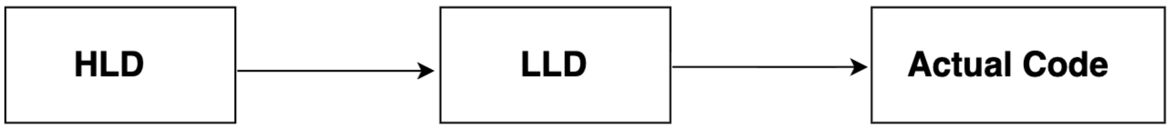
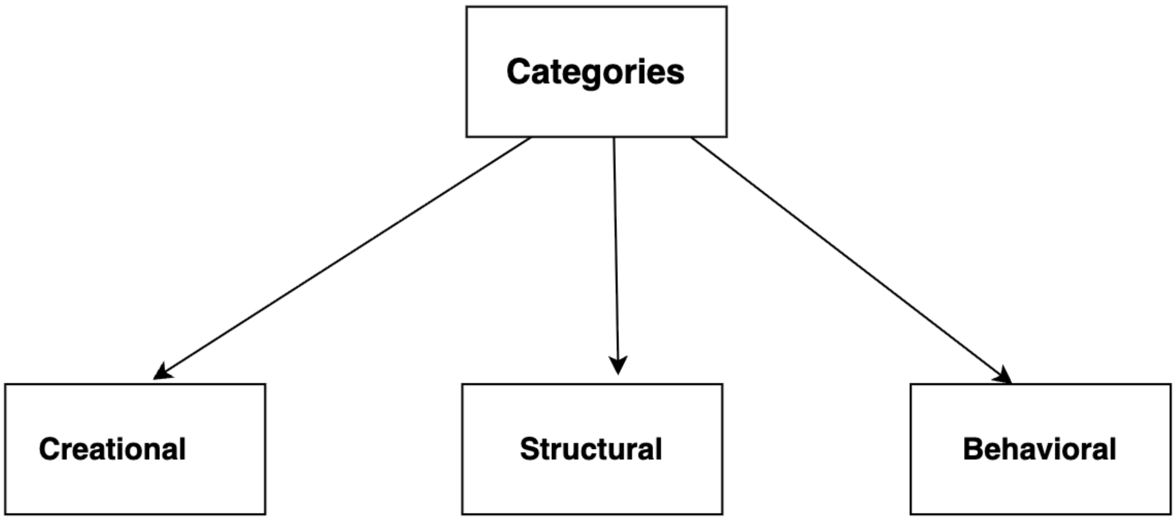
What and Why is LLD (System Design) important ??

- It majorly focuses on classes and objects within a system.



Categories of LLD patterns:



Creational:

- It controls the creation of objects in a system.
- Various types of creational patterns are —

- Singleton
- Builder
- Factory
- Abstract Factory
- Object Pool
- Prototype

Structural:

- It focuses on how different classes/objects are arranged together so that larger problem can be solved in most flexible way.
- Various patterns are —

- Decorator
- Proxy
- Composite
- Adapter
- Bridge
- Facade
- Flyweight

Behavioural:

- It focuses on how different objects interact with each other.

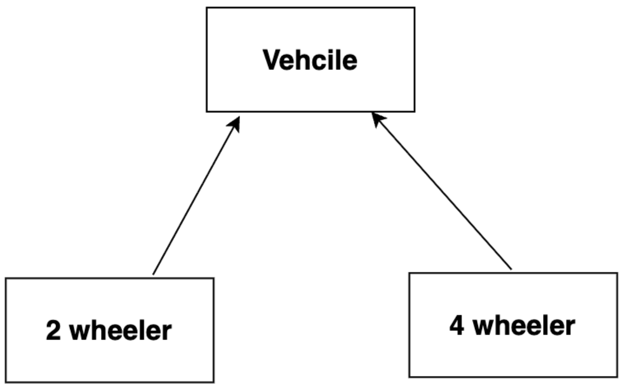
- Like in other words, with above structural pattern we build skeleton of the system but how skeleton behaves (coordination, interaction, responsibility) is all guided by Behavioural Pattern.

- Various Patterns are

- State
- Strategy
- Observer
- Chain of Responsibility
- Template
- Iterator
- Interpreter
- Command
- Visitor
- Mediator
- Memento
- Null object

Has-a and is-a relationship

is-a = This is nothing but inheritance



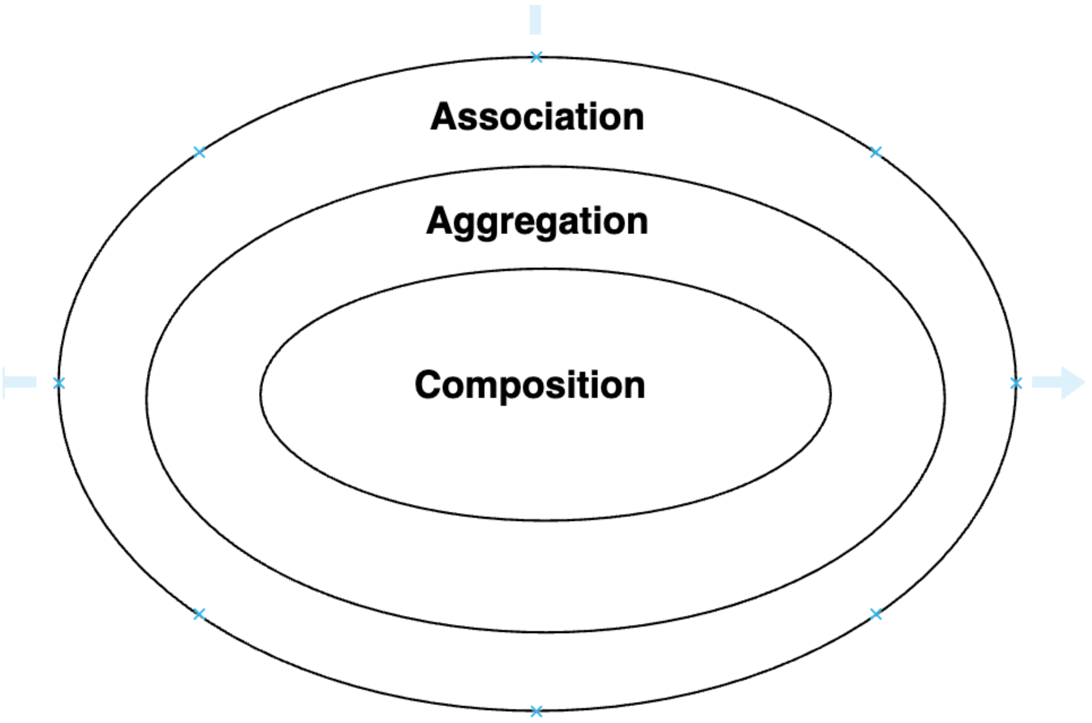
2 wheeler is a `Vehcile` —> (2 wheeler is inheriting features of Vehcile)

4 wheeler is a `Vehcile` —> (4 wheeler is inheriting features of Vehicle)

has-a = This is nothing but shows link between 2 objects

Ex: Library has Books
Ex: House has Rooms

has-a (association has two types they are — Aggregation and Composition relationships)



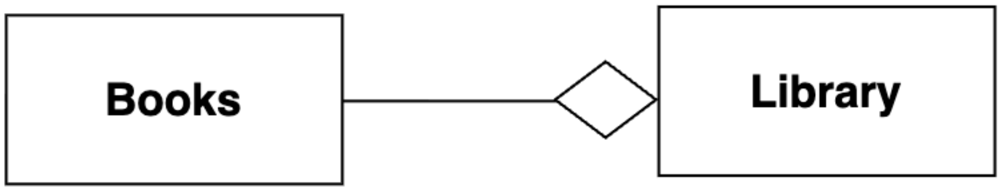
Aggregation: ——◇

This is used when we want to depict the relationship between two objects is weak. That is one object is independent of another object (More precisely, even if one object gets deallocated from memory, other stays as is). It is depicted by arrow with open-diamond head.

Ex:
Library has books.
Here Library can live independently without Books and vice-versa.

```
class Library {
   //Library class just has books, no operations related to books,
like updating books and all. So even if Library class gets
deallocated, Books object still stays in memory.
   var array: [Book] = [ ]

}
```



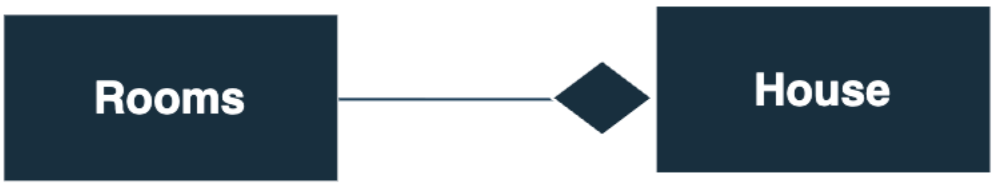
Composition: ——◆

This is used when we want to depict the relationship between two objects is strong. That is one object is dependent of another object (More precisely, even if one object gets deallocated from memory, other stays also gets deallocated). It is depicted by arrow with close-diamond head.

Ex:
House has Rooms.
Here Rooms cannot exist independently without House.

```
class House {
    //House class has Room, and operations related to Room, like
updating books and all. So if house gets deallocated, Room also
gets deallocated.
   var array: [Room] = [ ]

   array.add(Room)
   add.remove(Room)
}
```


Rooms
House