SOLID Principles for Programming and Software Design

SOLID is a mnemonic acronym that stands for the five design principles of Object-Oriented class design. These principles are:

S - Single-responsibility Principle

display and validate the information of person separately -> code example

O - Open-closed Principle

L - Liskov Substitution Principle

“Let Φ(x) be a property provable about objects x of type T. Then Φ(y) should be true for objects y of type S where S is a subtype of T”.

real life example ->

Φ(x) = a pen's ability to write

objects x of type T = objects of Pen

Φ(y) = any ball-point pen, gel pen or a roller-ball pen can write as much as any other type of pen

objects y of type S where S is a subtype of T = a ball-point pen, a gel pen or a roller-ball pen

I - Interface Segregation Principle

D - Dependency Inversion Principle

---------------------------------------------

The S.O.L.I.D Principles in Pictures

Images for better understanding

Ref: https://medium.com/backticks-tildes/the-s-o-l-i-d-principles-in-pictures-b34ce2f1e898

Ref: https://www.freecodecamp.org/news/solid-principles-for-programming-and-software-design/

---------------------------------------------

SOLID: The First 5 Principles of Object Oriented Design

skip this one (uses php)

Ref: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

---------------------------------------------

https://realpython.com/solid-principles-python/

refer this one if you use python (uses python)

---------------------------------------------

The SOLID Principles of Object-Oriented Programming Explained in Plain English

important article (using Java)

Ref: https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/

---------------------------------------------

A Solid Guide to SOLID Principles

important article (using Java)

Ref: https://www.baeldung.com/solid-principles

---------------------------------------------

difficult to understand

Ref: https://github.com/mikeknep/SOLID/tree/main/liskov\_substitution

---------------------------------------------

difficult to understand

Ref: https://github.com/chethu/SOLID-Principles

---------------------------------------------

easy to understand examples

Ref: https://github.com/kawser2133/SOLID-design-principles

---------------------------------------------

Low Level Design Primer Course

https://www.youtube.com/playlist?list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k

-----------------------------------------------

Low Level Design 101 | Steps and Resources to learn LLD as a beginner | sudocode | System Design

Low Level Design Theory

- Understanding basics of OOD and principles

- Understanding OOAD

- Ample knowledge of design patterns to solve and write code

- SOLID Principles

Low Level Design Practice

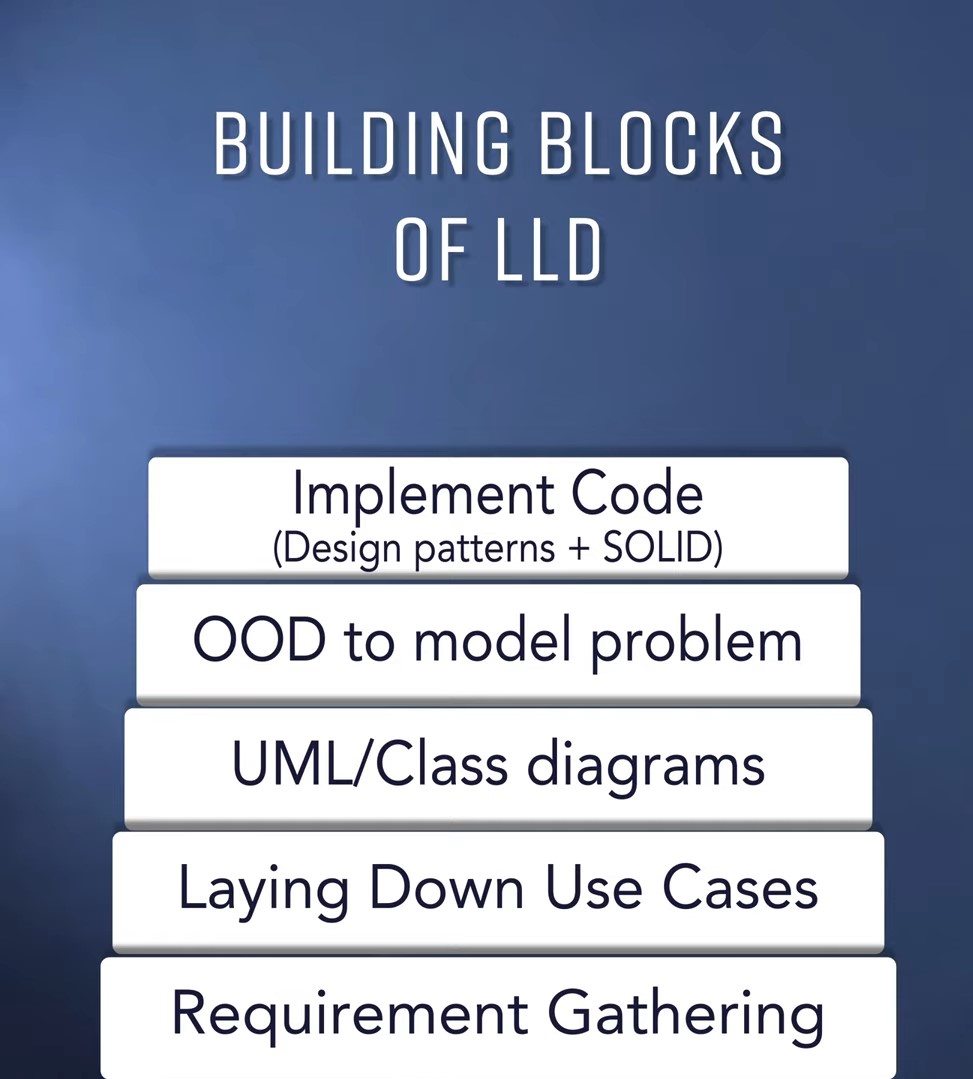
- Pick up a problem, solve it, and then analyse for betterment

Ref: https://www.youtube.com/watch?v=B3zrIwz\_t4M&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=1

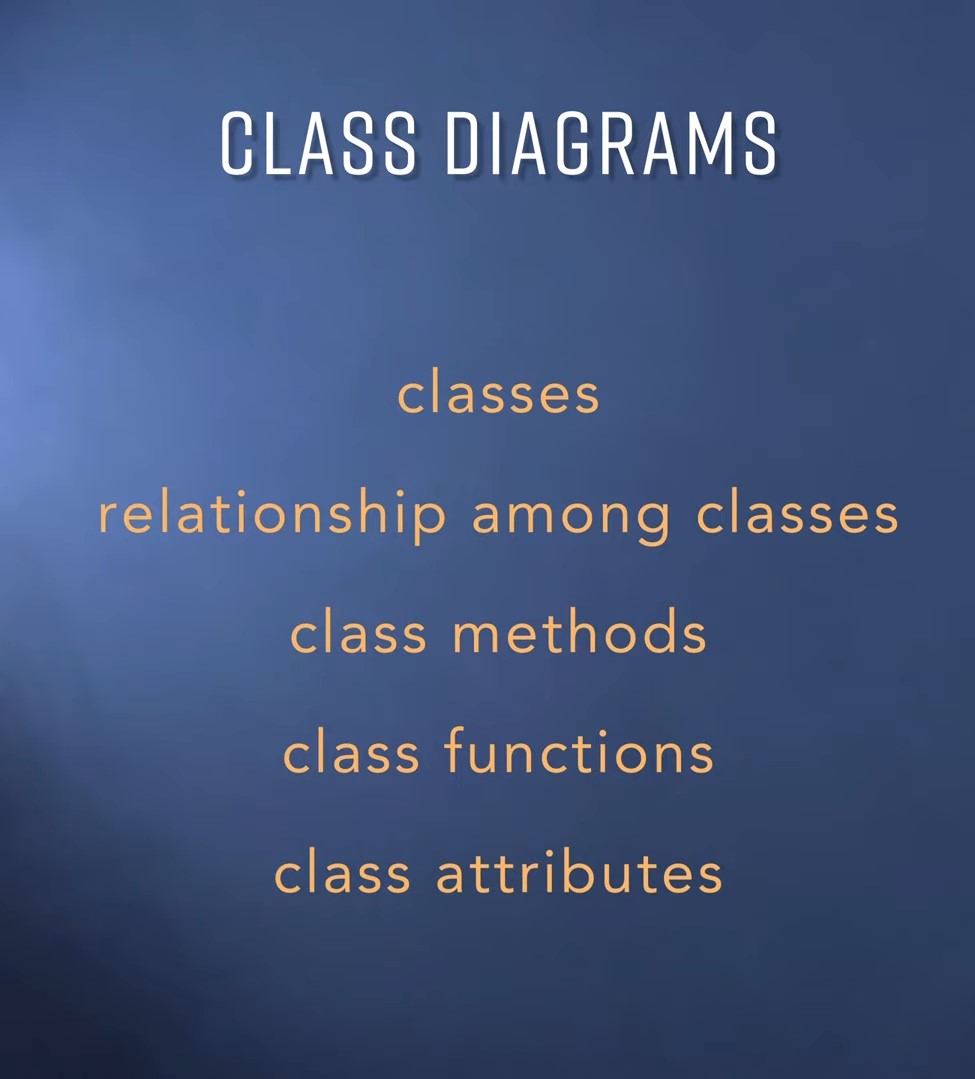
-----------------------------------------------

Low Level Design 102 |What is LLD(Low Level Design)? Simplified steps to learn and break LLD problem

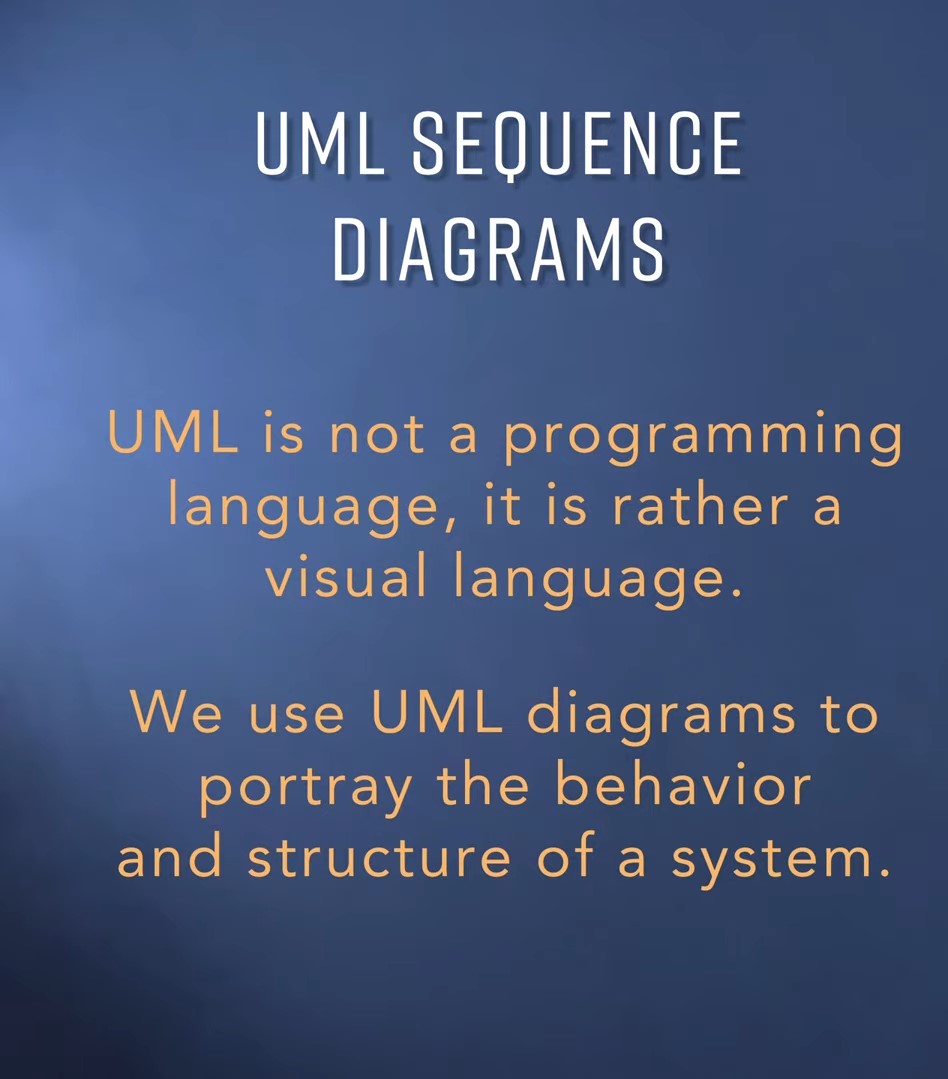
LLD ->



[Image: LLD-1]



[Image: LLD-2]



[Image: LLD-3]

Ref: https://www.youtube.com/watch?v=5Tz9EUMHOGI&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=2

-----------------------------------------------

Low Level Design 103 | How to design low level applications using objects and classes | OOD

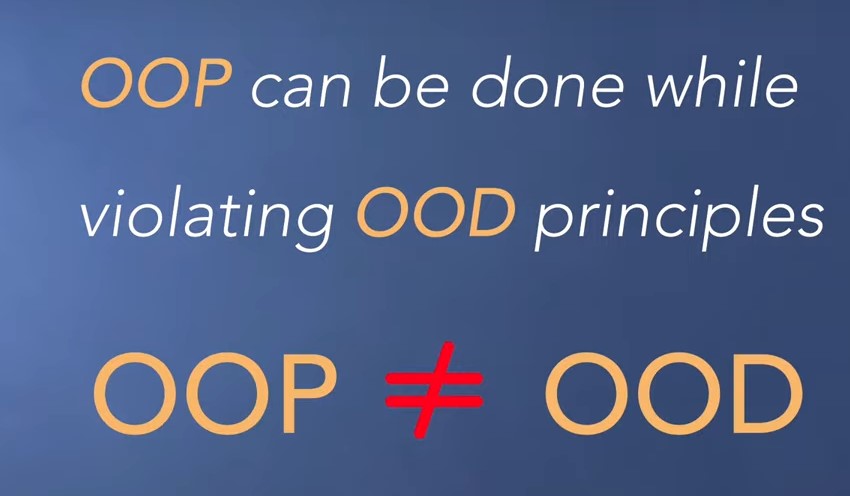
OOD is not the same as OOP



[Image: LLD-4]



[Image: LLD-5]



[Image: LLD-6]

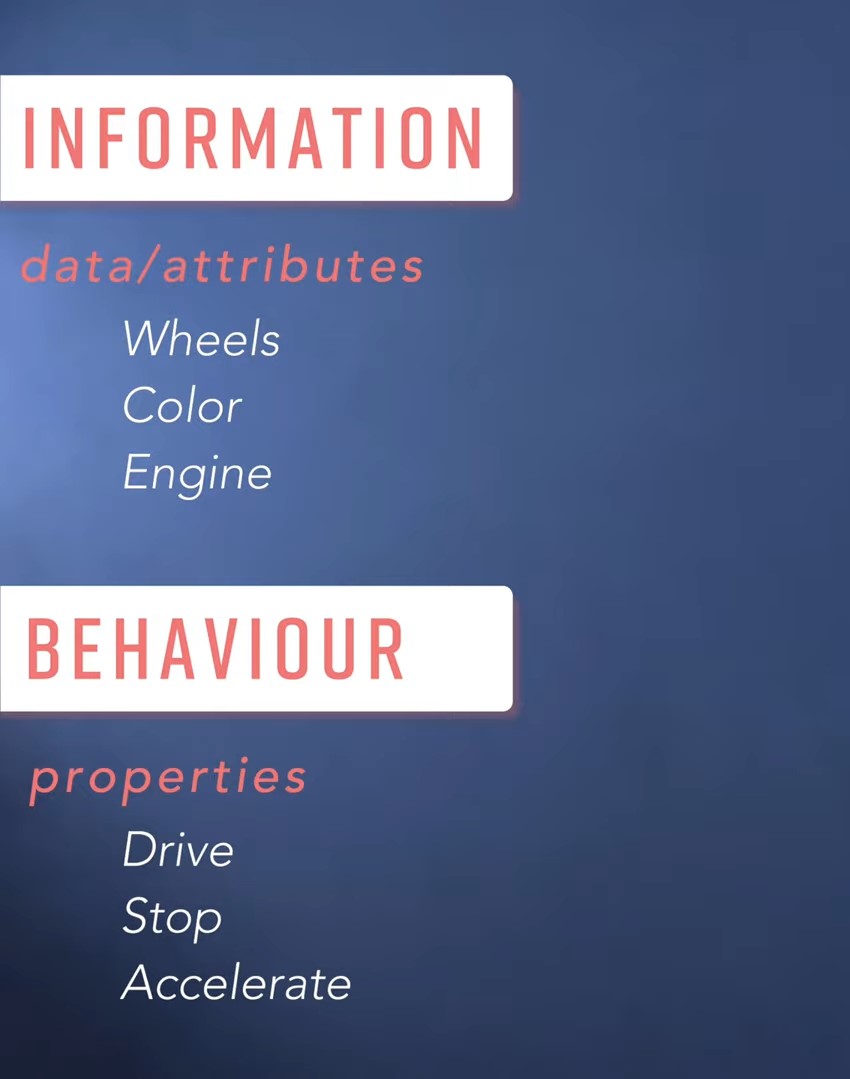
cookie cutter -> class

cookies -> objects

NVT - Noun Verb Technique

Noun <-> Objects(Classes)

Verb <-> Behavior



[Image: LLD-7]



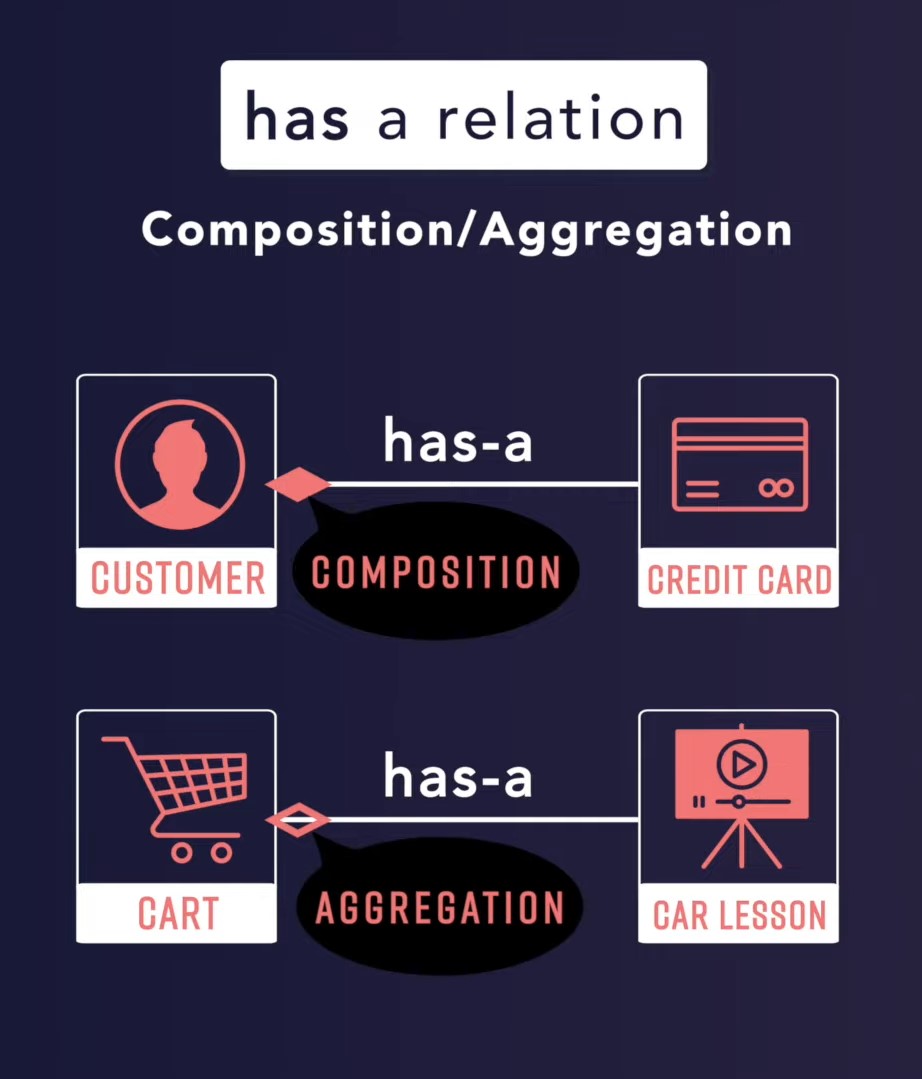
[Image: LLD-8]

Ref: https://www.youtube.com/watch?v=FbQYifw9Be8&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=3

-----------------------------------------------

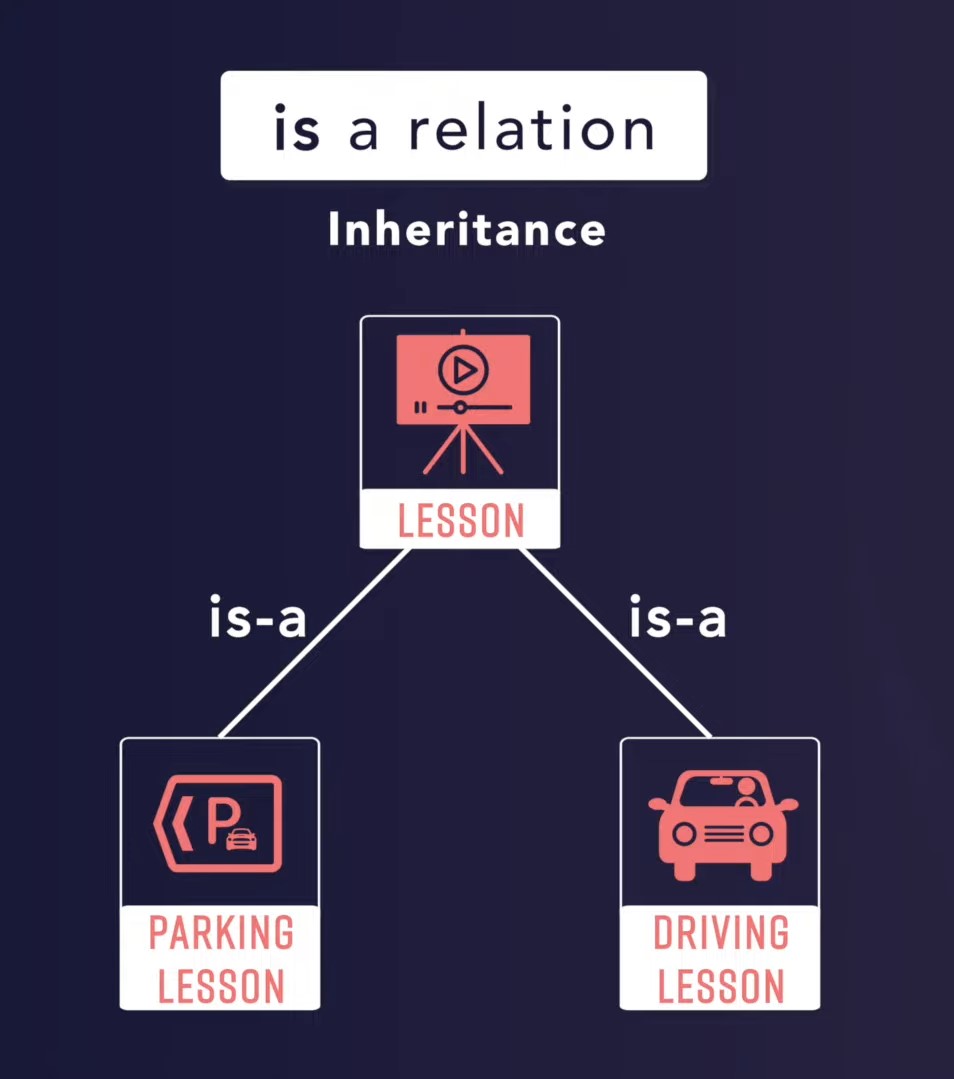
Low Level Design 104 | How to build Classes in Object Oriented Design | 2022 | System Design

has-a relation

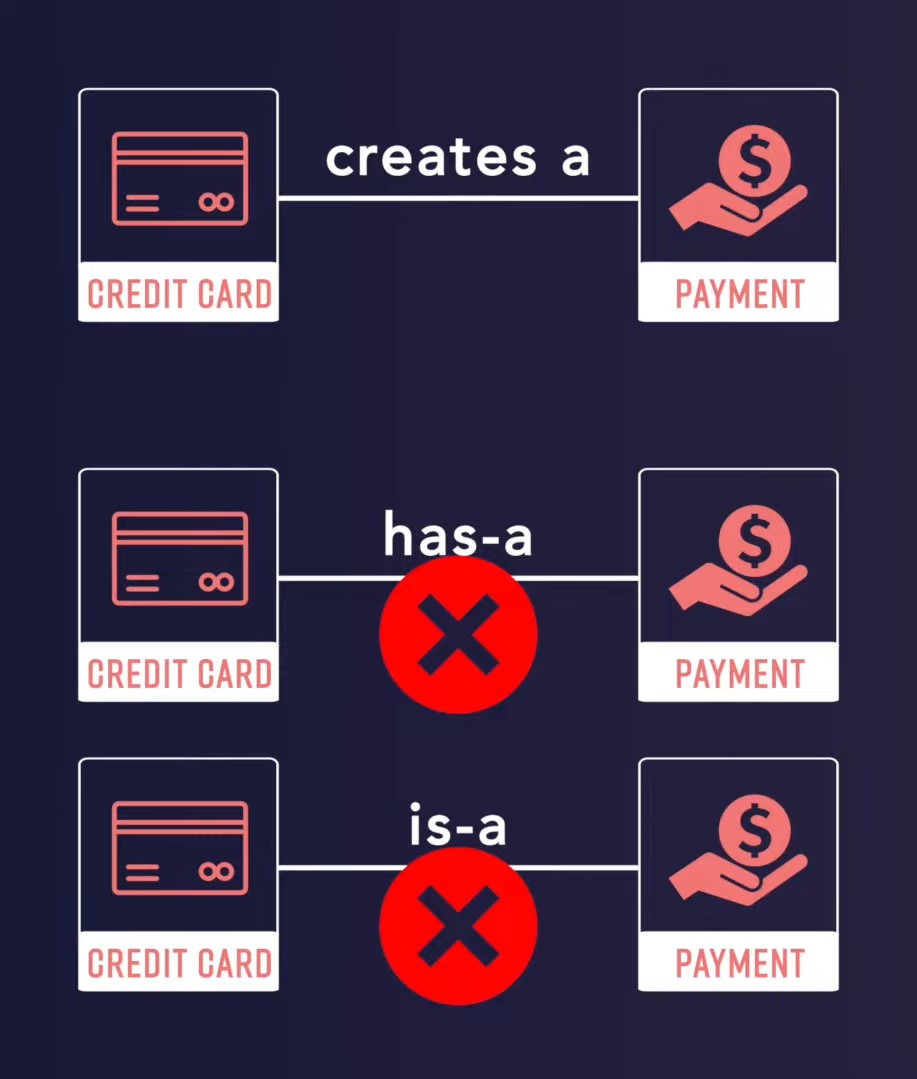


[Image: LLD-9]

is-a relation



[Image: LLD-10]



[Image: LLD-11]

Implementation of Abstraction is called Encapsulation.

Ref: https://www.youtube.com/watch?v=\_wmyYdOkCo4&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=4

-----------------------------------------------

Low Level Design 105 | Single Responsibility Principle in SOLID | 2022 | System Design

Note: It does not mean that a class should have only one public method, it implies that a module should be changed by only one business stakeholder or a group of stakeholders.

Design is subjective, there is no single right answer. If you keep decomposing, you will end up with a lot of classes. It is up to your judgement to decompose or combine the functionalities, but it is necessary that the software is fulfilling the stated business requirements and is also easy to maintain or change.

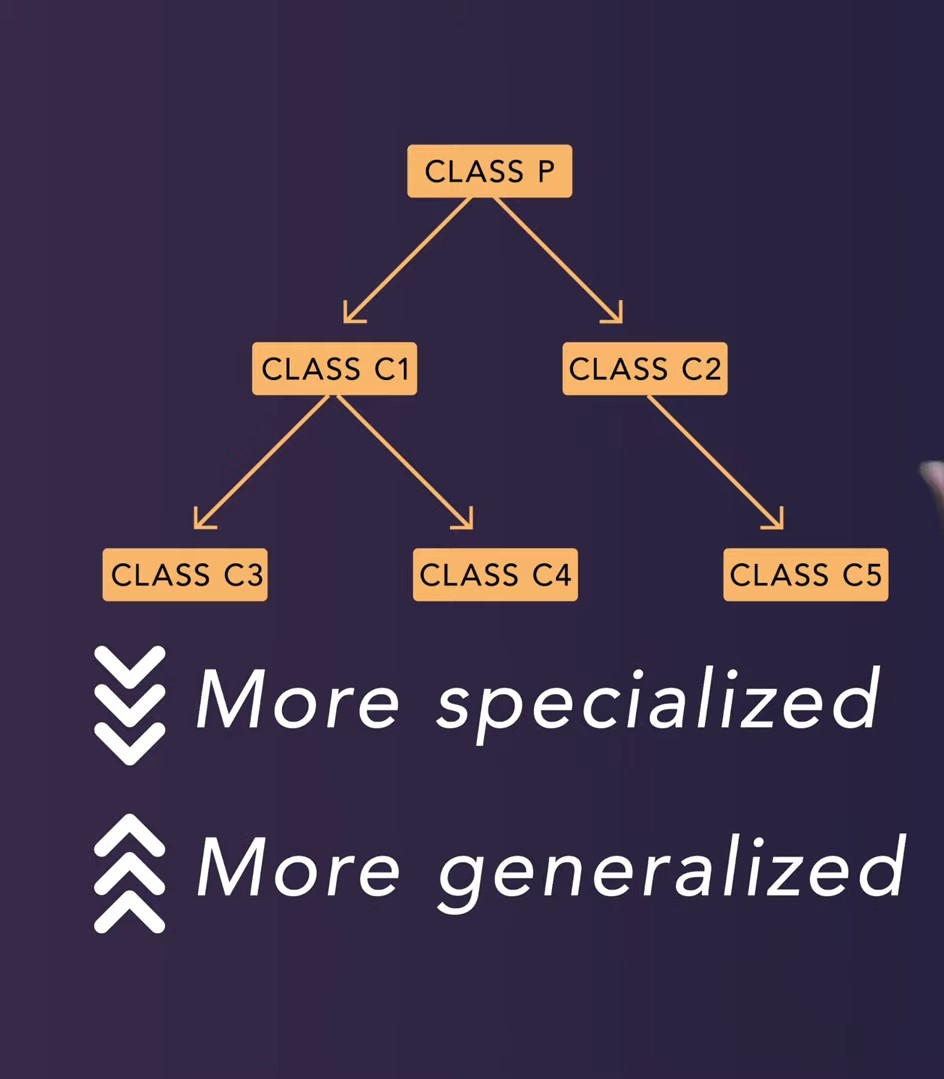
Ref: https://www.youtube.com/watch?v=-T0H--a8WnY&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=5

-----------------------------------------------

Low Level Design 106 | Inheritance and Polymorphism | 2022 | System Design

Inheritance -

parent-child relationship



[Image: LLD-12]

Polymorphism -

e.g. getArea() method in Circle, Square and Rectangle is overridden as its existing implementation in the super class Shape

Ref: https://www.youtube.com/watch?v=tIavGKafBV8&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=6

-----------------------------------------------

Low Level Design 107 | Liskov Substitution Principle | 2022 | System Design



[Image: LLD-13]

Ref: https://www.youtube.com/watch?v=HbGDobtxzWk&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=7

-----------------------------------------------

Low Level Design 108 | Interface Segregation Principle | 2022 | System Design

Avoid large interfaces that have a lot of functionality

Instead, make smaller interfaces with their own functionality

Ref: https://www.youtube.com/watch?v=-E-E6pd-psU&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=8

-----------------------------------------------

Low Level Design 108 | Dependency Inversion Principle | 2022 | System Design

Ref: https://www.youtube.com/watch?v=\_CQuOfIqaGE&list=PLTCrU9sGybupCpY20eked6blbHI4zZ55k&index=9