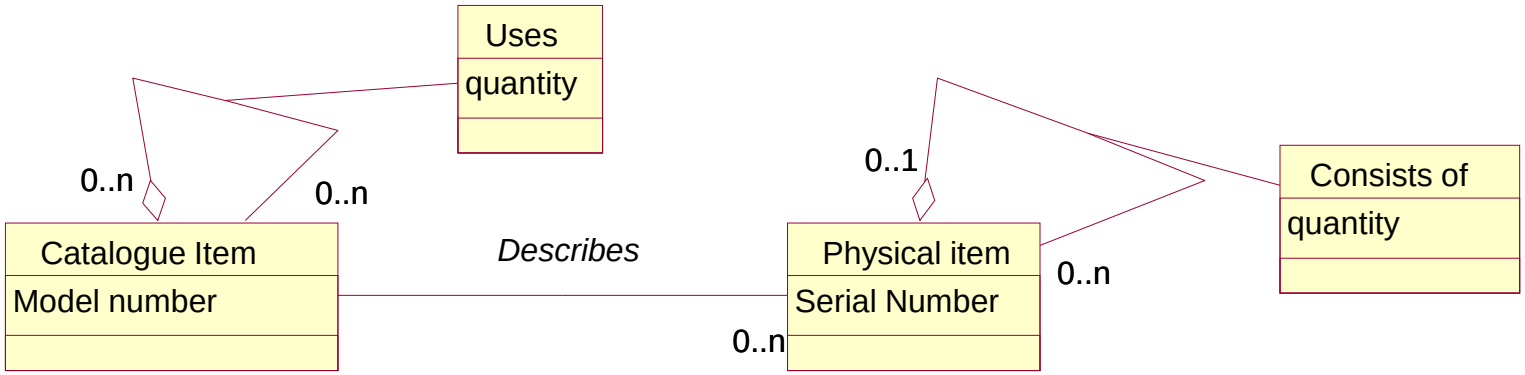


Layout not shared by polygons (inclusive aggregation)



Point belongs to many Polygons (referential aggregation)
If polygon destroyed a point could remain

Homework: referential or inclusive?



Generalization / Specialization

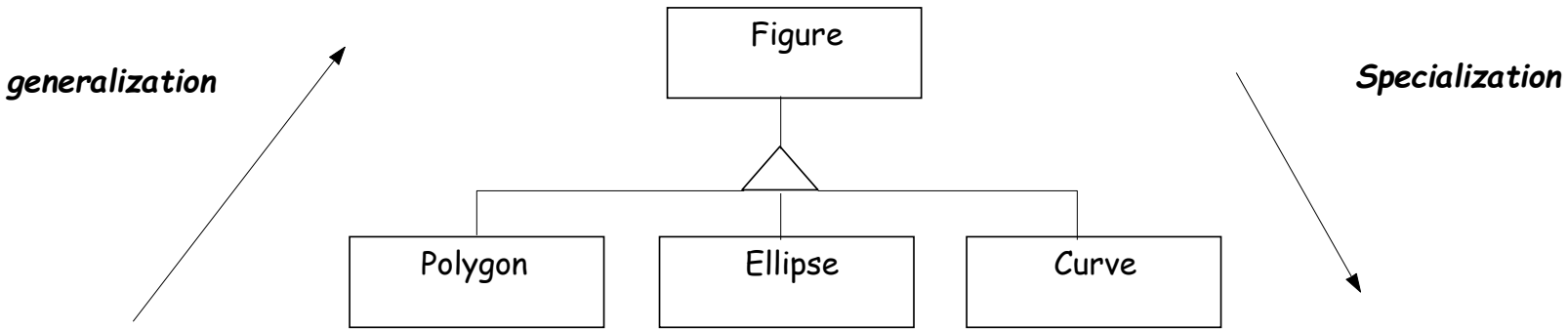
- Hierarchies of classes allow the management of the complexity in terms of taxonomic classifications.
- Starting from classes that have in common a number os attributes and operations, using generalization, a more generic class (super class) can be obtained from these initial classes (subclasses).
- Shared attributes and operations are placed in the superclass

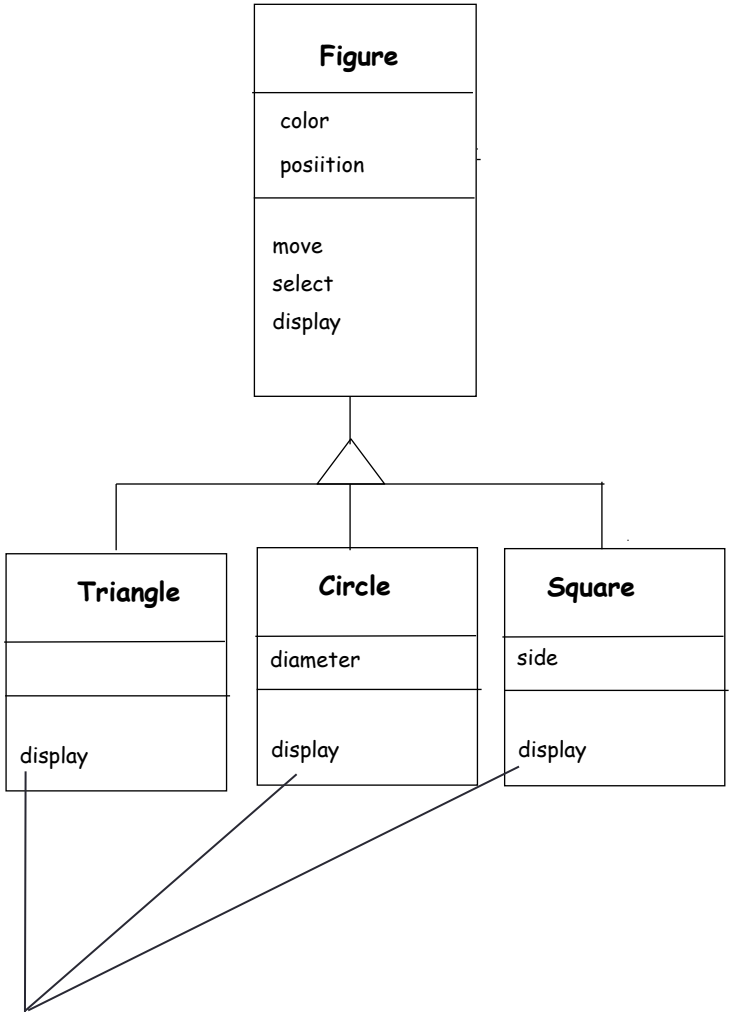
Generalization / Specialization

- The specialization is the opposite relationship. Starting from a superclass the subclasses are obtained.

- Subclasses inherit attributes and operations defined in the superclass.

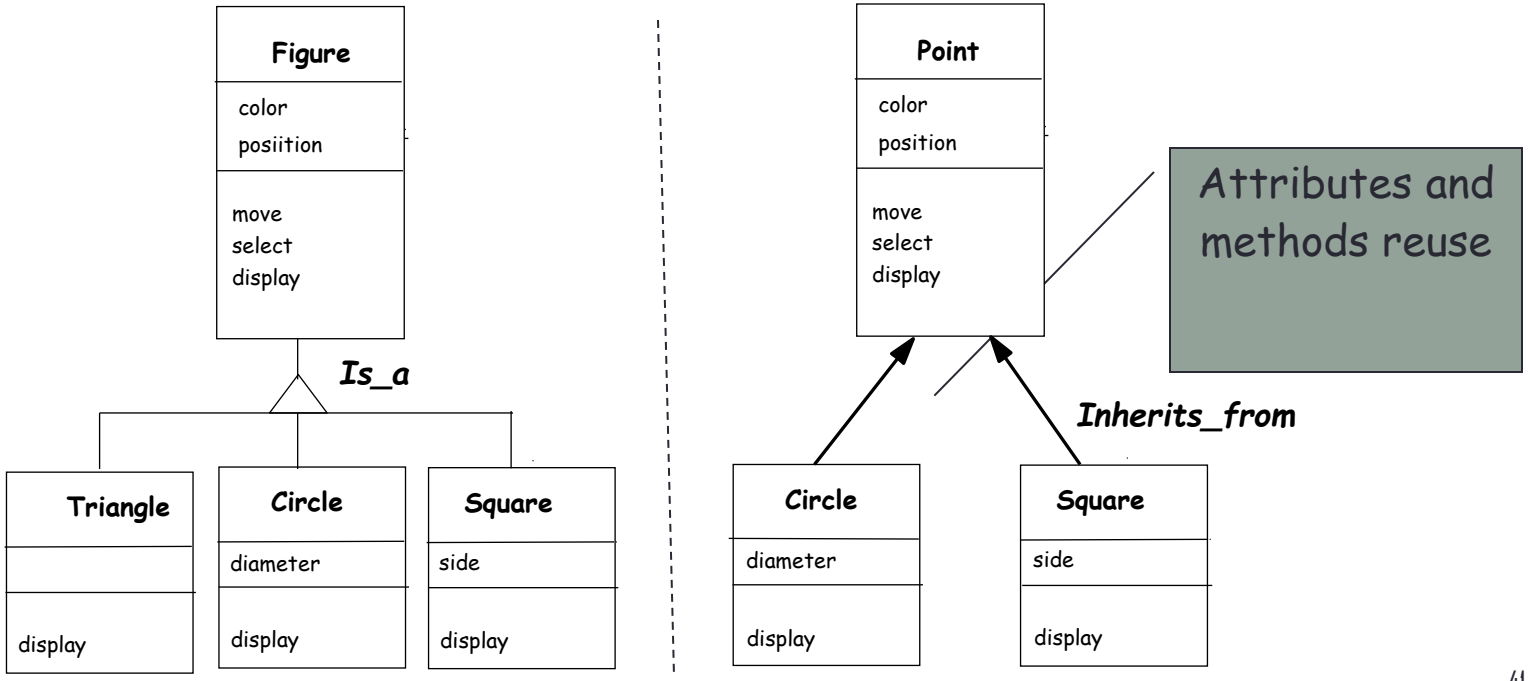
- Each instance in a subclass is also an instance of the parent class. (relationship **is_a**).



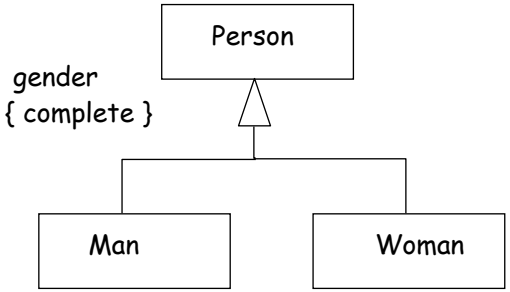


operation redefinition

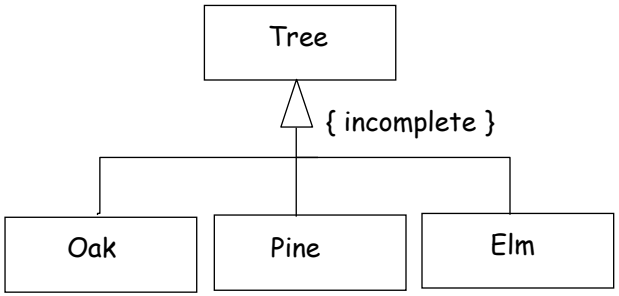
- The specialization relationship is used for conceptual modelling whereas inheritance is a code reusing mechanism during the implementation phase.



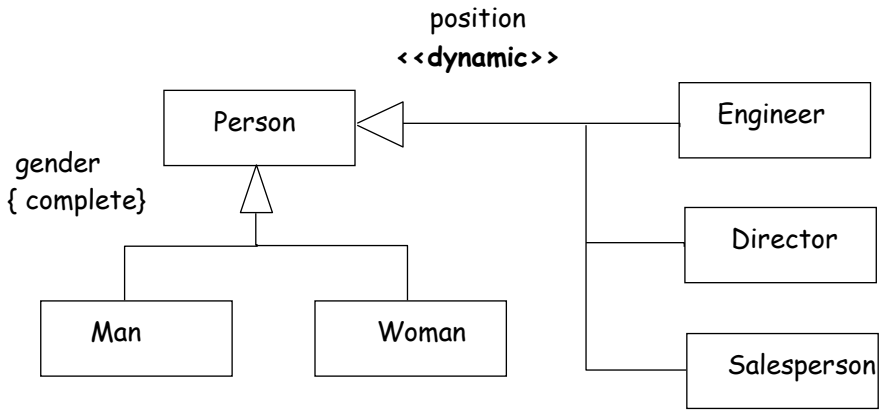
- Two types of **restrictions**:
  - **Complete**: All children classes are specified in the model



  - **Incomplete**: Not all children specified



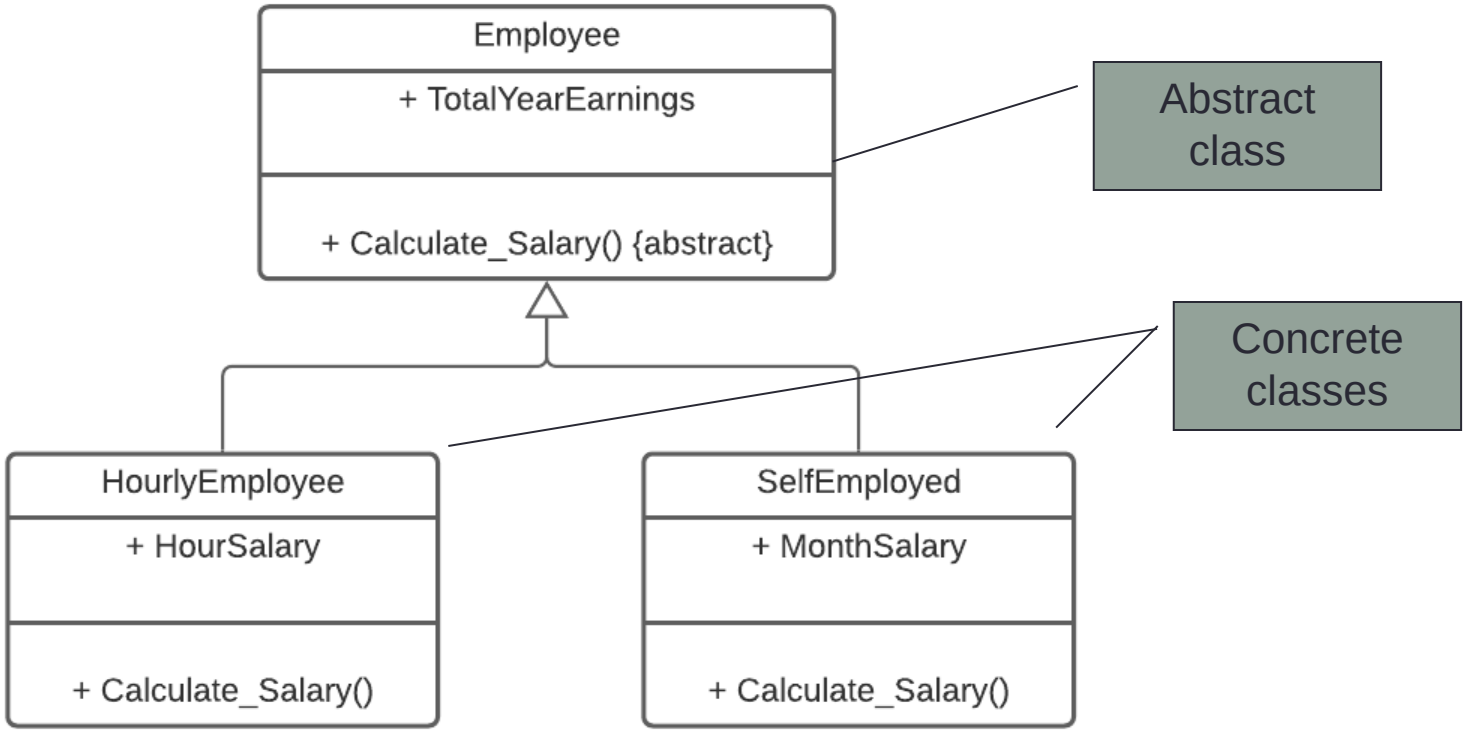
- A specialization is dynamic if an object may change to a different class within the hierarchy.



Abstract Classes

- An abstract class has no instances. Its descendant concrete classes have them.
- An abstract class has at least one method without code (abstract method)
- Abstract classes are used to defined the operations that are inherited by the descendant classes. They provide the protocol (interface) but without giving a concrete implementation.

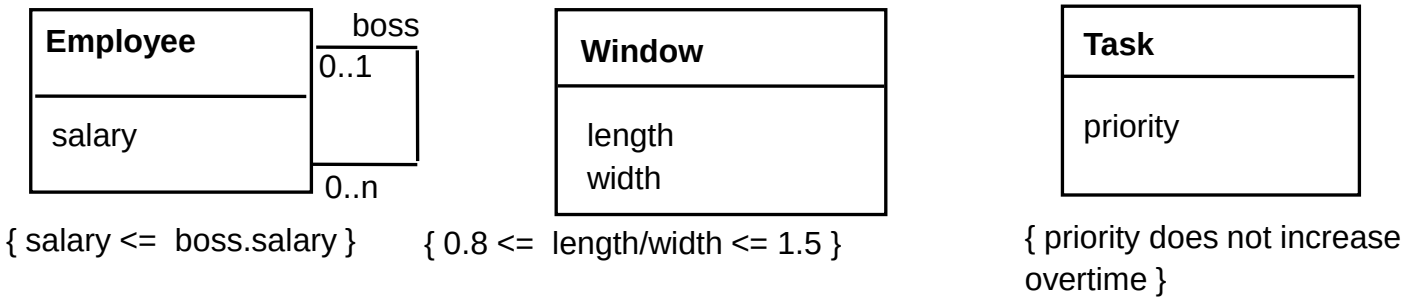
- All concrete subclasses must provide an implementation for an abstract method.



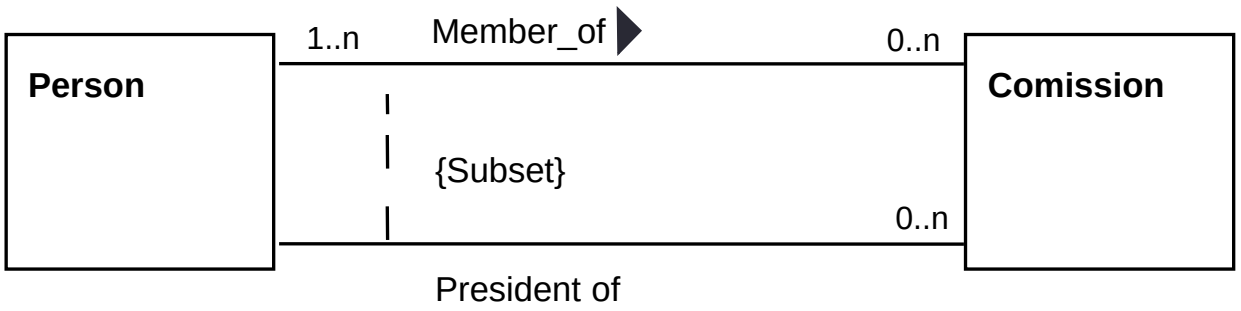
- Homework: What is the difference between an abstract class and an interface?

Restrictions

- Restrictions are functional relationships between entities in the model.
- Usually expressed in a declarative form but also natural language can be used
- These restrictions may refer to values of the attributes of an object.



- They are also used between association relationships.



Exclusive associations

- An axclusive association (or-association) consists of set of associations that relate an initial class (source) with several destination classes.

- Taking an object of the source class, it is at most related with one object of a destination class.