UML:

* Unified Modeling Language is way of represent things in design diagram
* UML makes understanding the feature easier and easy to implement
* It is a graphical way of representing relationship between classes

There are 2 types of UML:

* Structural (concern on the structure of the code)
  + **Class Diagram (imp)**
  + Component Diagram
  + Package Diagram
  + Object Diagram
* Behavioral (concern on how the system works and interacts)
  + **Sequential Diagram (imp)**
  + **Use Case Diagram**
  + Activity Diagram (imp)

Important UML Diagrams in LLD:

**Use Case Diagram:**

* Visual representation of how users interact with the system
* Helpful for understanding functional requirement in user perspective

Notations:

* Actor:
  + External Entities who interacts with the system. They initiates the usecases
* Use Case:
  + These are the actual functionalities of the system
* System Boundary:
  + Limits of the system

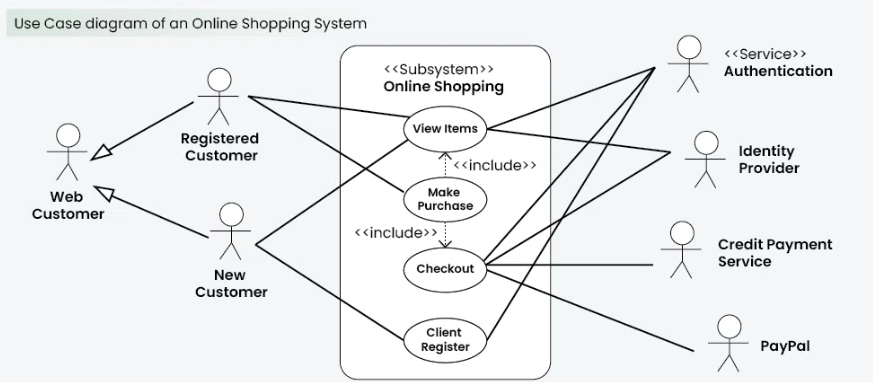
Association Relationships:

* The Association Relationship represents a communication or interaction between an actor and a use case.

Steps to Draw a usecase diagram:

1. Identify Actors
2. Identify Use Cases
3. Connect Actors and Use cases
4. Add System Boundary
5. Define Relationships
6. Review and Refine
7. Validate

Example Online Shopping System

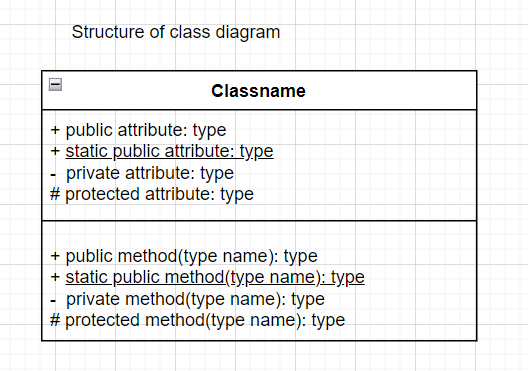


**Class Diagram:**

* These diagrams visually represent the structure and relationships of classes in a system.
* Class contains the attributes and methods
* There are access levels for attributes and methods.
  + **Public:** represented with “+”
  + **Private:** represented with “- “
  + **Protected:** represented with “#”
* Static method and properties are represented by underling the attribute/method

**Notation:**

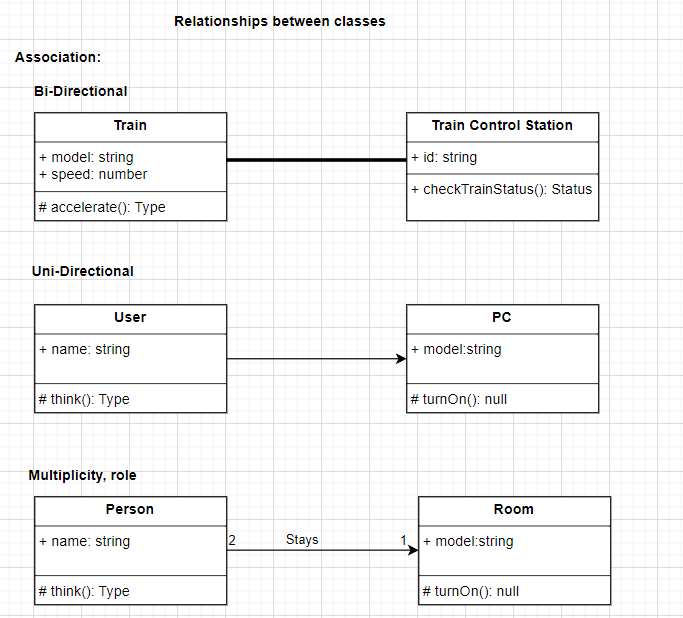
* Attributes – properties of the class
* Methods – behavior of the class



**Relationship between classes:**

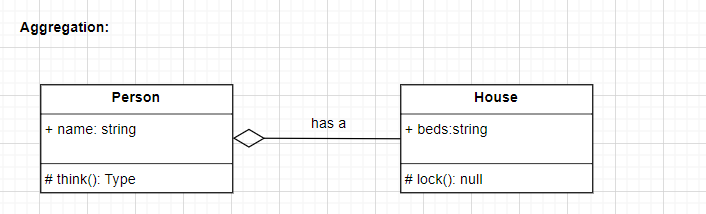
**Association Relationship:**

1. Two classes can called each other.
2. There are different types:
   1. Bi-Directional
   2. Uni-Directional
   3. Multiplicity, Role (represent the no of one class members can access other class)



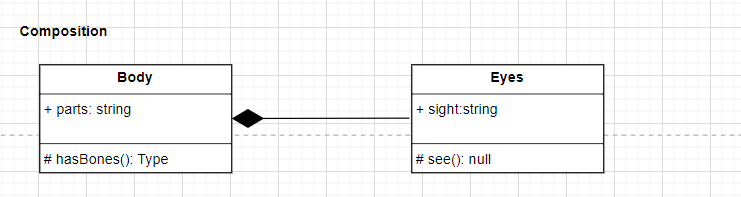
**Aggregation Relationship:**

1. Wherever you see a relationship like Class A **has an relationship** with Class B where B can exist without A
   1. Eg: vehicle has a tyre



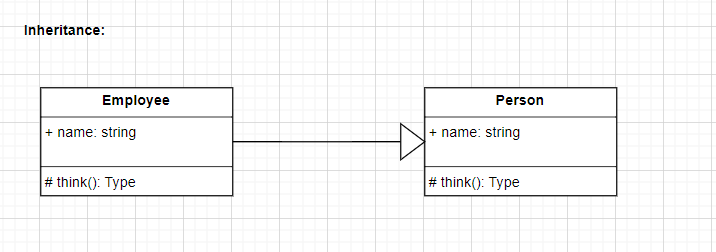
**Composition Relationship:**

1. Wherever you see a relationship like Class A **has an relationship(part of)** with Class B where B cannot exist without A
   1. Eg: body has an eye



**Inheritance Relationships:**

1. Wherever you see a relationship like Class A **is a** Class B
   1. Eg: Employee is a Person

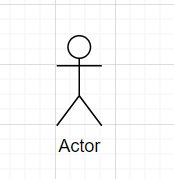


**Sequence Diagram:**

* It is interaction between the object in a sequential manner.
* Sequence diagrams describe how and in what order the objects in a system function.

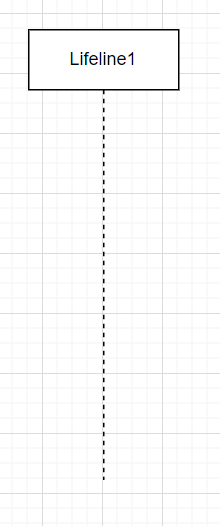
**Notation:**

* **Actor:**
  + A type of role which interacts with the system and the objects in it.
  + Sequence diagrams describe how and in what order the objects in a system function.

****

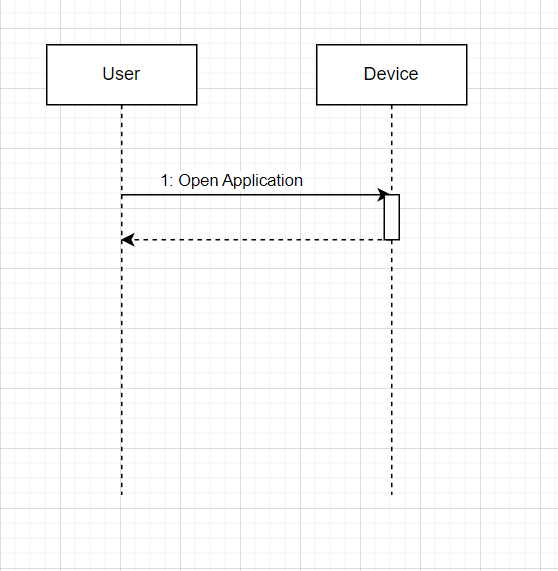
**Lifeline:**

* A lifeline is a named element which depicts an individual participant in a sequence diagram.
* Each Instance in the sequence diagram is represented with lifeline
* It is basically a rectangle box with the title inside it



**Messages:**

* Communication between objects is depicted using messages
* These messages are in sequential order of the life line
* Messages are represented with arrow from one lifeline/actor to other



**Types of messages:**

1. Synchronous Messages:
   1. A synchronous message waits for a reply before the interaction can move forward.
   2. Represented with solid line arrow
2. Asynchronous Messages:
   1. A asynchronous messages does not wait for a reply with continuous to move forward in the interaction
   2. Represented with dotted line arrow.
3. Create Messages:
   1. These messages indicate the creation of a instance
   2. Represented with <<create>> in message with dotted and solid arrow to other lifeline
4. Delete Messages:
   1. These message indicate the deletion of a instance
   2. Represented with cross mark at the end of the lifeline
5. Self Messages:
   1. Certain scenarios might arise where the object needs to send a message to itself. Such messages are called Self Messages
   2. Represented with a U shaped arrow.
6. Reply Messages:
   1. The message being sent from the receiver to the sender.
   2. Represent a return/reply message using an open arrow head with a dotted line.
7. Found Messages:
   1. Found message is used to represent a scenario where an unknown source sends the message.
   2. It is represented using an arrow directed towards a lifeline from an end point.
8. Lost Message:
   1. A Lost message is used to represent a scenario where the recipient is not known to the system.
   2. Represented using an arrow directed towards an end point from a lifeline.
9. Guards:
   1. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

REFERENCE: <https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/>