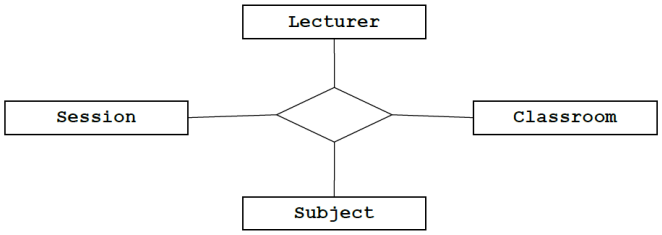
UML

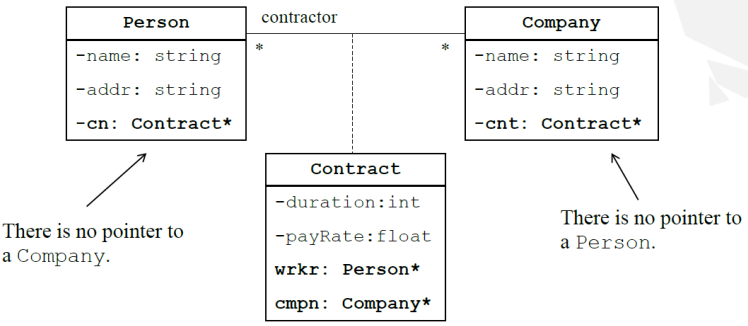
Class / Object Relations

- **Dependency**
  - One directional relationship indicating that one class uses another, or depends on it in some way. This is generally class to class.
  - It's like using the functionality of another class instance, such as cout.
  - X - -► Y X has access to info stored in Y
- **Association**
  - One object creates a link to another, this is through the “uses” type of relationship and can be multi directional with a multiplicity.
  - We may label associations to further explain the link between two classes
  - Optional associations can be created with a pointer, this will return null if no object has been set.
  - X —— Y X uses Y and Y uses X
- **Aggregation**
  - Aggregation is a stronger form of association, this reflects a “Owns” type of relationship. Destruction of either container/ composite does not destroy the other.
  - X ◇—— Y X owns Y
- **Composition**
  - Composition is the strongest version of association where the same “Owns/ is a part of” type of relationship is used, however the components are deleted along with the parent object.
  - This is created within CPP through the use of dynamic memory, where each component is deleted when the object's destructor is called.
  - X ◆—— Y Y is a part of X
- **Generalisation**
  - Inheriting qualities from a parent class, further attributes/methods can then be defined.
  - X ◄—— Y Y is a child of X

- **N-ary Associations**
  - Associations can be between more than two classes however most programming languages cannot express these properly. Instead we generally promote this relationship into an association class

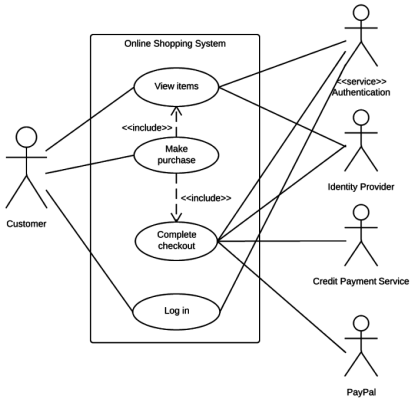


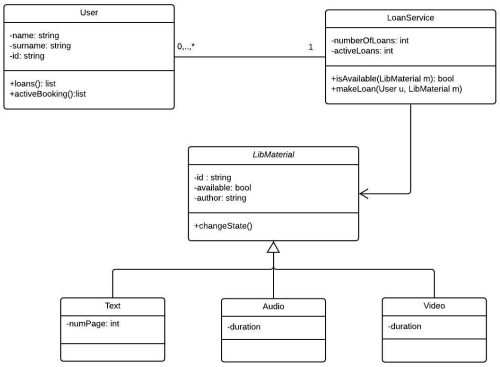
  -
- **Association class**
  - Association classes are used as an association between two objects, this association link is formed as a class to hold extra information about the association.

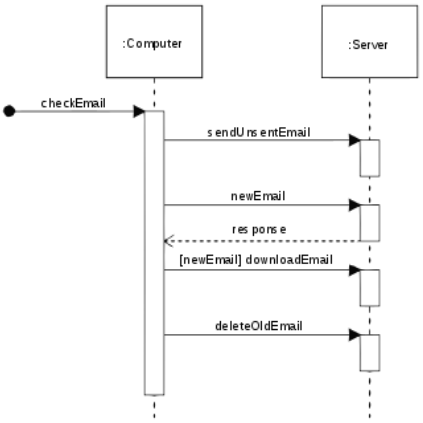


  -

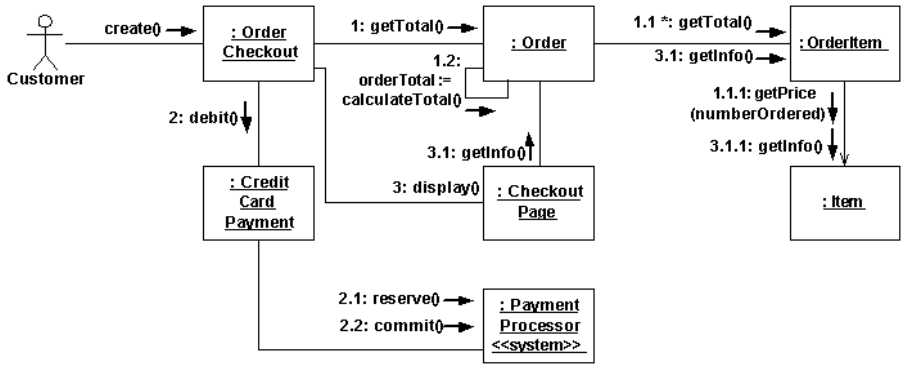
Types of UML

- **Use Case Diagram**
  - A very high level diagram of a users interactions with a system
  - 

- **Class Diagram**
  - Normal UML class diagram created previously
  - 

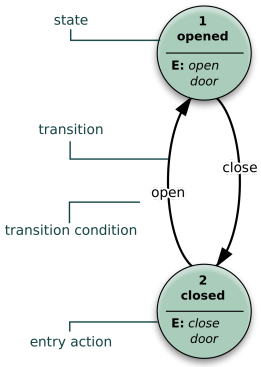
- **Sequence Diagram**
  - Object interactions ordered by a time sequence
  - 

- **Collaboration Diagram**
  - High level design of object communications as the execution of a program is carried out.



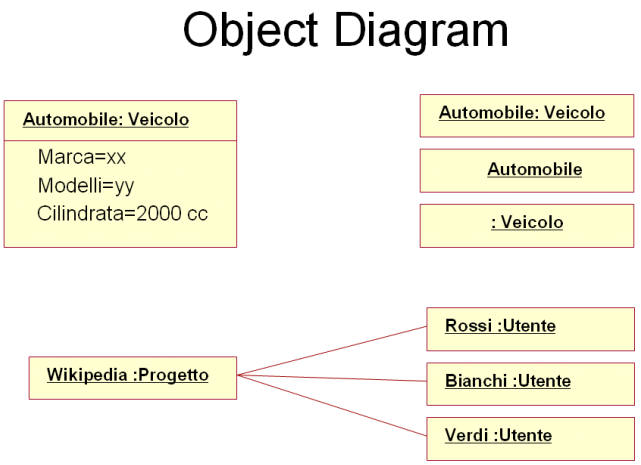
  -

- **State Diagram**
  - Describes the behaviour of systems when it is in different states.



  -

- **Object Diagram**
  - Where we define each object that we create, giving a snapshot of the system at a particular moment



  -