Method Overriding in Java

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding

In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class

Method overriding is one of the way by which java achieve [Run Time Polymorphism](https://www.geeksforgeeks.org/dynamic-method-dispatch-runtime-polymorphism-java/).The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed. In other words, it is the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed.

### **Usage of Java Method Overriding**

* Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
* Method overriding is used for runtime polymorphism

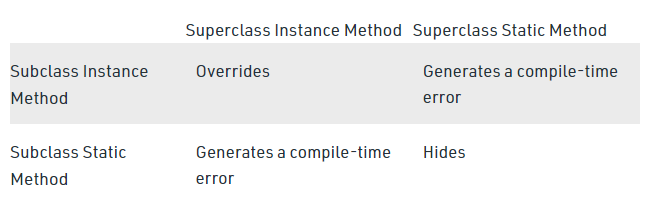
### **Rules for method overriding**

**Overriding and Access-Modifiers :** The [access modifier](https://www.geeksforgeeks.org/access-modifiers-java/) for an overriding method can allow more, but not less, access than the overridden method. For example, a protected instance method in the super-class can be made public, but not private, in the subclass. Doing so, will generate compile-time error.

**Final methods can not be overridden :** If we don’t want a method to be overridden, we declare it as [final](https://www.geeksforgeeks.org/final-keyword-java/). Please see [Using final with Inheritance](https://www.geeksforgeeks.org/using-final-with-inheritance-in-java/).

**Static methods can not be overridden(Method Overriding vs Method Hiding) :**When you define a static method with same signature as a static method in base class, it is known as [method hiding](https://www.geeksforgeeks.org/can-we-overload-or-override-static-methods-in-java/).

The following table summarizes what happens when you define a method with the same signature as a method in a super-class.



**Private methods can not be overridden :** [Private methods](https://www.geeksforgeeks.org/can-override-private-methods-java/) cannot be overridden as they are bonded during compile time. Therefore we can’t even override private methods in a subclass.(See [this](https://www.geeksforgeeks.org/can-override-private-methods-java/) for details).

**The overriding method must have same return type (or subtype) :** From Java 5.0 onwards it is possible to have different return type for a overriding method in child class, but child’s return type should be sub-type of parent’s return type. This phenomena is known as [**covariant return type**](https://www.geeksforgeeks.org/covariant-return-types-java/).

**Invoking overridden method from sub-class :** We can call parent class method in overriding method using [super keyword](https://www.geeksforgeeks.org/super-keyword/).

// A Java program to demonstrate that overridden

// method can be called from sub-class

// Base Class

class Parent {

    void show()

    {

        System.out.println("Parent's show()");

    }

}

// Inherited class

class Child extends Parent {

    // This method overrides show() of Parent

    @Override

    void show()

    {

        super.show();

**Overriding and constructor :** We can not override constructor as parent and child class can never have constructor with same name(Constructor name must always be same as Class name).

**Overriding and abstract method:** Abstract methods in an interface or abstract class are meant to be overridden in derived concrete classes otherwise a compile-time error will be thrown.

## **Why Method Overriding ?**

As stated earlier, overridden methods allow Java to support [run-time polymorphism](https://www.geeksforgeeks.org/dynamic-method-dispatch-runtime-polymorphism-java/). Polymorphism is essential to object-oriented programming for one reason: it allows a general class to specify methods that will be common to all of its derivatives while allowing subclasses to define the specific implementation of some or all of those methods. Overridden methods are another way that Java implements the “one interface, multiple methods” aspect of polymorphism.

# Covariant Return Types in Java

As the ear hit eardrums “overriding” we quickly get to know that it can be done either virtue of different datatypes or arguments passed to a function what a programmer learned initially while learning [polymorphism in java.](https://www.geeksforgeeks.org/polymorphism-in-java/)  Before JDK 5.0, it was not possible to [override](https://www.geeksforgeeks.org/overriding-in-java/) a method by changing the return type. When we override a parent class method, the name, argument types, and return type of the overriding method in child class has to be exactly the same as that of the parent class method. The overriding method was said to be **invariant** with respect to return type.

Java version 5.0 onwards it is possible to have different return types for an overriding method in the child class, but the child’s return type should be a subtype of the parent’s return type. The overriding method becomes **variant** with respect to return type.

The co-variant return type is based on the [Liskov substitution principle](https://en.wikipedia.org/wiki/Liskov_substitution_principle" \t "_blank).

Now geeks you must be wondering about why to use for which we will be listing down the advantages as follows:

<https://www.geeksforgeeks.org/overriding-in-java/>

<https://www.geeksforgeeks.org/covariant-return-types-java/>

interview questions

<https://javaconceptoftheday.com/java-interview-questions-on-method-overriding/>