**Oops(objected oriented programming language)**

The popular object-oriented languages are [Java](https://www.javatpoint.com/java-tutorial), [C#](https://www.javatpoint.com/c-sharp-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), [Python](https://www.javatpoint.com/python-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc.

The main aim of object-oriented programming is to implement real-world entities.

Now I am going to prefer java to learn the object oriented programming concepts.

The basic oops concept such as inheritance, polymorphism, abstraction, encapsulation.

**\*\*class and objects:**

**Creating object for class**

**class** Student{

System.out.println(“first class”);

}

**class** TestStudent2{

**public** **static** **void** main(String args[]){

  Student s1=**new** Student();

 }

}

**Initializing object of the class**

**class** Student{

**int** id;

 String name;

}

**class** TestStudent2{

**public** **static** **void** main(String args[]){

  Student s1=**new** Student();

  s1.id=101;

  s1.name="Sonoo";

  System.out.println(s1.id+" "+s1.name);

 }

}

**Construction :**

**class** Student4{

**int** id;

    String name;

    Student4(**int** i,String n){

    id = i;

    name = n;

    }

**void** display(){System.out.println(id+" "+name);}

**public** **static** **void** main(String args[]){

    Student4 s1 = **new** Student4(111,"Karan");

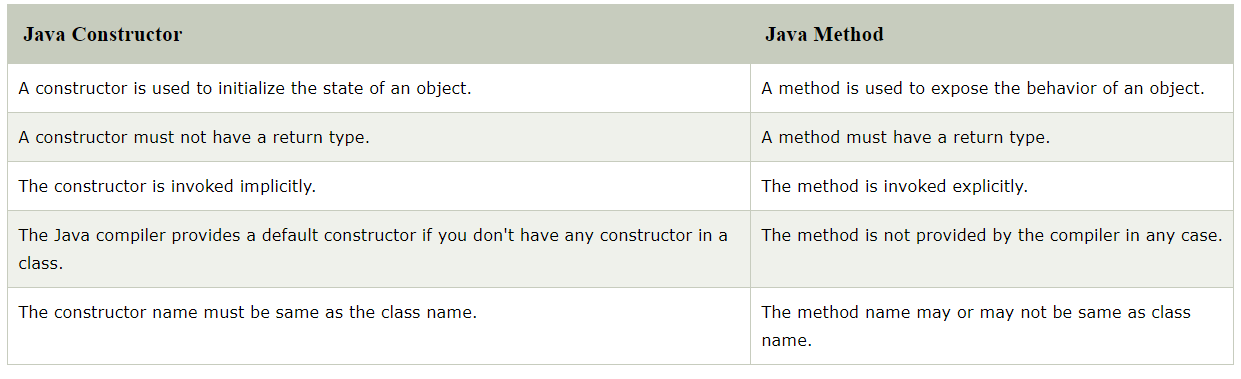
    Student4 s2 = **new** Student4(222,"Aryan");

        s1.display();

    s2.display();

   }

}

**D/B constructor and method** 

**Inheritance:**

**Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object.

Single Inheritance Example

**lass** 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[]){

Dog d=**new** Dog();

d.bark();

d.eat();

}}

## Multilevel 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();

}}

## Why multiple inheritance is not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

## Hierarchical Inheritance Example

**class** Animal{

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

}

**class** Dog **extends** Animal{

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

}

**class** Cat **extends** Animal{

**void** meow(){System.out.println("meowing...");}

}

**class** TestInheritance3{

**public** **static** **void** main(String args[]){

Cat c=**new** Cat();

c.meow();

c.eat();

//c.bark();//C.T.Error

}}

**Aggregation :**

**class** Operation{

**int** square(**int** n){

**return** n\*n;

 }

}

**class** Circle{

 Operation op;

**double** pi=3.14;

**double** area(**int** radius){

   op=**new** Operation();

**int** rsquare=op.square(radius);

**return** pi\*rsquare;

 }

**public** **static** **void** main(String args[]){

   Circle c=**new** Circle();

**double** result=c.area(5);

   System.out.println(result);

 }

}

**Polymorphism :**

Polymorphism can be done by using method overloading and method over hiding.

### **Overloading**

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

### **Different ways to overload the method**

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

**Changing the data type :**

**class** Adder{

**static** **int** add(**int** a, **int** b){**return** a+b;}

**static** **double** add(**double** a, **double** b){**return** a+b;}

}

**class** TestOverloading2{

**public** **static** **void** main(String[] args){

System.out.println(Adder.add(11,11));

System.out.println(Adder.add(12.3,12.6));

}}

**Changing parameter size:**

**class** Adder{

**static** **int** add(**int** a,**int** b){**return** a+b;}

**static** **int** add(**int** a,**int** b,**int** c){**return** a+b+c;}

}

**class** TestOverloading1{

**public** **static** **void** main(String[] args){

System.out.println(Adder.add(11,11));

System.out.println(Adder.add(11,11,11));

}}

**Overhiding :**

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

### **Usage of Java Method Overriding**

* Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
* Method overriding is used for runtime polymorphism

#### **Rules for Java Method Overriding**

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

**class** Bank{

**int** getRateOfInterest(){**return** 0;}

}

**class** SBI **extends** Bank{

**int** getRateOfInterest(){**return** 8;}

}

**class** ICICI **extends** Bank{

**int** getRateOfInterest(){**return** 7;}

}

**class** AXIS **extends** Bank{

**int** getRateOfInterest(){**return** 9;}

}

**class** Test2{

**public** **static** **void** main(String args[]){

SBI s=**new** SBI();

ICICI i=**new** ICICI();

AXIS a=**new** AXIS();

System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());

System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());

System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());

}

}

**Abstraction :**

abstract class Shape

{

    String color;

    abstract double area();

    public abstract String toString();

        public Shape(String color) {

        System.out.println("Shape constructor called");

        this.color = color;

    }

    public String getColor() {

        return color;

    }

}

class Circle extends Shape

{

    double radius;

    public Circle(String color,double radius) {

        super(color);

        System.out.println("Circle constructor called");

        this.radius = radius;

    }

    @Override

    double area() {

        return Math.PI \* Math.pow(radius, 2);

    }

    @Override

    public String toString() {

        return "Circle color is " + super.color +

                       "and area is : " + area();

    }

}

class Rectangle extends Shape{

    double length;

    double width;

    public Rectangle(String color,double length,double width) {

        super(color);

        System.out.println("Rectangle constructor called");

        this.length = length;

        this.width = width;

    }

    @Override

    double area() {

        return length\*width;

    }

    @Override

    public String toString() {

        return "Rectangle color is " + super.color +

                           "and area is : " + area();

    }

}

public class Test

{

    public static void main(String[] args)

    {

        Shape s1 = new Circle("Red", 2.2);

        Shape s2 = new Rectangle("Yellow", 2, 4);

        System.out.println(s1.toString());

        System.out.println(s2.toString());

    }

}

**Interface :**

**Multilevel iheritance:**

**interface** Printable{

**void** print();

}

**interface** Showable **extends** Printable{

**void** show();

}

**class** TestInterface4 **implements** Showable{

**public** **void** print(){System.out.println("Hello");}

**public** **void** show(){System.out.println("Welcome");}

**public** **static** **void** main(String args[]){

TestInterface4 obj = **new** TestInterface4();

obj.print();

obj.show();

 }

}

**Multiple inheritance:**

**interface** Printable{

**void** print();

}

**interface** Showable{

**void** show();

}

**class** A7 **implements** Printable,Showable{

**public** **void** print(){System.out.println("Hello");}

**public** **void** show(){System.out.println("Welcome");}

**public** **static** **void** main(String args[]){

A7 obj = **new** A7();

obj.print();

obj.show();

 }

}

**Encapsulation :**

**package** com.javatpoint;

**public** **class** Student{

**private** String name;

**public** String getName(){  //Getter

**return** name;

}

**public** **void** setName(String name){  //setter

**this**.name=name  }}