Introduction to Object Oriented Programming :→

* Suppose you want to represent a real world entity called "Student", with given properties:

(Student) → name *, section *, roll No *, email ID *

Samar, A, 101, samar@gmail.com

These are called: → attributes, → properties, → fields } any one of these is fine

This same entity can be represented by using a class:

```
→ class Student {
    String name; char section;
    int rollNo; String emailID;
}
```

Has no Memory

A "class" is a blueprint or template to create objects.

* Code reusability [(Student)] → multiple objects of the same type

DRY → Don't repeat yourself. (Time)

Object → It is an
* instance of a class.
* reference
* object
* variable
* key

Constructor: A constructor is a special method used to initialize objects by using "new" keyword.

* It is same name as the class.
* It has no return type.
* There can be infinite constructors in a class. (Constructor Overloading)

**** If we create our own constructor, the "default constructor" gets deleted by JVM.

* Two types: ① Default / No-argument constructor
② Parameterized constructor.

* If we don't create constructor, the JVM generates default constructor.

"this" keyword refers to the calling object & all other fields from the current class.

P → sum
C → sum
this.sum
super.sum

"super" keyword is used for the parent class.

4 Pillars of Object Oriented Programming

① Encapsulation :→ Wrapping the data & the code inside a block so that they are not accidentally modified, is called encapsulation.

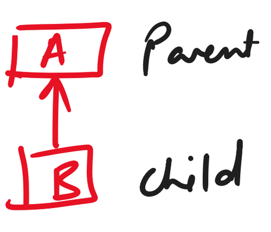
The block here is called the class.
Encapsulation is achieved by using "private" access modifier.
We can access those members outside the class by using two special methods:
① setter ② getter

Inheritance :→ (NO) can a class extend multiple classes?

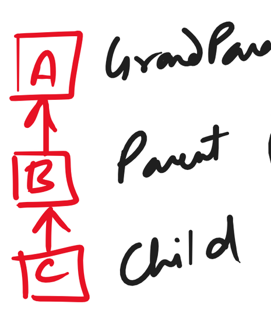
The property by which a child class object can use members of the parent class, is called inheritance.

In programming, we generally have 5 types of inheritance.

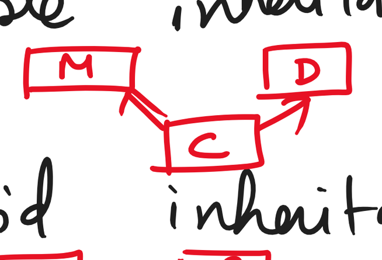
① single level



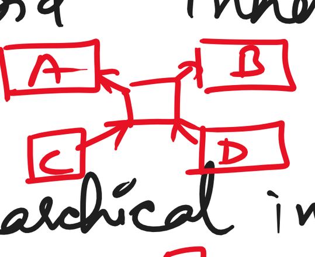
② multi level (extends)



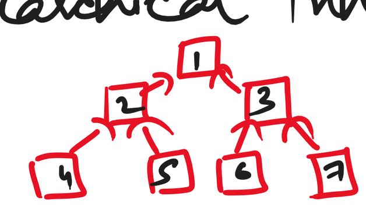
③ ✗ multiple inheritance (Interface)



④ Hybrid inheritance



⑤ Hierarchical inheritance



Diamond Problem:→ (Object)

Interface

```
Class Cat                    Class Dog
makesound() {                make sound() {
  "meow";!                     "woof"!
}                            }
class Animal extends Cat, Dog {
    Animal cog;
    cog.makeSound(); ✗
}
```

JVM confused ←

* Abstraction: (Better User Experience)

functionality ← showing → what is being done
implementation hiding → how it is done

Hiding the implementation & showing only the functionality to the end-user is called data abstraction.

① Abstract classes & methods (0 – 100 %)
↑ It can have both normal & abstract methods.

② Interfaces (100 %)
It can have any empty methods.

Gnana →

Place	Role
Class	Student
Canteen	Customer
Home	Daughter
Library	Student

Poly + Morph
many forms/shapes

The process by which same method behaves differently is Polymorphism

① static | compile time | over loading → Same Class
① no of parameters ① return type
② dynamic | run time | over Riding → Different classes