**\*\*\*OOP’S IN JAVA\*\*\***

**What is the Opp’s?**

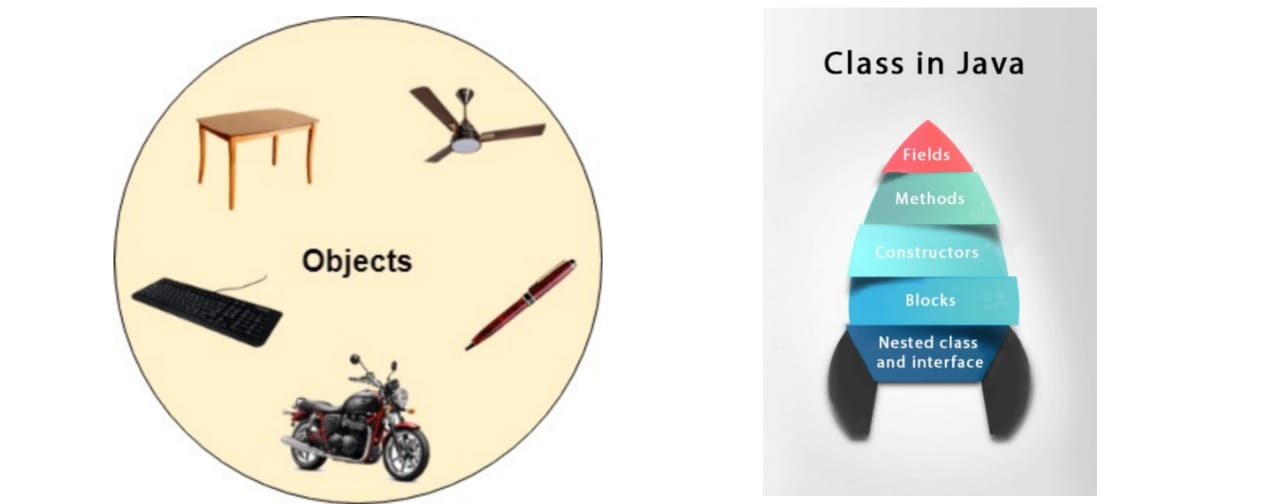
It is a methodology to develop/design a software. The programming language which satisfaction is the oops principal we can called have a object-oriented programming language

**Why do we need to Opps?**

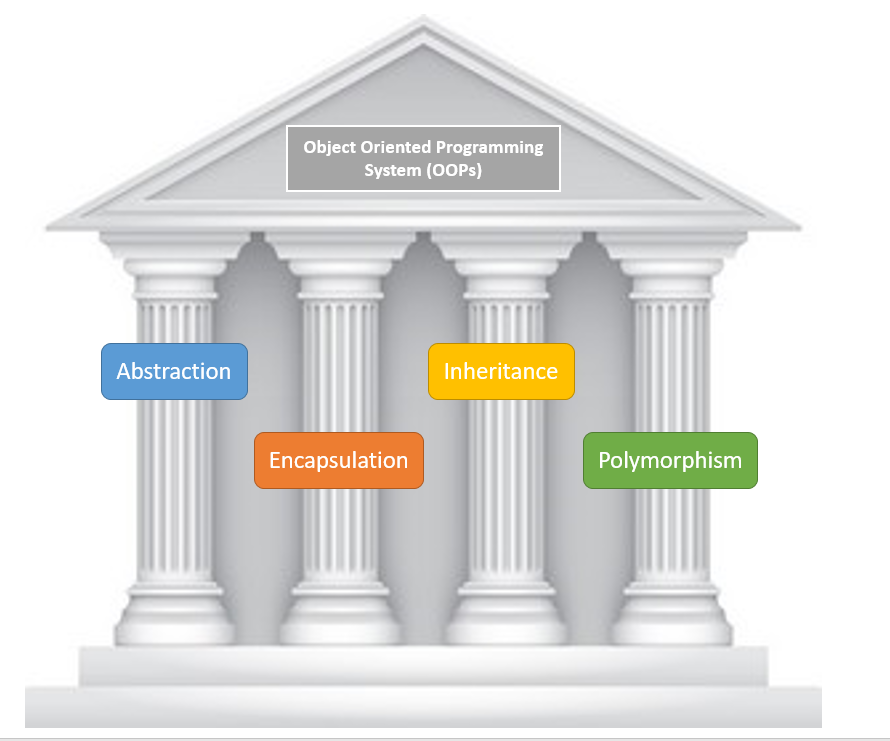
it simplifies the programming and software to develop

**There are two features: -**

1. object
2. class



**there are 4 types of Opps principles:-**



1. Encapsulation
2. Abstraction
3. Polymorphism
4. Inheritance
5. **Encapsulation** :-

**What is Encapsulation?**

It is a oops concept to biding data and number variables/methods to gather into one single unit .it is called as encapsulation.

Data numbers (variables) need to be declared as a ‘private ‘keyword

Data methods (POZZO class) like Using the two setter and getter the methods to Set and get the values

The best example of encapsulation is Pozzo class

Pozzo class means DTO class (Data transfer object)

In Pozzo class or including in three types of concepts

1. Setter method

2. Getter method

3. Constructor

Encapsulation= Data hiding + Abstraction

**Example Program: -**

class Kalyan

{

private String name;

private int id;

private String designation;

public void setId(int id)

{

this.id=id;

}

public void getId(){

System.out.println("setter value"+ id);

}

public void setName(String name)

{

this.name=name;

}

public void getName()

{

System.out.println("setter value"+ name);

}

public void setDesignation(String designation)

{

this.designation=designation;

}

public void getDesignation()

{

System.out.println("setter value"+ designation);

}

public static void main(String[] args){

System.out.println("hello world");

Kalyan k1=new Kalyan();

k1.setId(100);

k1.getId();

k1.setName("geetha");

k1.getName();

k1.setDesignation("prasad");

k1.getDesignation() }

}

1. **Abstraction:-**

What is abstraction?

abstraction is a process off Hiding the functionality and exploring `/visibility Whatever functionality to the particular user

Why do we need to abstraction?

To implementations hiding the functionality with the respective role to the corresponding applications like SBI, LIC…. etc

How To Implement the abstraction?

To declared the abstract method/ class. We can use the ‘abstract’ keywords

Where to use the abstraction?

In our Java application to develop different kinds of components at the time of application development.

Syntax: -

Abstract class classname;//class declaration

Abstract void methodname();//method declaration

Syntax: -

class sh2ape

{

Int colour;

// an abstract function

Abstract void draw ();

}

Example: -

class KalyanAll1

{

public static void main(String[] args)

{

System.out.println("main method");

Mobile m1 = new Mi();

m1.Mi(20,30);

}

}

abstract class Mobile

{

private int watsapp;

private int facebook;

abstract int Mi(int watsapp,int facebook);

}

class Mi extends Mobile

{

private int watsapp;

private int facebook;

int Mi(int watsapp,int facebook)

{

int MiResult=watsapp\*facebook/2;

System.out.println("Result in mi::"+MiResult);

return MiResult;

}

}

1. polymorphism :-

What is polymorphism?

In the concept of the polymorphism which we can perform a single action in different ways. Polymorphism is derived 2 words poly and morphism; poly means many morphisms means forms so polymorphism means many forms

Why do we need to polymorphism?

To write a method than can correctly process lots of different types of functionalities that have the same name

How to implement the polymorphism?

The implement the polymorphism to use the method overloading and method overriding concepts.

There are two types of polymorphism: -

1. method overloading
2. method overriding

1 method overloading: - (compile time polymorphism)

If a class has multiple methods are same name, return type are same, parameters are different it is called method overloading

* method overloading increases the readability of the program
* Two write overriding only 1 class

Types of overloading: -

1. by change numbers of arguments
2. by change the data types

1.by change numbers of arguments: -

class Find

{

void add(int a,int b)

{

System.out.println("resultis"+(a+b));

}

void add(int a,int b,int c)

{

System.out.println("resultis"+(a+b+c));

}

public static void main(String[] args)

{

Find f1 = new Find();

f1.add(2,8);

f1.add(25,54,78);

}

}

2.by change the data types: -

class Find2

{

void add(int a,int b)

{

System.out.println("resultis"+(a+b));

}

void add(long a,long b,long c)

{

System.out.println("results"+(a+b+c));

}

public static void main(String[] args)

{

Find f1 = new Find();

f1.add(2,8);

f1.add(28,94,18);

}

}

1. method overriding: - (Runtime/Dynamic polymorphism)

If a class has multiple methods are same name, return type are same, parameters are same it is called method overriding

* it we can used super keyword.
* Parent class and chilled class are method same name.
* Parent class and chilled class are parameters are same.
* There must be an IS-A relation ship
* Two write overriding 2 class

Example :-

class Apple

{

void cost()

{

System.out.print("1kg apple cost =240");

}

}

class Mango extends Apple

{

void cost()

{

super.cost();

System.out.print("1kg mango cost 160");

}

}

class Fruit

{

public static void main(String[] args)

{

System.out.println("Hello World!");

Mango m1 = new Mango();

m1.cost();

}

}

1. Inheritance :-

What is inheritance?

Inheritance is a kind of the mechanism are process of mechanism where we have going to get are assess the data form the super class to chilled class

Why do we need to inheritance: -

The most important use of inheritance in Java is code reusability. The code that is present in the parent class can be directly used by the child class.

How To Implement the inheritance?

To inherit the parent class, a child class must include a keyword called "extends." The keyword "extends" enables the compiler to understand that the child class derives the functionalities and members of its parent class.

Extends keyword: -

Extends is the one of the keywords among the 54 plus keywords

What is the role and responsibility are Extends keyword?

By using extends keyword you know assess the data numbers are data from the parent class to our chilled class

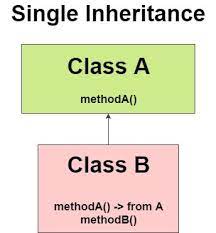
Where to use the inheritance?

The most important use of inheritance in Java is code reusability. The code that is present in the parent class can be directly used by the child class. Method overriding is also known as runtime polymorphism. Hence, we can achieve Polymorphism in Java with the help of inheritance

there are five types are inheritance: -

1. Single inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Multiple inheritance
5. Hybrid inheritance
6. Single inheritance: -

Single inheritance in Java refers to the concept of a subclass inheriting properties and behaviour from a single superclass. In other words, a subclass can extend only one superclass. The subclass is created by using the “extends” keyword followed by the name of the superclass



Example: -

class Animal{

void eat()

{

System.out.println("eating...");

}

}

class Dog extends Animal

{

void bark()

{

System.out.println("barking...");

}

}

class TestInheritance

{

public static void main(String args[]){

System.out.println("Main Method");

Dog d=new Dog();

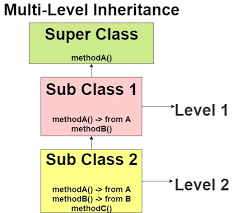
d.bark();

d.eat();

}}

1. Multi-level inheritance: -

In Multi-Level Inheritance in Java, a class extends to another class that is already extended from another class. For example, if there is a class A that extends class B and class B extends from another class C, then this scenario is known to follow Multi-level Inheritance.



Example: -

class Animal

{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class BabyDog extends Dog

{

void weep(){System.out.println("weeping...");

}

}

class TestInheritance2{

public static void main(String args[]){

BabyDog d=new BabyDog();

d.weep();

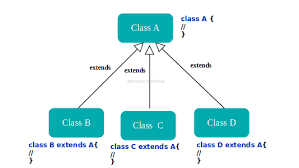
d.bark();

d.eat();

}}

1. Hierarchical inheritance: -

Hierarchical inheritance in Java is a type of inheritance in which the same class is inherited by more than one class. In other words, when several classes inherit their features from the same class, the type of inheritance is said to be hierarchical



Example: -

class A

{

public void methodA()

{

System.out.println("method of Class A");

}

}

class B extends A

{

public void methodB()

{

System.out.println("method of Class B");

}

}

class C extends A

{

public void methodC()

{

System.out.println("method of Class C");

}

}

class D extends A

{

public void methodD()

{

System.out.println("method of Class D");

}}

class JavaExample

{

public static void main(String args[])

{

B obj1 = new B();

C obj2 = new C();

D obj3 = new D();

//All classes can access the method of class A

obj1.methodA();

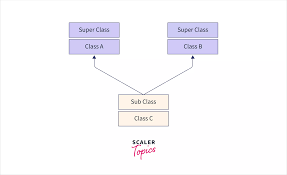
obj2.methodA();

obj3.methodA();

}}

1. Multiple inheritance: -

Multiple inheritance in java is the capability of creating a single class with multiple super classes. Unlike some other popular object-oriented programming languages like C++, java doesn't provide support for multiple inheritance in classes



Example: -

class Bike

{

public void petrol()

{

System.out.println("bike");

}

}

class TwoViler

{

public void petrol()

{

System.out.println("Two viler");

}

}

class Hero extends Bike,TwoViler

{

}

public class Main

{

public static void main(String[] args)

{

Hero obj = new Hero(); // creating object of class hero

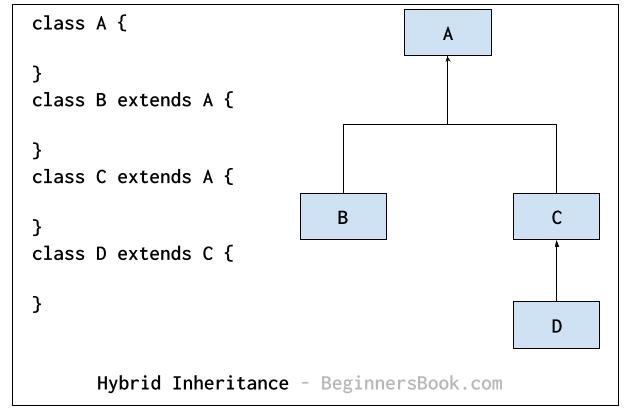
obj.execute(); // petrol() method is present in both class Bike and TwoViler

}

}

1. Hybrid inheritance: -

Hybrid Inheritance in Java is a type of inheritance where a class extends two or more classes, where one or more of them is a combination of different types of inheritance. This means that the child class inherits properties and behaviours from both parent classes, including any intermediate classes.



Example: -

class HumanBody

{

public void displayHuman()

{

System.out.println("Method defined inside HumanBody class");

}

}

interface Male

{

public void show();

}

interface Female

{

public void show();

}

public class Child extends HumanBody implements Male, Female

{

public void show()

{

System.out.println("Implementation of show() method defined in interfaces Male and Female");

}

public void displayChild() {

System.out.println("Method defined inside Child class"); }

public static void main(String args[])

{

Child obj = new Child();

System.out.println("Implementation of Hybrid Inheritance in Java");

obj.show();

obj.displayChild();

}

}