**Method overriding in java**

Declaring a method in **sub class** which is already present in **parent class** is known as method overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. In this case the method in parent class is called overridden method and the method in child class is called overriding method.

The main advantage of method overriding is that the class can give its own specific implementation to an inherited method **without even modifying the parent class code**.

This is helpful when a class has several child classes, so if a child class needs to use the parent class method, it can use it and the other classes that want to have different implementation can use overriding feature to make changes without touching the parent class code.

Method Overriding is an example of [**runtime polymorphism**](https://beginnersbook.com/2013/04/runtime-compile-time-polymorphism/) or dynamic method dispatch

Method overriding performs only if two classes have **is-a** relationship. It means class must have inheritance. In other words, It is performed between two classes using inheritance relation.

In overriding, method of both classes **must** have **same name** and equal number of **parameters**.

Method overriding is also referred to as **runtime polymorphism** because calling method is decided by JVM during runtime.

1. Method name must be same for both parent and child classes.

2. Access modifier of child method must not restrictive than parent class method.

3. Private, final and static methods cannot be overridden.

4. There must be an IS-A relationship between classes (inheritance).

## **Rules of method overriding in Java**

1. Argument list: The argument list of overriding method (method of child class) must match the Overridden method (the method of parent class). The data types of the arguments and their sequence should exactly match.
2. [**Access Modifier**](https://beginnersbook.com/2013/05/java-access-modifiers/) of the overriding method (method of subclass) cannot be more restrictive than the overridden method of parent class. For e.g., if the Access Modifier of parent class method is public then the overriding method (child class method) cannot have private, protected and default Access modifier, because all of these three access modifiers are more restrictive than public.
3. However, this is perfectly valid scenario as public is less restrictive than protected. Same access modifier is also a valid one.
4. private, static and final methods cannot be overridden as they are local to the class. However static methods can be re-declared in the sub class, in this case the sub-class method would act differently and will have nothing to do with the same static method of parent class.
5. Overriding method (method of child class) can throw [**unchecked exceptions**](https://beginnersbook.com/2013/04/java-checked-unchecked-exceptions-with-examples/), regardless of whether the overridden method (method of parent class) throws any exception or not. However, the overriding method should not throw [**checked exceptions**](https://beginnersbook.com/2013/04/java-checked-unchecked-exceptions-with-examples/) that are new or broader than the ones declared by the overridden method. We will discuss this in detail with example in the upcoming tutorial.
6. Binding of overridden methods happen at runtime which is known as [**dynamic binding**](https://beginnersbook.com/2013/04/java-static-dynamic-binding/).
7. If a class is extending an [**abstract class**](https://beginnersbook.com/2013/05/java-abstract-class-method/) or implementing an [**interface**](https://beginnersbook.com/2013/05/java-interface/) then it has to override all the abstract methods unless the class itself is a abstract class.

class Human{

//Overridden method

public void eat()

{

System.out.println("Human is eating");

}

}

class Boy extends Human{

//Overriding method

public void eat(){

System.out.println("Boy is eating");

}

public static void main( String args[]) {

Boy obj = new Boy();

//This will call the child class version of eat()

obj.eat();

}

}

## Super keyword in Method Overriding

The [**super keyword**](https://beginnersbook.com/2014/07/super-keyword-in-java-with-example/) is used for calling the parent class method/constructor. super.myMethod() calls the myMethod() method of base class while super() calls the [**constructor**](https://beginnersbook.com/2013/03/constructors-in-java/) of base class. Let’s see the use of super in method Overriding.  
As we know that we we override a method in child class, then call to the method using child class object calls the overridden method. By using super we can call the overridden method as shown in the example below:

class ABC{

public void myMethod()

{

System.out.println("Overridden method");

}

}

class Demo extends ABC{

public void myMethod(){

//This will call the myMethod() of parent class

super.myMethod();

System.out.println("Overriding method");

}

public static void main( String args[]) {

Demo obj = new Demo();

obj.myMethod();

}

}

Output:

Class ABC: mymethod()

Class Test: mymethod()

As you see using super keyword, we can access the overriden method.