

Method Overloading in Java

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs.

So, we perform method overloading to figure out the program quickly.

Advantage of method overloading

Method overloading *increases the readability of the program*.

Different ways to overload the method

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

In java, Method Overloading is not possible by changing the return type of the method only.

1) Method Overloading: changing no. of arguments

In this example, we have created two methods, first add() method performs addition of two numbers and second add method performs addition of three numbers.

In this example, we are creating static methods so that we don't need to create instance for calling methods.

```

class Adder{
    static int add(int a,int b){return a+b;}
    static int add(int a,int b,int c){return a+b+c;}
}

class TestOverloading1{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11));
        System.out.println(Adder.add(11,11,11));
    }}

```

Output:

```

22
33

```

2) Method Overloading: changing data type of arguments

In this example, we have created two methods that differs in [data type](#). The first add method receives two integer arguments and second add method receives two double arguments.

```

class Adder{
    static int add(int a, int b){return a+b;}
    static double add(double a, double b){return a+b;}
}

class TestOverloading2{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11));
        System.out.println(Adder.add(12.3,12.6));
    }}

```

Output:

```

22
24.9

```

Q) Why Method Overloading is not possible by changing the return type of method only?

In java, method overloading is not possible by changing the return type of the method only because of ambiguity. Let's see how ambiguity may occur:

```
class Adder{
    static int add(int a,int b){return a+b;}
    static double add(int a,int b){return a+b;}
}
class TestOverloading3{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11)); //ambiguity
    }
}
```

Output:

Compile Time Error: method add(int,int) is already defined in class Adder

System.out.println(Adder.add(11,11)); //Here, how can java determine which sum() method should be called?

Note: Compile Time Error is better than Run Time Error. So, java compiler renders compiler time error if you declare the same method having same parameters.

Can we overload java main() method?

Yes, by method overloading. You can have any number of main methods in a class by method overloading. But **JVM** calls main() method which receives string array as arguments only. Let's see the simple example:

```
class TestOverloading4{
    public static void main(String[] args){System.out.println("main with String[]");}
    public static void main(String args){System.out.println("main with String");}
    public static void main(){System.out.println("main without args");}
}
```

Output:

```
main with String[]
```

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.

Usage of Java Method Overriding

- o Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- o 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.

```
//Java Program to demonstrate why we need method overriding
```

```
//Here, we are calling the method of parent class with child
```

```
//class object.
```

```
//Creating a parent class
```

```
class Vehicle{  
    void run(){System.out.println("Vehicle is running");}  
}
```

```
//Creating a child class
```

```
class Bike extends Vehicle{
    public static void main(String args[]){
        //creating an instance of child class
        Bike obj = new Bike();
        //calling the method with child class instance
        obj.run();
    }
}
```

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.

```
//Java Program to illustrate the use of Java Method Overriding
```

```
//Creating a parent class.
```

```
class Vehicle{
    //defining a method
    void run(){System.out.println("Vehicle is running");}
}
```

```
//Creating a child class
```

```
class Bike2 extends Vehicle{
    //defining the same method as in the parent class
    void run(){System.out.println("Bike is running safely");}
```

```
public static void main(String args[]){  
    Bike2 obj = new Bike2();//creating object  
    obj.run();//calling method  
}  
}
```

Output:

```
Bike is running safely
```

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