**SNS COLLEGE OF TECHNOLOGY**
**(AN AUTONOMOUS INSTITUTION)**
COIMBATORE – 35
**DEPARTMENT OF COMPUTER SIENCE AND ENGINEERING**

**UNIT 3**

# Method Overriding in Java

1. Understanding the problem without method overriding
2. Can we override the static method
3. Method overloading vs. method overriding

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

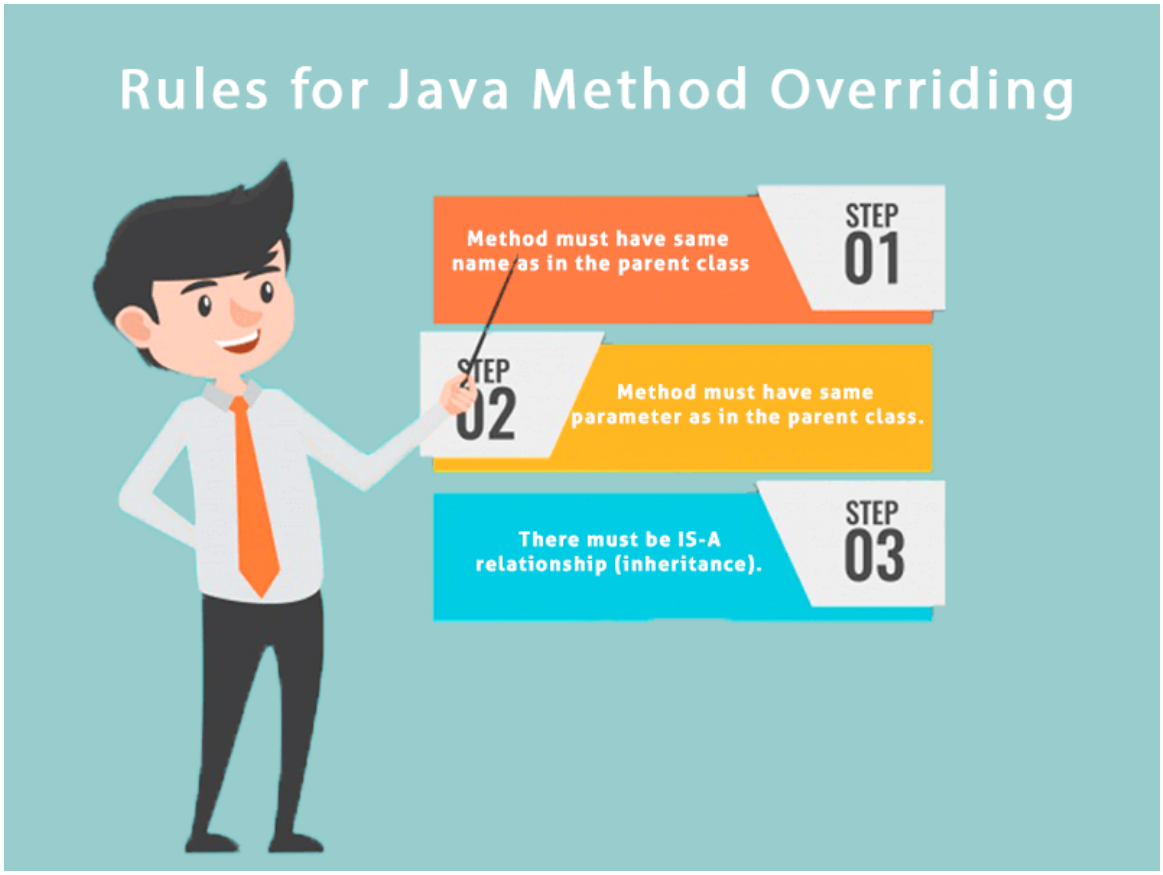
In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

## 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 Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).




## Understanding the problem without method overriding

Let's understand the problem that we may face in the program if we don't use method overriding.

```
1.  //Java Program to demonstrate why we need method overriding
2.  //Here, we are calling the method of parent class with child
3.  //class object.
4.  //Creating a parent class
5.  class Vehicle{
6.    void run(){System.out.println("Vehicle is running");}
7.  }
8.  //Creating a child class
9.  class Bike extends Vehicle{
10.   public static void main(String args[]){
11.   //creating an instance of child class
12.   Bike obj = new Bike();
13.   //calling the method with child class instance
14.   obj.run();
15.   }
```

```
16.         }
```

**Test it Now**

Output:

```
Vehicle is running
```

Problem is that I have to provide a specific implementation of run() method in subclass that is why we use method overriding.

---

## Example of method overriding

In this example, we have defined the run method in the subclass as defined in the parent class but it has some specific implementation. The name and parameter of the method are the same, and there is IS-A relationship between the classes, so there is method overriding.

```
1.          //Java Program to illustrate the use of Java Method Overriding
2.          //Creating a parent class.
3.          class Vehicle{
4.            //defining a method
5.            void run(){System.out.println("Vehicle is running");}
6.          }
7.          //Creating a child class
8.          class Bike2 extends Vehicle{
9.            //defining the same method as in the parent class
10.           void run(){System.out.println("Bike is running safely");}
11.
12.           public static void main(String args[]){
13.           Bike2 obj = new Bike2();//creating object
14.           obj.run();//calling method
15.           }
16.         }
```

**Test it Now**

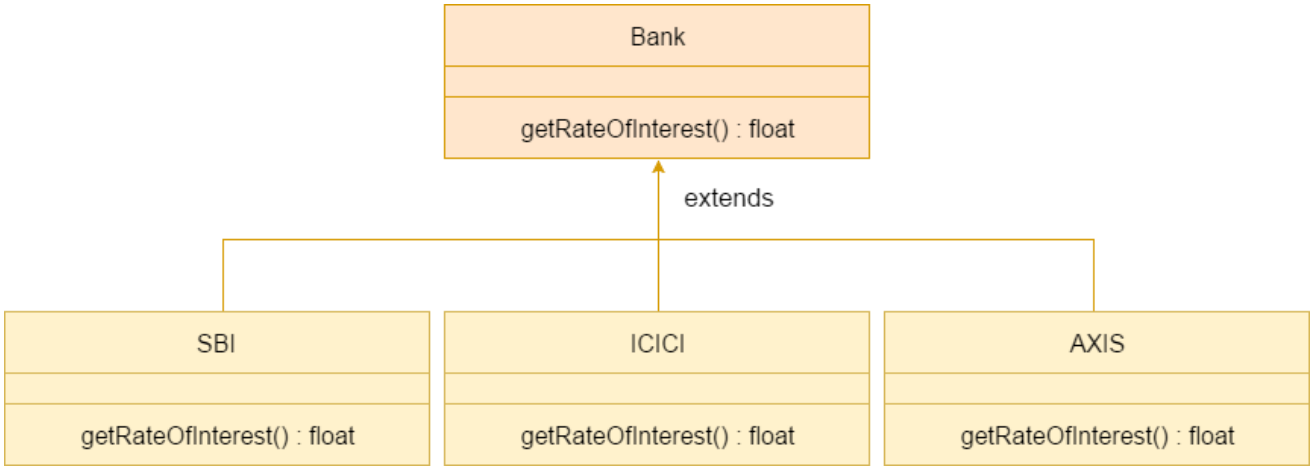
Output:

```
Bike is running safely
```

---

# A real example of Java Method Overriding

Consider a scenario where Bank is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks could provide 8%, 7%, and 9% rate of interest.




> Java method overriding is mostly used in Runtime Polymorphism which we will learn in next pages.

```
1.  //Java Program to demonstrate the real scenario of Java Method Overriding
2.  //where three classes are overriding the method of a parent class.
3.  //Creating a parent class.
4.  class Bank{
5.  int getRateOfInterest(){return 0;}
6.  }
7.  //Creating child classes.
8.  class SBI extends Bank{
9.  int getRateOfInterest(){return 8;}
10. }
11.
12. class ICICI extends Bank{
13. int getRateOfInterest(){return 7;}
14. }
15. class AXIS extends Bank{
16. int getRateOfInterest(){return 9;}
17. }
18. //Test class to create objects and call the methods
19. class Test2{
20. public static void main(String args[]){
```

```
21.         SBI s=new SBI();
22.         ICICI i=new ICICI();
23.         AXIS a=new AXIS();
24.         System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
25.         System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
26.         System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
27.         }
28.         }
```

**Test it Now**

```
Output:
SBI Rate of Interest: 8
ICICI Rate of Interest: 7
AXIS Rate of Interest: 9
```

## Can we override static method?

No, a static method cannot be overridden. It can be proved by runtime polymorphism, so we will learn it later.

---

## Why can we not override static method?

It is because the static method is bound with class whereas instance method is bound with an object. Static belongs to the class area, and an instance belongs to the heap area.

---

## Can we override java main method?

No, because the main is a static method.

# Difference between method overloading and method overriding in java

There are many differences between method overloading and method overriding in java. A list of differences between method overloading and method overriding are given below:

| No. | Method Overloading | Method Overriding |
|---|---|---|
| 1) | Method overloading is used *to increase the readability* of the program. | Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class. |
| 2) | Method overloading is performed *within class*. | Method overriding occurs *in two classes* that have IS-A (inheritance) relationship. |
| 3) | In case of method overloading, *parameter must be different*. | In case of method overriding, *parameter must be same*. |
| 4) | Method overloading is the example of *compile time polymorphism*. | Method overriding is the example of *run time polymorphism*. |
| 5) | In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter. | *Return type must be same or covariant* in method overriding. |