=> Polymorphism :-

-> Poly (many) + Morhism (forms, structure)

-> Real world example : water, many brand clothes in one shop, person(doctor, engineer etc), single institute has multiple trainer etc

-> Advantage :- It provides the flexibility to develop an application i.e. it allows us to perform a single task by different ways.

-> Types of Polymorphism :-

1. Compile Time Polymorphism
2. Runtime Polymorphism

=> Compile Time Polymorphism :-

-> It is also known as Static Polymorphism or Early Binding

-> If the polymorphism is achieved at compile time then it is compile time polymorphism

=> Runtime Polymorphism :-

-> It is also known as Dynamic Polymorphism or Late Binding

-> If the polymorphism is existed at runtime then it is known as Runtime Polymorphism

-> Runtime Polymorphism can be achieved by "Method Overriding"

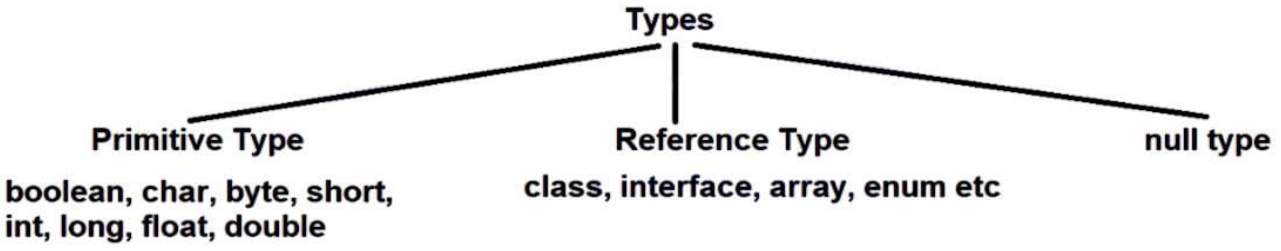
=> Method Overloading :-

-> The process of compiler trying to resolve the method call based on reference type is known as method overloading

-> Rules for method overloading :-

1. Same name
2. Within same class
3. Different parameters
    - -> No of parameters
    - -> Type of parameters

What is reference type ?




```
int a=10;
```

```
class Test
{
        psvm()
        {
                Test t = new Test();
        }
}
```

What is parameter and argument ?

```
class Test
{
        void show(int a)
        {
                //coding
        }
        public static void main(String[] args)
        {
                Test t=new Test();
                t.show(20);
        }
}
```

Parameter

Arguments

```
public class Test {
    void sum(){
        System.out.println("A");
    }
    void sum(int a){
        System.out.println("B");
    }
    void sum(int a, int b){
        System.out.println("C");
    }
    void sum(float a, float b){
        System.out.println("D");
    }
}

public class MehtodOverriding1 {
    public static void main(String[] args) {
        Test t = new Test() ;
        t.sum();
        t.sum(10);
        t.sum(10,20) ;
        t.sum(10.2f, 20.4f);
    }
}
```

==========

```
public class Test {
    void display(int a){
        System.out.println("Hi from int");
    }
    void display(String a){
        System.out.println("hi from String");
    }
}

public class MehtodOverriding1 {
    public static void main(String[] args) {
        Test t = new Test() ;
        t.display('D');
    }
}
```

==========

varargs:

```
public class Test {
    void display(int a){
        System.out.println("Hi from int");
    }
    void display(int... a){
        System.out.println("hi from varargs");
    }
}

public class MehtodOverriding1 {
    public static void main(String[] args) {
        Test t = new Test() ;
        t.display();
        t.display(10);
        t.display(10,20);
    }
}
```

==========

```
public class Test {
    void display(int a){
        System.out.println("Hi from int");
    }
    void display(int... a){
        for(int i:a){
            System.out.println(i);
        }
    }
}

public class MehtodOverriding1 {
    public static void main(String[] args) {
        Test t = new Test() ;
        t.display();
        t.display(10);
        t.display(10,20,30,40);
    }
}
```

==========

```
public class Test {
    void display(Object a){
        System.out.println("1");
    }
    void display(String a){
            System.out.println("2");
    }
}

public class MehtodOverriding1 {
    public static void main(String[] args) {
        Test t = new Test() ;
        t.display(null);          ==> Give preference to child class
    }
}
```

==========

Interview Questions :-

1. Operator Overloading is not supported in java but + operator is overloaded

-> Operator Overloading concept is achieved only by java designers but it cannot be achieved by developers like us.

2. What is difference between Compile Time Polymorphism & Runtime Polymorphism

3. What is varargs

4. Can we overload main method -> Yes

```
public class MehtodOverriding1 {
    public static void main(String[] args) {
        System.out.println("1");
        MehtodOverriding1 obj = new MehtodOverriding1() ;
        int [] arr = {1,2,3,4} ;
        obj.main(arr) ;
    }

    public static void main(int [] arr) {
        System.out.println("2");
    }
}
```

5. Can we overload constructors -> Yes

```
class Test
{
   Test()
   {
      System.out.println("1");
   }
   Test(int a)
   {
      System.out.println("2");
   }
}
class MethodOverloading13
{
   public static void main(String[] args)
   {
      new Test();
      Test t2=new Test(10);
   }
}
```