Object Oriented Programming In Java

- **Object Oriented Programming:** It is a programming paradigm to design a program using classes and objects.
  - It simplifies the software development & maintainance by providing some concepts.
- **Classes:** It is a user defined datatype which defines it's properties & it's functions.
- **Objects:** It is a runtime entity. It is a instance of the class.
  - An object can operate on both data members and member functions.
- **'This' keyword:** In Java it refers to the current instance of the class. Used while,
  - passes the current object as a parameter to another method.
  - refer to the current class instance variable.
- **Constructor:** Constructor is a spacial method which is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally.
  - At the time of calling the constructor, the memory is allocated for the object.
  - 3 tipes of constructor are there.
  (i) Parameterized (ii) Non-Parameterized (iii) default.
- **Polymorphism:** Polymorphism is the ability to present the same interface for differing underlying forms.
  - With polymorphism, each of the classes will have different underlying data. Precisely 'Poly' means 'many' and 'morphism' means 'forms'.

Types: (i) Compile time (Static) (ii) Runtime (Dynamic)

(i) Compiletime Polymorphism: Implemented at the time of compilation. It is known as static Polymorphism.
example: method overloading.

Method overloading: By this technique we are allowed to have more than one function or method with the same name but with different functionalities. It can be done by either (i) the type of parameters passed are different or (ii) the number of parameters passed are not same.

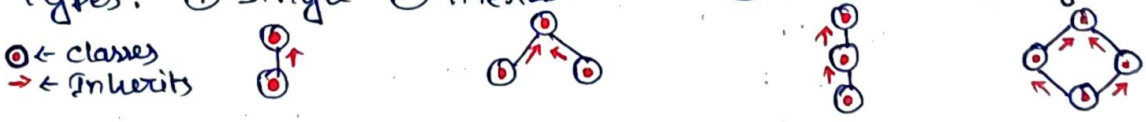
(ii) Runtime Polymorphism: It is also known as Dynamic Polymorphism.
example: function overriding.

Function overriding: Child ~~call~~ class override the parent class, it is known as function overriding.

- Inheritance: Inheritance is a process in which one object acquires all the properties and behaviours of it's parent Object automatically using 'extends' keyword.

* the class which inherits → Derived class/child class.
* the class which is inherited → Base class/Parent class.

Types: (i) Single (ii) Hierarchical (iii) Multilevel (iv) Hybrid.



- Packages: Group of similer types of classes, interfaces and sub-packages. It can be in built or user defined.
use 'import' keyword to use them.

- Access Modifiers: where can be used defined by access modifiers.

	class	Packages	Subclasses	World.
Private:	Yes	NO	NO	NO
Protected:	Yes	Yes	Yes	NO
Default:	Yes	Yes	NO	NO
Public:	Yes	Yes	Yes	Yes

- **Encapsulation**: Encapsulation is the process of combining data and functions into single class unit, the data is kept private not accessed directly & aparate setter & getter methods are provided to manipulate the data.
  * encapsulation makes the concept of ~~encapsulation~~ Data hiding possible.

Data hiding: A language feature to restrict access to members of an object. using 'private', 'protected' keyword we can achive that in java.

- **Abstraction**: Hiding the unnecessary details and showing only the essential parts / functionalities to the specific user. achived using 'abstract' keyword.

It is achived by two ways - (in java)

① abstract class.
② Interfaces (Pure abstraction)

① **Abstract class**:
- must be declared using abstract key word.
- can have abstract & non abstract methods.
- abstract methods must have to explained in child classes. (body)
- It can not be initiated.
- Can have final methods which will force the child classes not to chage the body of the method.

- **Interfaces**:
  - All the fields in the interfaces are public static & final by default.
  - All methods are public abstract by default.
  - A class that implements an interface must implements all methods declared in the interface.
  - Interface supports the functionalities of multiple inheritance.

- 'Static' keyword: mainly used for memory mangment.
  - Static can be ① variable
    ② Method
    ③ Block
    ④ Nested class.
  - Static variable only gets memory once in the class area during class loding.

- 'Super' keyword:
  - used to reffer immediate parent class object / intance variable.
  - can be used to reffer immediate parent class method.
  - can be used to reffer immediate parent class constructor.

- 'Final' keyword:
  - Used to restrict the user.
  - Final can be ① variable
    ② method
    ③ class.
  - final method inheritated but can't override.