Polymorphism

- Poly means many
- Morphism means forms or behaviors

“Polymorphism” is the process of exhibiting more than one form. Or

“polymorphism” ability to generate more than one form.

“Polymorphism” plays an important role in allowing multiple Objects to have different internal structural and by sharing the same external interface.

For an Example an operator “+” in java is polymorphic natured operator. i.e the behavior of the operator is getting varying in different situations.

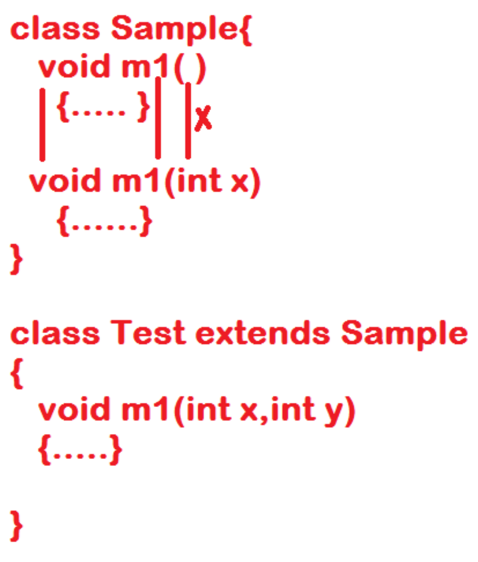
- If we pass two integers as an operands then internally it will perform an addition and produce the sum Eg: 10+20 -> 30
- If we pass two strings as an operands then internally it will perform concatenation and produce the concatenated output. [“sai”+”baba” –> “saibaba”]

- Polymorphism is classified into 2 Types
  - Compile-time Polymorphism
  - Run-Time Polymorphism

- Compile-Time Polymorphism:
    - In compile time polymorphism always takes place based on reference of the Object
    - In compile time polymorphism memory decision will done at that time of compilation
        - Example : method overloading
- **Run-time Polymorphism:**
    - **In Runtime polymorphism always takes place based on object we are passing to the reference**
    - In Runtime polymorphism the decision will be done at that time of execution
        - Example: Method Overriding

- What is difference between overloading and Overriding ?

  Overloading :
    - It is the process of defining more than one method with the same name for different purpose
    - In method overloading , method name must be same, list of argument should not be same,return type may or may not be same
    - Overloading can be done either in the same class or in its inherited class



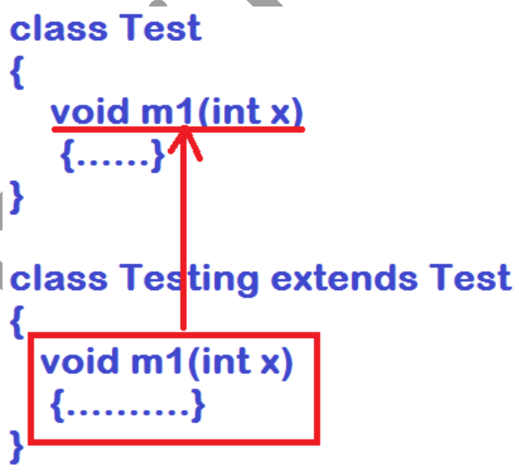
```
class Sample{
  void m1( )
   {..... }
  void m1(int x)
   {......}
}

class Test extends Sample
{
  void m1(int x,int y)
  {.....}

}
```

Overriding :

- It is the process of defining sub class method same as super class method
- In overriding method name must be same, list of arguments must be same ,and return type must be same
- Overriding is possible only in its inherited class [is-a relationship]



```
class Test
{
   void m1(int x)
    {......}
}

class Testing extends Test
{
  void m1(int x)
   {..........}
}
```

When do we go for overriding ?

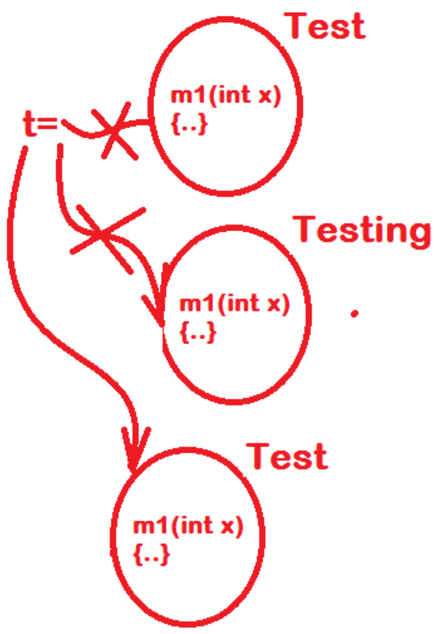
- Whenever you want use super class method in the sub class with different implementation , then will go for overriding

Example on Method Overriding :

```
class Test
{
   void m1(int x)
    { S.o.pln("M1 of Test"); }
}
class Testing extends Test
{
   void m1(int x)
    { S.o.pln("M1 of Testing "); }
}
class OREx1{
  p s v main(String args[ ]){
   Test t=new Test( );
            t.m1(111);

            t=new Testing( );
            t.m1(222);

            t=new Test();
  }         t.m1(111);
}
```



Note:

- **The super class reference variable can hold the same class object.and it can also hold any of it's subclass Object implicitly called upcasting or generalization**

```
class SuperA
{ }
class SubB extends SuperA
{ }
class Demo
{
    public static void main(String args[ ])
    {
            SuperA a=new SuperA( );
            SubB b=new SubB();
           a=b;  //SubB object we are assigning to SuperA ref
    }
}
```

- **A SubClass reference variable can hold the object of same class . but it can't hold the object of it's super class implicitly.**
- But we can make it possible explicitly using **"downcasting" or specialization**

```
class SuperA
{ }
class SubB extends SuperA
{ }
class Demo{
        public static void main(String args[ ])  {
                SuperA a=new SuperA( );
                SubB b=new SubB();
               // b=a;  we are assigning SuperA Object to the
                                              //reference of subB  CE
                                    b=(SubB)a;
        }
}
```

Imp How many methods in Class “A”:

Ans : all the methods of java.lang.Object class.

<u>Object is the super class for every class In java.</u>

```
//A.java

class A{ }

//javac A.java

>JDK1.6

//javap A

  Compiled From “A.java”

  Class A extends java.lang.Object {   //it is the super class for every
object

       A();

  }
```

```
//TestIQ
//  primitive datatype --> wrapper classes
//   int   --> Integer
//   float  ---> Float
class TestIQ
{
        public static void main(String args[ ])
        {
                Object o = new Object( );
                Integer i=new Integer(10);
                Float f=new Float(3.14f);
                String s=new String("Shashi");

                o=i;  //posibile when Integer is sub class on Object
                o=f;
                o=s;

                i=(Integer)o;
                f=(Float)o;
                s=(String)o;

                Number n;
                n=i;  //it is posibile when Integer is the sub class of Number
                                //javap java.lang.Integer
                n=f;  //javap   java.lang.Float
                n=s;
        }
}
```

```
//javap java.lang.Object
class Testing
{
        int x,y;

        Testing(int x,int y)
        { this.x=x; this.y=y; }

        public static void main(String args[])
        {
                Testing t1=new Testing(10,20);
                Testing t2=new Testing(10,20);

                if( t1.equals(t2) )
                { System.out.println("Both Are same "); }
                else
                { System.out.println("Both Are Not Same "); }
        }
}
```

When do we go method overriding ?

- Whenever you want use the super class method in the sub class with different implementation
- java.lang.Object is the super class for Every class in java

Can u override predefined methods in Java?

- Yes,

<u>public boolean equals(java.lang.Object)</u> from java.lang.Object class which is meant for comparing hash code of two object, the following program demo how to override equals( ) of Object to compare content of two object

```
//java.lang.Object
  //public boolean equals(Object)
 class Testing                    //super class for Testing -> Object
 {
        int x,y; //instance fields

        Testing(int x,int y)
           { this.x=x; this.y=y; }

       public boolean equals(Object o) //non static or instance
               {   Testing obj=(Testing)o;
                       if(x==obj.x &&  y==obj.y)
                         return true; else return false; }

        public static void main(String args[])
        {
                Testing t1=new Testing(10,20);
                Testing t2=new Testing(10,20);

                if(t1.equals(t2))
                        System.out.println("Both Are Same ");
                else
                        System.out.println("Both Are Not Same");
        }
 }
```