What is Method Overriding in Java ? Example Tutorial

Method overriding in Java

Method overriding in Java is a concept based on polymorphism OOP concept allows the programmer to create two methods with the same name and method signature on the interface and its various implementation and the actual method is called at runtime depending upon the type of object at runtime. Method overriding allows you to write flexible and extensible code in Java because you can introduce new functionality with minimal code change. **Method overriding** is different than the method overloading in Java which we have discussed in the last article.

In *method overloading*, Only the name of two overloaded methods are the same but method signature must be different while in the method overriding, the method signature must be the same. method overriding represent true polymorphic behavior, where the only name needs to be the same underlying method logic can be different.

In this Java tutorial, we will see **What is method overriding in Java**, the Rules to override a method in Java, and an example of *How to override a method in Java*. We won't discuss the difference between method overloading and overriding in Java, maybe some other post.

Rules of method overriding in Java

There are few rules which needs to be followed while `overriding` any method in Java, failure to follow these rules results in a compile-time error Java.

1. The first and most important rule regarding method overriding in Java is that you can only `override` a method in a subclass. You can not override the method in the same class.

2. A second important rule of method overriding in Java that name and signature of the method must be the same in Super class and Sub class or in the interface and its implementation.

3. The third rule to override a method in Java is that the overriding method can not reduce the accessibility of the overridden method in Java. For example, if the overridden method is public then the overriding method can not be protected, private or package-private;

But the opposite is true overriding method can increase the accessibility of the method in Java, i.e. if the overridden method is protected then The overriding method can be protected or public.

4. Another worth noting rule of method overriding in Java is that the overriding method can not throw checked Exception is higher in the hierarchy than the overridden method. This means if the overridden method throws `IOException` then the overriding method can not throw `java.lang.Exception` in its throws clause because of `java.lang.Exception` comes higher than `IOException` in Exception hierarchy.

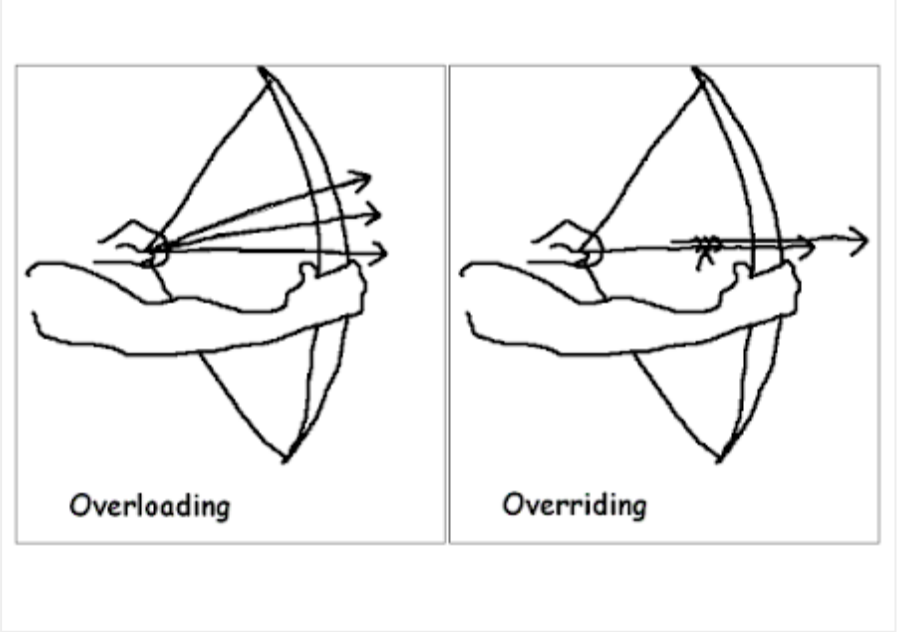
This rule doesn't apply to RuntimeException in Java, which is not even needed to be declared in a throws clause in Java.

5. You can not override private, static and final the method in Java. private and static method are bonded during compile time using static binding in Java and doesn't resolve during runtime. the overriding the final method in Java is a compile-time error. Though private and static methods can be hidden if you declare another method with the same and signature in the subclass.

6. The overridden method is called using dynamic binding in Java at runtime based upon the type of Object. As shown in the following example of method overriding in Java.

7. If you are extending the abstract class or implementing the interface then you need to override all abstract methods unless your class is not abstract. an abstract method can only be used by using method overriding.

8. Always use `@Override` annotation while the overriding method in Java. Though this is not a rule it's one of the best Java coding practices to follow. From Java 6 you can also use `@Override` annotation on a method inherited from the interface as well. If you are not familiar with annotations in Java then you can also check out these free Java Programming courses to learn Java from scratch.



Method Overriding Example in Java

Now we know *what is method overriding in Java* and the *rules of method overriding*, It's time to see an example of how to override a method in Java. In this example, we have used the Runnable interface which has an abstract `run()` method. We have two class `Task` and `PeriodicTask` which implements the Runnable interface and override run method.