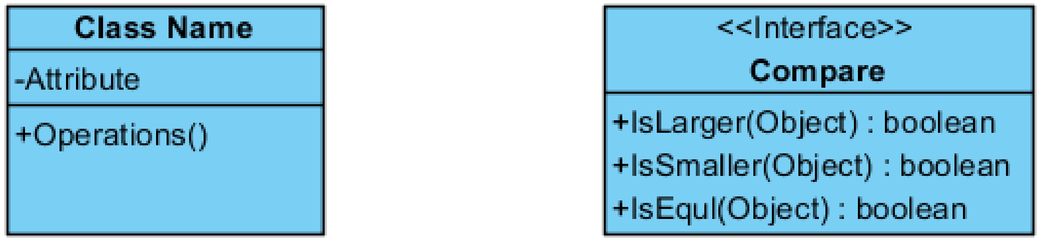
Information system design
Seminar 4

UML Class Diagram

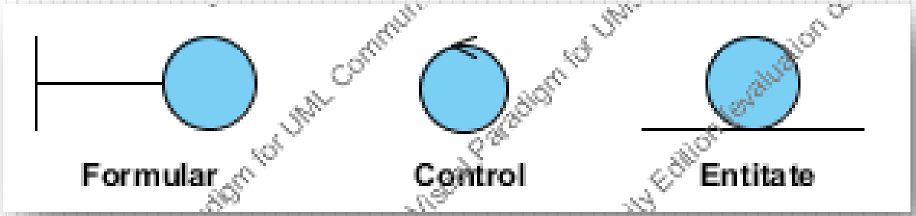
Defining a class

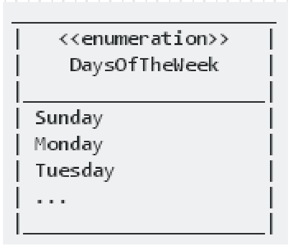
- Set of objects that have the same characteristics and constraints.
- The characteristics of a class are attributes and operations.
- Abstract classes can not be instantiated. Their role is to enable other classes to inherit them, for reuse of characteristics.
- An interface describes a set of characteristics and public obligations. Actually, it specifies a contract. Any instance that implements the interface must provide the services provided by the contract.


Class Name
-Attribute
+Operations()
<<Interface>>
Compare
+IsLarger(Object) : boolean
+IsSmaller(Object) : boolean
+IsEqul(Object) : boolean

Examples of common stereotype classes

- <<entity>> – a passive class, which does not initiate interactions;
- <<control>> – initiates interactions, contains a transactional components and acts as separator between the entities and limits;
- <<boundary>> – it is located an the periphery of the system, but inside. It’s the contact element to the actor or to other systems.
- <<enumeration>> - it is used to define data types whose values are listed.
- <<primitive>> - a form of class that represents predefined data types, such as Boolean.





```
<<enumeration>>
DaysOfTheWeek

Sunday
Monday
Tuesday
...
```

Attributes -1

- Each attribute is described at least by its name.
- Additional information can be added, and the general structure of an attribute is:
- [visibility][/]name[:type][multiplicity][=default value] [{property}]
- Visibility may be:
  - + Public: can be viewed and used by anyone
  - - Private: only the class itself has access
  - # Protected: the class and subclasses have access
  - ~ Package: only classes in the same package have access
- / symbolizes a derived attribute

Attributes -2

- UML allows specification of multiplicity for attributes ,when you want to define more than one value for an attribute. They have the following meanings

Multiplicity	Meaning
1	Exactly 1 (default)
2	Exactly 2
1..4	From 1 to 4 (inclusive)
3, 5	3 or 5
1..*	At least one or more
*	Unlimited (including 0)
0..1	0 or 1

- Property indicates an additional property that applies to the attribute:
    - {readonly}: the attribute can be read but not modified
    - {ordered}, {unordered}: an ordered or unordered set
    - {unique}, {nonunique}: the set of values may or may not contain identical items

Operations

- The general form of an operation is:
- [visibility] name ([direction] parameter list) [:returned type] [{property}]
- Visibility - the same as in class
- Direction - 'in' | 'out' | 'inout' | 'return'
- Return type - whether they return something, if it’s a function
- An example of an operation’s property: {query} – it does not change the stat of an object or other objects

Attribute examples:

- - age: Integer {age>18}
- # name:String[1..2]=“Ioana”
- ~ Id:String {unique}
- / TotalValues:Real=0

Operations examples :

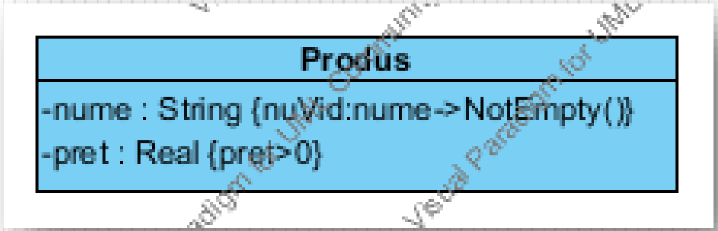
- + setAge (out Age: Integer)
- + getAge(in Id:String): Integer {query}
- - changeName(inout Name:String)

Constraints

- A constraint is an expression that restricts a certain element of a class diagrams.
- This may be a formal expression (written in Object Constraint Language - OCL) or it can have a semi-formal or informal format.
- They are represented in braces.
- They can be written immediately after defining an element or as a comment.
- A constraint can have a name, as follows: :
- **{name : Boolean expression}**

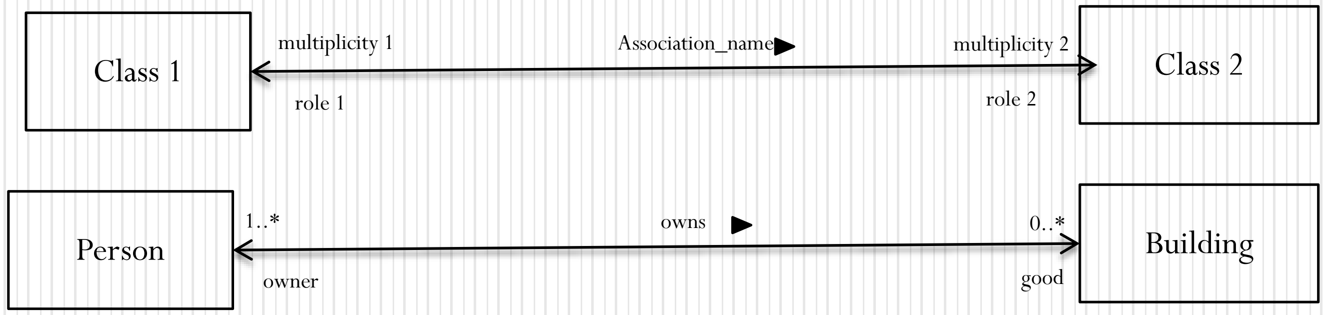
Examples of OCL constraints:

context Organisation
  inv: self. departaments→isUnique (name)
  inv: departaments.employees→isUnique (code)



Relationships between classes - 1

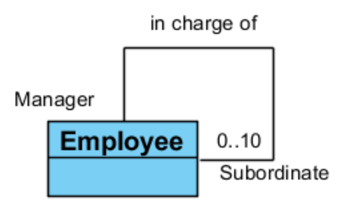
1. *An association* implies establishing a relationship between classes.It is characterized by:
   - name (optional)
   - multiplicities – specified at both ends of the association
   - roles of the association: specified at each end of the association and contain a brief and representative description (1-2 nouns)
   - navigation direction



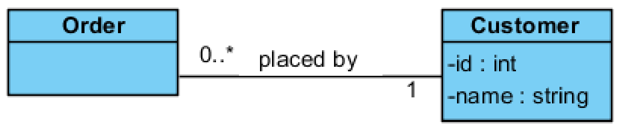
Relationships between classes -2

Types of associations :

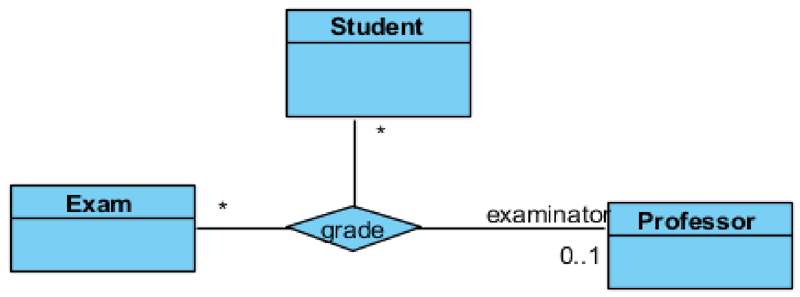
- Unary: connects a class to itself.



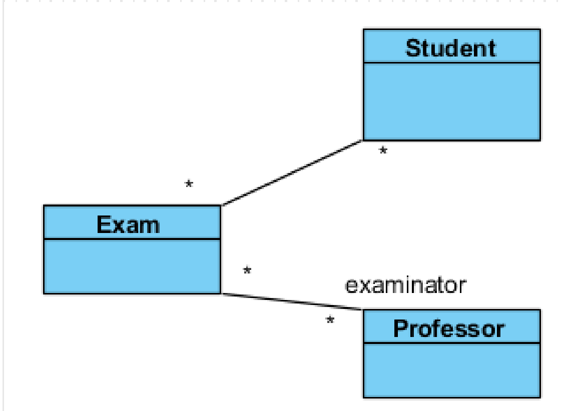
- Binary: between two classes.



- Ternary: they are usually transformed in binary association.

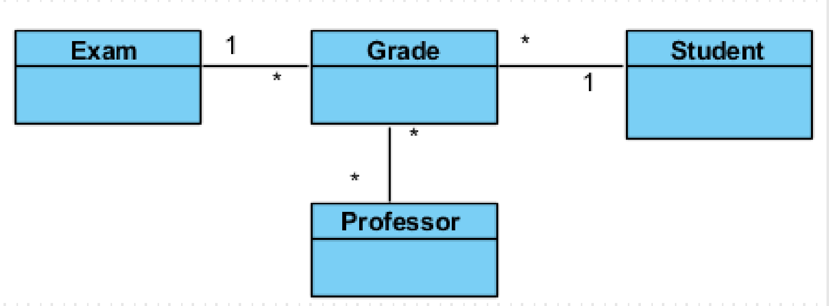


N-ary association



Varianta 2: This transformation created a model with a different meaning.

- ✓ First, in this representation, an exam can be scored by several teachers.
- ✓ Secondly, it is not clear which teacher evaluated a particular student in an exam.

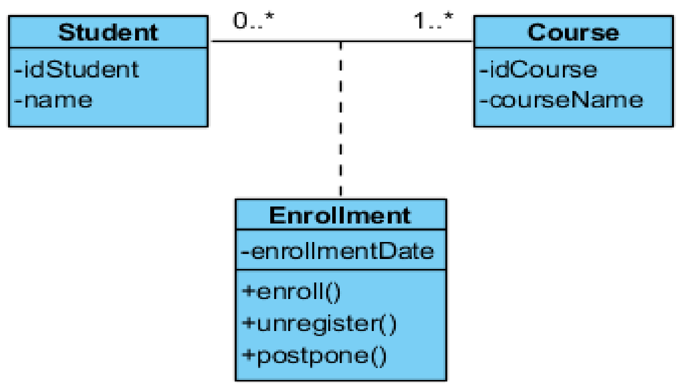


Varianta 3: An additional class was added, Grade.

✓ this model allows a student to be graded several times for the same exam, which is not possible with the ternary association model

Relationships between classes -3

- The association modeled as a class allows the relationship to have attributes and operations.

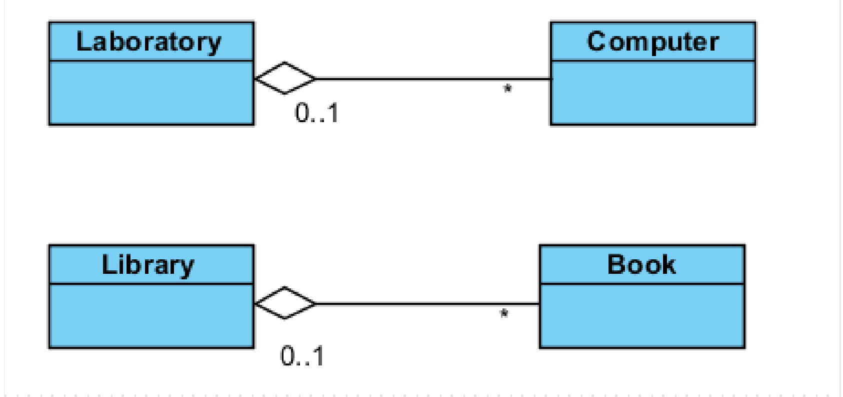


2. *The Aggregation relationship* is a binary form of association representing a ‘part – of ‘ type relationship .

- It can be of two types:
  - Shared aggregation (aggregation)
  - Composed aggregation (composition / notn-shared association)

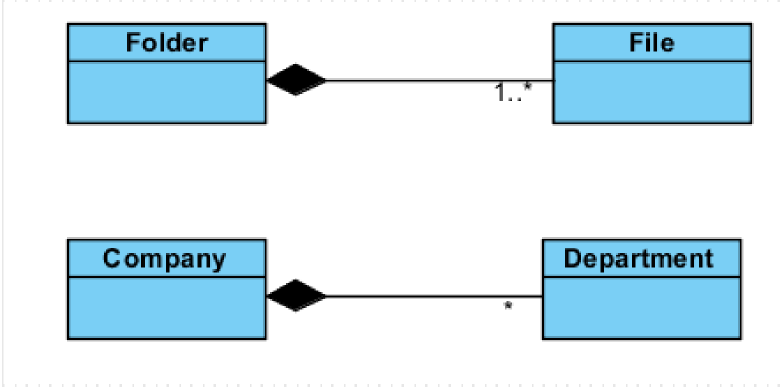
Relationships between classes -4

- *Shared aggregation* is a weak form of aggregation in which the instances of the parts are independent from the whole, as follows:
  - The same elements can be aggregated by several other 'whole' classes
  - If you delete the class that aggregates, the aggregated classes will continue to exist.
  - It is represented using the shape of a diamond at the end of the association corresponding to the aggregating class.

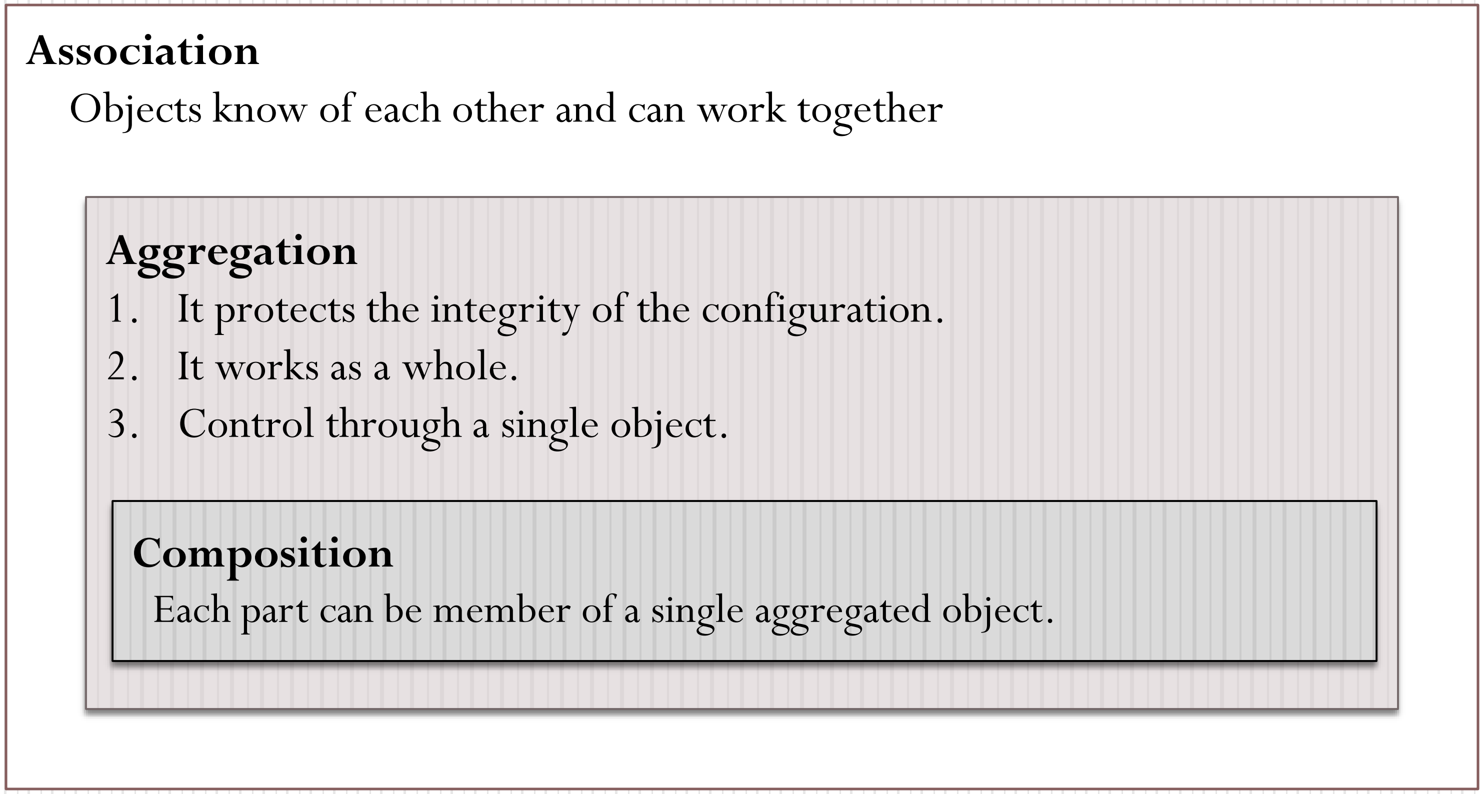


Relationships between classes -5

- *Composed aggregation* is a powerful of aggregation form in which the instances of the parts are independent from the whole, as follows: :
  - If the aggregating class is deleted, the aggregated classes will also be deleted.
  - It is represented using the shape of a full diamond at the end of the association corresponding to the aggregating class.
  - When used for modeling objects in a certain domain, the deletion may be figuratively interpreted as an "end" and not as a physical destruction.



Relationships between classes -6



Association

Objects know of each other and can work together

Aggregation

1. It protects the integrity of the configuration.
2. It works as a whole.
3. Control through a single object.

Composition

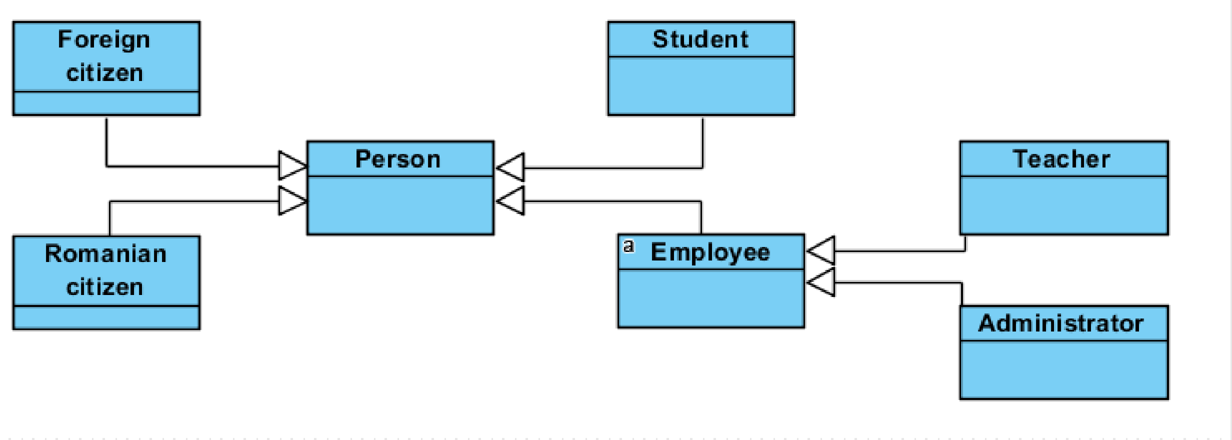
Each part can be member of a single aggregated object.

Relationsips between association, aggregation and composition

Relationships between classes -7

3 The generalization relationship is used to indicate inheritance between a general class (superclass) and a specific class (subclass)

- Also called informally 'is a type of' relationship.
- It is represented as an empty triangle placed at the end of the superclass.
- Subclasses inherit the characteristics and constraints of the superclass.
- Multiple inheritance is allowed.



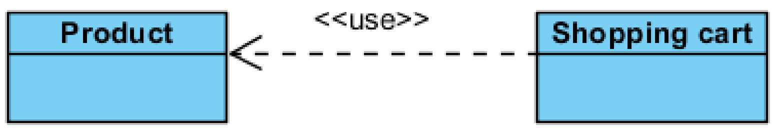
Relationships between classes -8

4. The dependency relationship is used to show a wide range of dependencies between elements of a model

- In the analysis stage, the dependency type can be unspecified.
- At the design stage, dependencies will be personalized with stereotypes or will be replaced with connectors specific to the technology used.
- It is represented as a dotted line from the dependent class "client" to the "supplier" class, with an arrow at the end of "supplier" class .
- In class diagrams, the most important dependencies are ‘use’ and ‘abstraction’

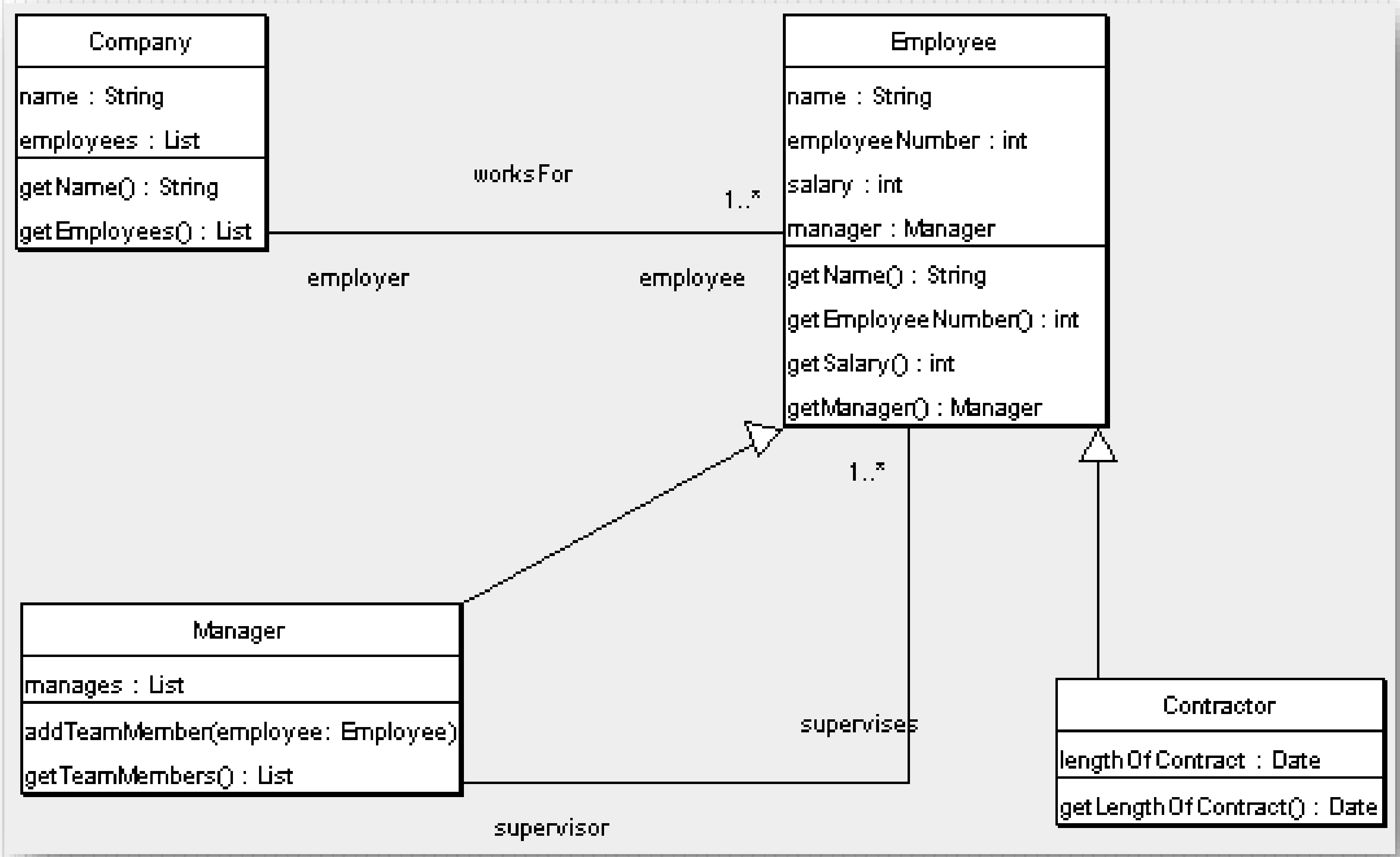
Relationships between classes -9

- Use dependency (<<use>>, <<create>>, <<call>> etc.) is a relationship in which a 'client' class needs another class or group of classes (supplier) to operate
- An abstraction dependency relates two elements or set of elements (called customer and supplier), representing the same concept, but at different levels of abstraction or seen from different views



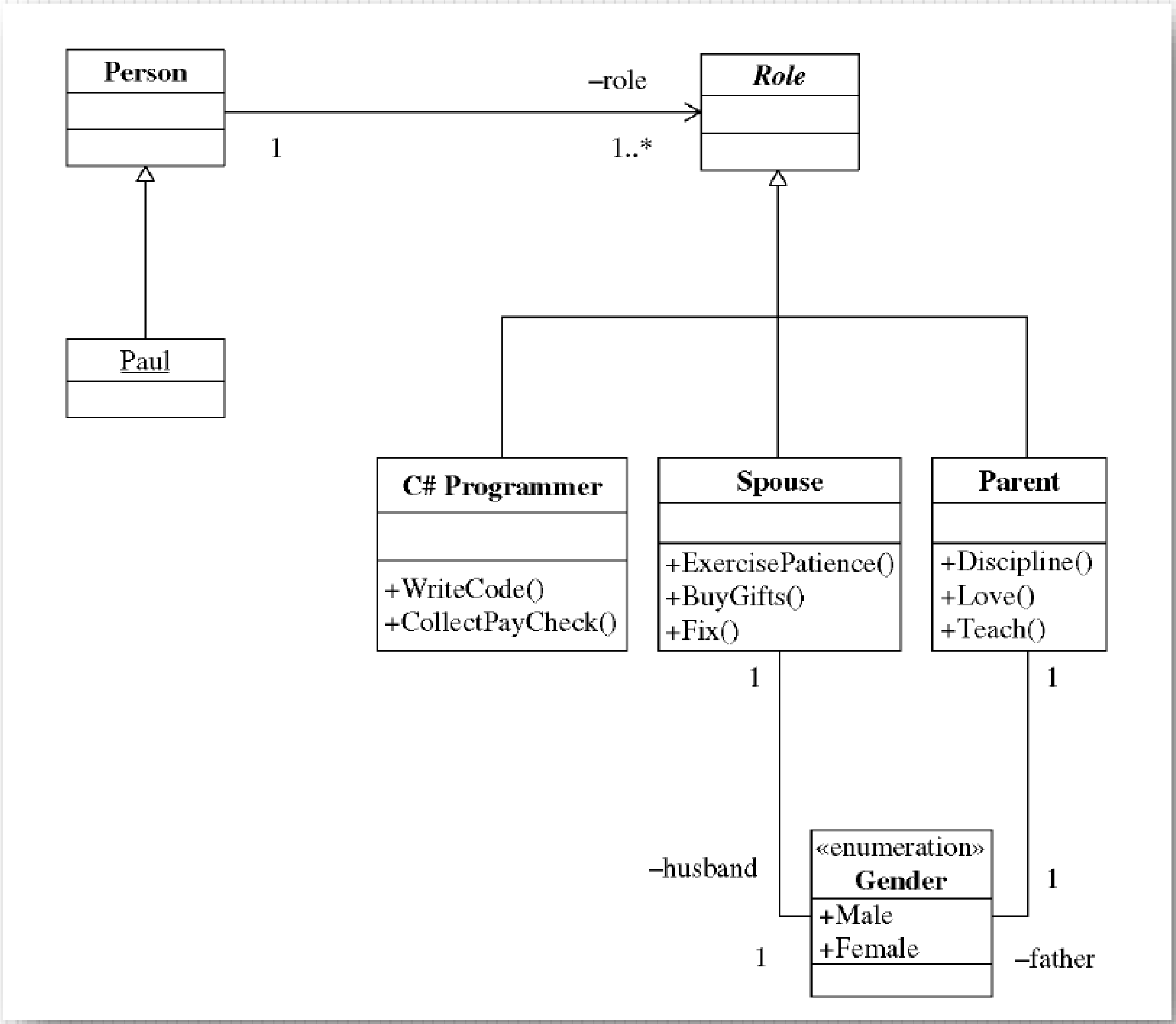
Exercise 1

Read the information represented in the diagram below



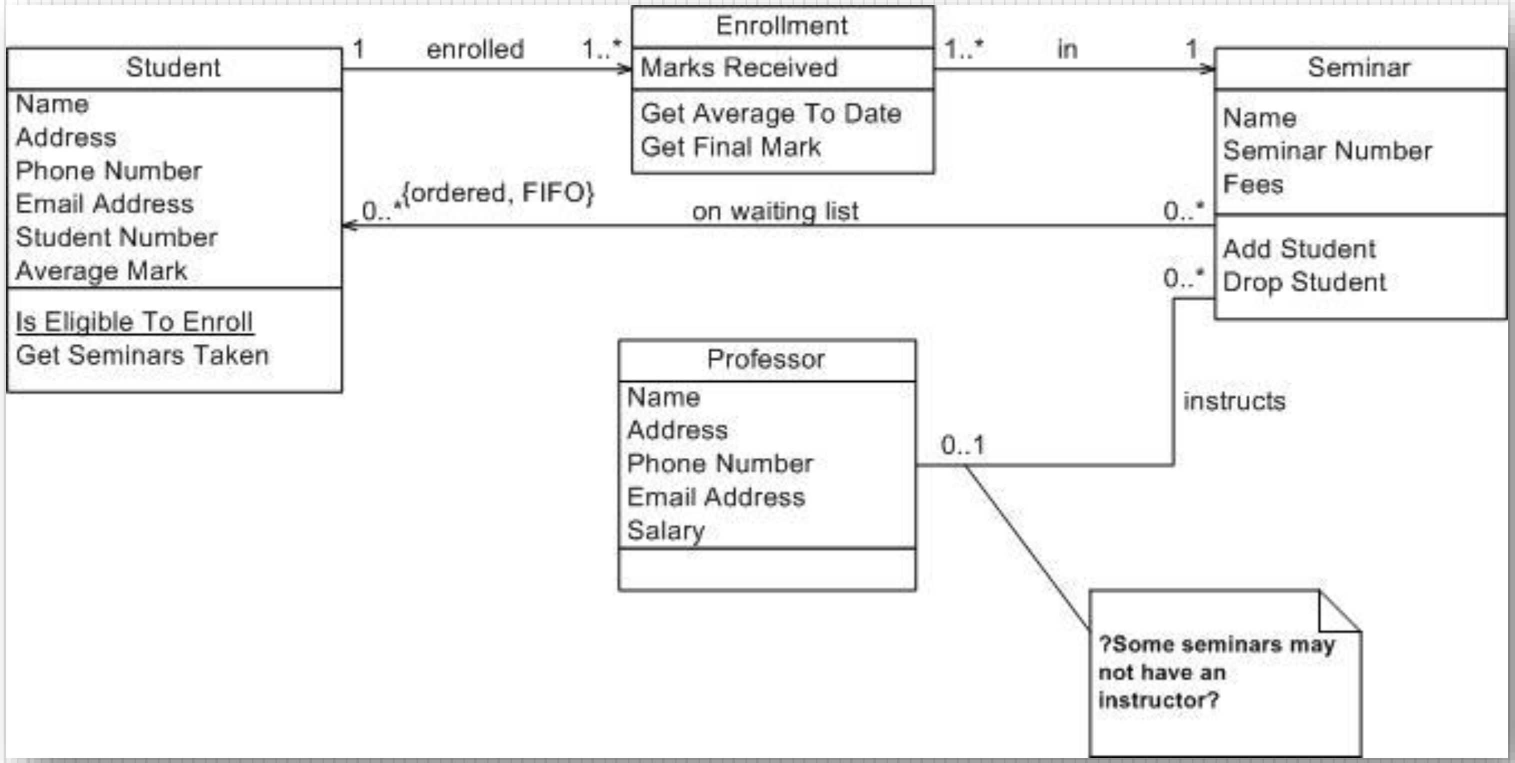
Exercise 2

Read the information represented in the diagram below



Seminar 3

Read the information represented in the diagram below



Seminar scenario

Create the class diagram for the scenario below

The project goal is to develope a software application for the management of a hotel business unit. In order to check in, a customer can request to reserve one or more rooms by e-mail or telephone. For this, he provides the receptionist with information on the period of accommodation and type of rooms required. Customers will get discounts if they reserve at least 3 rooms or if the period of accommodation exceeds 5 days. The receptionist checks availability and notifies the client of this and the estimated cost of accommodation. If there are no rooms available as requested, the receptionist can provide alternatives to the customer. The client may request a discount (additional or not) and the receptionist will decide the feasibility discount, assisted mandatory by the hotel manager. If the client agrees with the proposed price, they proceed to the reservation. For new customers, the receptionist asks identification data, which he introduces in the application.

Once at the hotel and if it has made a prior booking, the customer will provide his identification and / or booking number and the check in is finalized. If there is no reservation, the availability for the required period will be checked. When there is such a room, accommodation is made. At the end of the stay, the receptionist prepares a list of all the services used by the customer and their price. The list must be validated by the customer, then the final invoice is drawn up. The invoice can be paid partially or fully by bank transfer, cash or using a credit card. Also, before leaving the hotel, the customer is asked to complete a form to evaluate the services provided by the hotel premises