Difference between method Overloading and Overriding in java

BY CHAITANYA SINGH | FILED UNDER: OOPS CONCEPT

In this tutorial we will discuss the difference between overloading and overriding in Java. If you are new to these terms then refer the following posts:

1. Method overloading in java
2. Method overriding in java

Overloading vs Overriding in Java

1. Overloading happens at compile-time while Overriding happens at runtime: The binding of overloaded method call to its definition has happens at compile-time however binding of overridden method call to its definition happens at runtime.
2. Static methods can be overloaded which means a class can have more than one static method of same name. Static methods cannot be overridden, even if you declare a same static method in child class it has nothing to do with the same method of parent class.
3. The most basic difference is that overloading is being done in the same class while for overriding base and child classes are required. Overriding is all about giving a specific implementation to the inherited method of parent class.
4. Static binding is being used for overloaded methods and dynamic binding is being used for overridden/overriding methods.
5. Performance: Overloading gives better performance compared to overriding. The reason is that the binding of overridden methods is being done at runtime.
6. private and final methods can be overloaded but they cannot be overridden. It means a class can have more than one private/final methods of same name but a child class cannot override the private/final methods of their base class.
7. Return type of method does not matter in case of method overloading, it can be same or different. However in case of method overriding the overriding method can have more specific return type (refer this).
8. Argument list should be different while doing method overloading. Argument list should be same in method Overriding.

Overloading example

```
//A class for adding upto 5 numbers
class Sum
{
```

```
    int add(int n1, int n2)
    {
        return n1+n2;
    }
    int add(int n1, int n2, int n3)
    {
        return n1+n2+n3;
    }
    int add(int n1, int n2, int n3, int n4)
    {
        return n1+n2+n3+n4;
    }
    int add(int n1, int n2, int n3, int n4, int n5)
    {
        return n1+n2+n3+n4+n5;
    }
    public static void main(String args[])
    {
        Sum obj = new Sum();
        System.out.println("Sum of two numbers: "+obj.add(20, 21));
        System.out.println("Sum of three numbers: "+obj.add(20, 21, 22));
        System.out.println("Sum of four numbers: "+obj.add(20, 21, 22, 23));
        System.out.println("Sum of five numbers: "+obj.add(20, 21, 22, 23, 24));
    }
}
```

Output:

```
Sum of two numbers: 41
Sum of three numbers: 63
Sum of four numbers: 86
Sum of five numbers: 110
```

Here we have 4 versions of same method `add`. We are overloading the method `add()` here.

Overriding example

```
package beginnersbook.com;
class CarClass
{
    public int speedLimit()
    {
        return 100;
    }
}
class Ford extends CarClass
{
    public int speedLimit()
    {
        return 150;
    }
    public static void main(String args[])
    {
        CarClass obj = new Ford();
        int num= obj.speedLimit();
        System.out.println("Speed Limit is: "+num);
    }
```

```
}
```
Output:

```
Speed Limit is: 150
```
Here `speedLimit()` method of class `Ford` is overriding the `speedLimit()` method of class `CarClass`.