**Object Oriented Programming System (OOPS) using Java**

* **OOPs** : A methodology of coding which divides into classes and objects.
* **Class :** blueprint of objects. Collection of member function and variables.
* **Object:** It is an entity of class. Which has its own properties, functions and variables.
* **Constructor:**  It is used to initialize the data of a object. Automatically called when we create object.

Constructor **Name** and **Class** name should be **same**.

Constructor **does not return any value**. Not even void.

Constructor can be parameterized or non parameterized constructor

* **Overloading**
* **Constructor Overloading**: we can create multiple constructor we call it as constructor overloading.
* **Method Overloading :** If we create multiple method with same name then it called method overloading.

Number of arg may differ

Type of arg may differ

Order of arg may differ

* **This Keyword: Refers current invoking object** 
  + this () -> for calling current class constructor from inside different constructor of same class .

**Inheritance :** By using inheritance one class is allowed to inherit the properties of another class.

The class who inherits the properties is child/derived/sub class. And the class whos properties going to inherit is parent/base/super class.

Eg.

Class Dog extends Animal {}

Types : single Inheritance, Multilevel Inheritance, Hierarchical Inheritance (multiple childs),

Multiple Inheritance (not supported with class but we can use Interfaces ) ,

**Super Keyword**

by using this keyword we can access current class properties whereas by using **super** keyword we can use parent class properties.

We can use **super();** to call parent class constructor. In this case **super();** must be the first statement to execute.

* **Overriding :** 
  + Child class overrides the parent method. And redefines the body of method. Its called method overriding.
  + Here name, parameters basically method signature remains as it is. Only execution code (method body) changes.
* **Encapsulation:** 
  + Wrapping of data members(variables) and member methods in a single unit.
* **Abstraction:** 
  + Showing features and hiding internal details (implementation)
  + Abstract class and Abstract methods
  + We cannot create object of abstract Class. We need to inherit that class and override the method.
* **Interface**:
  + Provides 100% abstraction.
  + All variables are by default public static final
  + All methods are by default public abstract
* Abstract vs Interface
* **Polymorphism :** 
  + Ability to take many forms
  + Compile time(static binding) and Runtime Polymorphism(Dynamic binding)
  + Compile time polymorphism uses overloading.
  + Runtime polymorphism uses overriding.