=> Method Overriding :-

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

-> Overriding is the feature by which child class trying to change the implementation of parent class method

-> Rules for method overriding :-

1. Same name
2. Within different class
3. Same parameters
   - -> No of parameters
   - -> Type of parameters
   - -> Sequence of parameters
4. IS-A relationship

==========

When Parent and Child have the different Methods:

```
public class Parent {
    void show1(){
        System.out.println("Namaste");
    }
}

public class Child extends Parent{
    void show2(){
        System.out.println("What's UP");
    }
}

public class OverridingMain {
    public static void main(String[] args) {
        Parent parent = new Parent() ;
        parent.show1();

        Child child = new Child() ;
        child.show2();

        Parent obj = new Child() ;
        obj.show1();    ==> call the parent method
        //obj.show2() ;   //not possible
    }
}
```

==========

When Parent and Child have the same Method:

```
public class Parent {
    void show(){
        System.out.println("Namaste");
    }
}

public class Child extends Parent{
    void show(){
        System.out.println("What's UP");
    }
}

public class OverridingMain {
    public static void main(String[] args) {
        Parent parent = new Parent() ;
        parent.show();

        Child child = new Child() ;
        child.show();

        Parent obj = new Child() ;
        obj.show();    ==> call the Child method this time
    }
}
```

==========

```
public class Parent {
    void show(int a){
        System.out.println(a);
    }
}

public class Child extends Parent{
    void show(int a){
        System.out.println(a);
    }
}

public class OverridingMain {
    public static void main(String[] args) {
        Parent obj = new Child() ;
        obj.show(20);
    }
}
```

==========

-> Cases for method overriding :

1. If we change the return type in method overriding then it will provide compile time error

```
public class Parent {
    void show(int a){
        System.out.println(a);
    }
}

public class Child extends Parent{
    int show(int a){
        System.out.println(a);
    }
}

public class OverridingMain {
    public static void main(String[] args) {
        Parent obj = new Child() ;
        obj.show(20);
    }
}
```

2. Child class method should have equal or higher access modifier as compared to parent method access modifier in method overriding

```
public class Parent {
    void show(int a){
        System.out.println(a);
    }
}

public class Child extends Parent{
    public void show(int a){
        System.out.println(a);
    }
}
```

3. We cannot override private, final and static methods

```
public class Parent {
    private void show(int a){
        System.out.println(a);
    }
}

public class Child extends Parent{
    public void show(int a){
        System.out.println(a);
    }
}
```

4. We cannot override constructors
5. We cannot override main method

=> Typecasting : The process of converting data type into another is known as typecasting

=> Object Typecasting :

-> The process of converting one object into another object is known as Object Typecasting

-> Object typecasting is of 2 types :-

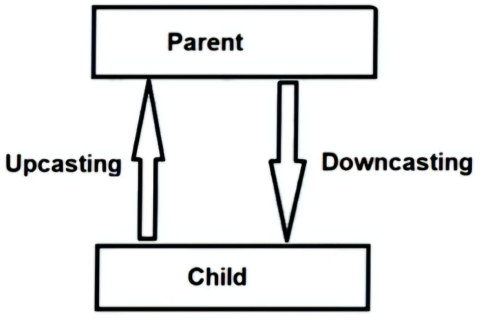
1. Upcasting
2. Downcasting

```
class A
{

}
class B extends A
{

}
class Test
{
        public static void main(String[] args)
        {
                A ob=new B();
        }
}
```

implicit upcasting




```
class A
{

}
class B extends A
{

}
class Test
{
        public static void main(String[] args)
        {
                B ob=new A();   //error
        }
}
```

implicit downcasting

```
A ob = new B();
B ob1 = (B) ob;
```

Explicit downcasting

What is upcasting & downcasting ?

-> Upcasting : Object typecasting in which child object is typecasted into parent object

Downcasting : Object typecasting which parent object is typecasted into child object

-> Upcasting : Implicit upcasting is possible

Downcasting : Implicit downcasting is not possible but forcefully we can do i.e. explicit downcasting is possible