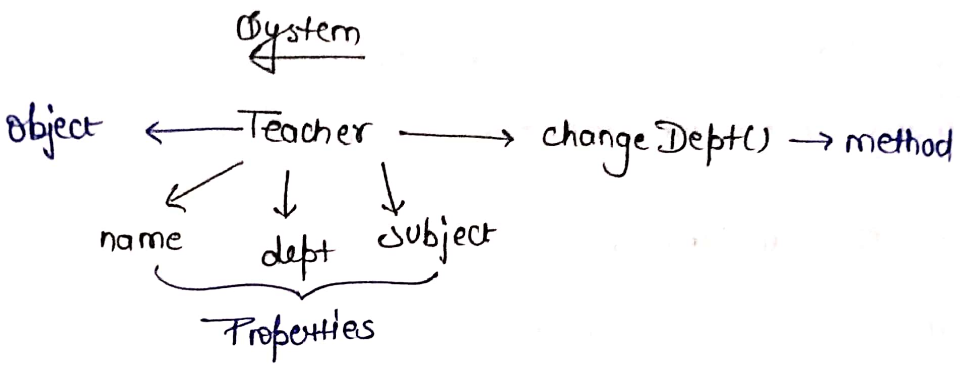
[OOP's]

classes & objects :-

→ objects are entities in the real-world.
→ class is like a blueprint of these entities.

Ex:-



Access modifiers :-

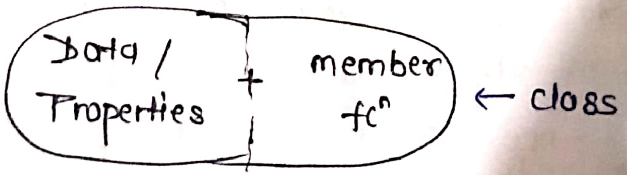
Private : data & methods accessible inside class

Public : Data & methods accessible to everyone

Protected : Data & methods accessible inside class & to its derived class.

Encapsulation :-

is wrapping up of Data & member fc



⇒ Data hiding → private (Access modifiers)

Constructor :- special method invoked automatically at time of object creation.

→ used for initialization.

→ same name as class

→ constructor doesnot have a return type

→ only called once (automatically), at object creation

→ Memory allocation happens when constructor is called

⇒ Types of Constructor –

1) Parameterized Constructor

2) Non-parameterized Constructor

3) Copy Constructor

⇛ This pointer :- this is a special pointer in C++ that points to the current object.

this→prop is same as *(this).prop

⇛ copy Constructor :- special Constructor used to copy properties of one object into another.

⇛ shallow & Deep copy :-

(Dynamic memory allocation's issue)

→ A <u>shallow</u> copy of an object copies of all the member values from one object to another.

→ A <u>deep</u> copy, on the other hand, not only copies the member values but also make copies of any dynamically memory that the members point to.

Destructor :- opposite of Constructor

→ Deallocate, statically allocate memory } only in C++

~ className ( ) { }

Inheritance :- when properties & member fc^n's of base class are passed on to the derived class.

$$\begin{tikzcd}
\text{class A (parent, Base)} \arrow[d, "\text{inherit}"] \\
\text{class B (child, Derived)}
\end{tikzcd}$$

⇛ mode of Inheritance :-

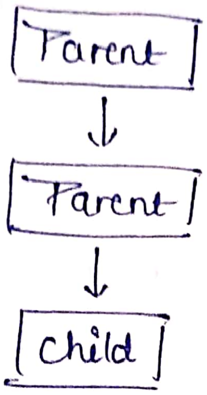
Base class	Derived class		
	Private mode	Protected Mode	Public Mode
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Private	Protected	Protected
Public	Private	Procteded	Public

⇛ Types of Inheritance :-

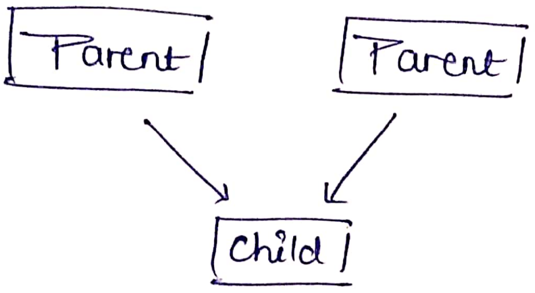
1) Single Inheritance –

$$\begin{tikzcd}
\boxed{\text{Parent}} \arrow[d] \\
\boxed{\text{Child}}
\end{tikzcd}$$

2) Multi-level Inheritance :-

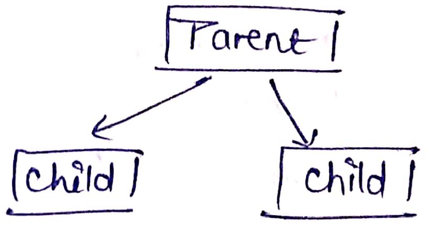


3) Multiple Inheritance :-

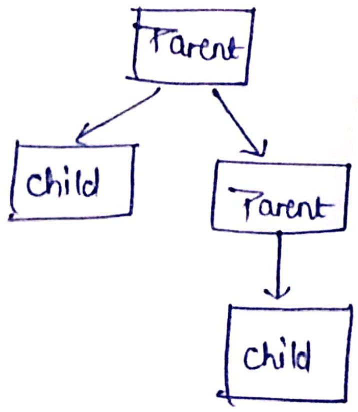


} not allowed in Java

4) Heirarchial Inheritance :-



5) Hybrid Inheritance :-



Polymorphism :- is the ability of objects to take on different forms or behave in different ways depending on the context in which they are used.

⇒ Types :-

1) Compile time Polymorphism (overloading)
2) Runtime Polymorphism (overridding)

⇒ fc^n overloading :- different parameters, same name of method

⇒ fc^n overridding :- Parent & child both contain the same fc^n with different implementation. The parent class fc^n is said to be overridden.

⇒ Ex of Compile time polymorphism —

1) fc^n overloading 2) operator overloading

⇒ Exe of Run time Polymorphism —

1) fc^n overridding

2) Virtual fc^n's :- is a member fc^n that you expect to redefined in Derived class.

→ Virtual fc^n are dynamic in nature.

→ Defined by the keyword "virtual" inside a base class and are always declared with a base class & overridden in a child class.

→ A virtual fc^n is called during Runtime.

(In Java, all non-static methods are virtual fc^n)

Abstraction :- Hiding all unnecessary details & showing only the important parts.

→ Abstract classes :-

→ used to provide a base class from which other classes can be derived

→ They cannot be instantiated and are meant to be inherited.

→ Typically used to define an interface for Derived classes.

→ Pure virtual fc^n :-

```
virtual void draw() = 0;
```

static Keyword :-

→ static variables –

→ Variables declared as static in a fc^n are created & initialised once for the lifetime of the Program. // In fc^n

→ static variables in a class are created & initialised once. They are shared by all the objects of the class. // In class