POLYMORPHISM

- **Polymorphism is one of the core concepts of Object-Oriented Programming OOPS**
- **The word polymorphism means (many forms) as many behaviour**
- **In polymorphism allows objects to be instances of their parent class rather than their actual class, and the same method name can behave differently based on the object**
- **It means with object we can execute a multiple behaviours using polymorphism**

Such as Example: object is shape but in shape we have many types like circle, triangle rectangle, square etc... Each shapes specific width, height, radius and perimeter...

In polymorphism we have two types:

Compile time polymorphism it's called as Method overloading

Same class, same method name, different parameters (arguments)

```
package javaprogram;
class additions
{
      // method over loading (same class,same method
name, different parameter)
    // without parameter(arguments)
       void add()
       {
       int a=10; int b=20;
       int c=a+b;
       System.out.println(c);
       }
    // with single parameter(arguments)
       void add(int c)
       {
       int a=10; int b=20;
```

```
        int d=a+b+c;
        System.out.println(d);
        }
        // with double parameter(arguments)
        void add(int c,int d)
        {
        int a=10; int b=20;
        int e=a+b+c+d;
        System.out.println(e);
        }
}
public class TopicPolymorphism {

          public static void main(String[] args) {

          additions addsvalue=new additions(); //
compile time polymorphism
          addsvalue.add(); // calling without
parameter(arguments)
          addsvalue.add(30);// calling with single
parameter(arguments)
          addsvalue.add(40, 50);// calling with
double parameter(arguments)
          }
    }
```

Runtime polymorphism it's called as Method overriding

Different class, same method name, same parameters (arguments)

```
package javaprogram;
class additions
{
     // method overriding this methods override to
subtraction for subtraction class
    // without parameter(arguments)
       void add()
```

```
        {
        int a=10; int b=20;
        int c=a+b;
        System.out.println(c);
        }
    // with single parameter(arguments)
        void add(int c)
        {
        int a=10; int b=20;
        int d=a+b+c;
        System.out.println(d);
        }
        // with double parameter(arguments)
        void add(int c,int d)
        {
        int a=10; int b=20;
        int e=a+b+c+d;
        System.out.println(e);
        }
}

class subtraction extends additions
{
     // method over riding (different class,same
method name, same parameter)
    // override from class additions without
parameter(arguments)
        void add()
        {
        int a=10; int b=20;
        int c=a-b;
        System.out.println(c);
        }
    // override from class additions with single
parameter(arguments)
        void add(int c)
        {
```

```
        int a=10; int b=20;
        int d=a-b-c;
        System.out.println(d);
        }
    // override from class additions with double
parameter(arguments)
        void add(int c,int d)
        {
        int a=10; int b=20;
        int e=a-b-c-d;
        System.out.println(e);
        }
}
public class TopicPolymorphism {

          public static void main(String[] args) {

          additions addsvaluetosub=new
subtraction(); // runtime polymorphism
          addsvaluetosub.add(); // calling without
parameter(arguments)
          addsvaluetosub.add(30);// calling with
single parameter(arguments)
          addsvaluetosub.add(40, 50);// calling with
double parameter(arguments)
          }
    }
```

Note: The both class should be in the relationship using a extends keyword that's define the current behaviour to execute

If we need to access both class properties we can create object for parent class additions

```
package javaprogram;
class additions
{
     // method overriding this methods override to
subtraction for subtraction class
    // without parameter (arguments)
       void add()
       {
       int a=10; int b=20;
       int c=a+b;
       System.out.println(c);
       }
    // with single parameter (arguments)
       void add(int c)
       {
       int a=10; int b=20;
       int d=a+b+c;
       System.out.println(d);
       }
       // with double parameter (arguments)
       void add(int c,int d)
       {
       int a=10; int b=20;
       int e=a+b+c+d;
       System.out.println(e);
       }
}

class subtraction extends additions
{
     // method over riding (different class,same
method name, same parameter)
    // override from class additions without
parameter(arguments)
```

```
        void add()
        {
        int a=10; int b=20;
        int c=a-b;
        System.out.println(c);
        }
    // override from class additions with single
parameter(arguments)
        void add(int c)
        {
        int a=10; int b=20;
        int d=a-b-c;
        System.out.println(d);
        }
    // override from class additions with double
parameter(arguments)
        void add(int c,int d)
        {
        int a=10; int b=20;
        int e=a-b-c-d;
        System.out.println(e);
        }
}
public class TopicPolymorphism {

          public static void main(String[] args) {

          additions addsvaluetosub=new
subtraction(); // runtime polymorphism
          addsvaluetosub.add(); // calling without
parameter(arguments)
          addsvaluetosub.add(30);// calling with
single parameter(arguments)
          addsvaluetosub.add(40, 50);// calling with
double parameter(arguments)

          additions addsvalue=new additions();
```

```
        addsvalue.add(); // calling without
parameter(arguments)
        addsvalue.add(60);// calling with single
parameter(arguments)
        addsvalue.add(70, 80);// calling with
double parameter(arguments)
         }
    }
```

Another option to we can access parent class property using a super keyword

```
package javaprogram;
class additions
{
     // method overriding this methods override to
subtraction for subtraction class
    // without parameter(arguments)
       void add()
       {
       int a=10; int b=20;
       int c=a+b;
       System.out.println(c);
       }
    // with single parameter(arguments)
       void add(int c)
       {
       int a=10; int b=20;
       int d=a+b+c;
       System.out.println(d);
       }
       // with double parameter(arguments)
       void add(int c,int d)
       {
       int a=10; int b=20;
       int e=a+b+c+d;
       System.out.println(e);
```

```
        }
}

class subtraction extends additions
{
     // method over riding (different class,same
method name, same parameter)
    // override from class additions without
parameter(arguments)
        void add()
        {
        int a=10; int b=20;
        int c=a-b;
        System.out.println(c);
        super.add();
        }
    // override from class additions with single
parameter(arguments)
        void add(int c)
        {
        int a=10; int b=20;
        int d=a-b-c;
        System.out.println(d);
        super.add(100);
        }
    // override from class additions with double
parameter(arguments)
        void add(int c,int d)
        {
        int a=10; int b=20;
        int e=a-b-c-d;
        System.out.println(e);
        super.add(200, 300);
        }
}
public class TopicPolymorphism {
```

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

        additions addsvaluetosub=new
subtraction(); // runtime polymorphism
        addsvaluetosub.add(); // calling without
parameter(arguments)
        addsvaluetosub.add(30);// calling with
single parameter(arguments)
        addsvaluetosub.add(40, 50);// calling with
double parameter(arguments)

        }
    }
```