Top of Form

#### Method Overloading in Java

If a [class](https://www.javatpoint.com/object-and-class-in-java) 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](https://www.javatpoint.com/java-programs).

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

#### 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](https://www.javatpoint.com/static-keyword-in-java) so that we don't need to create instance for calling methods.

1. **class** Adder{
2. **static** **int** add(**int** a,**int** b){**return** a+b;}
3. **static** **int** add(**int** a,**int** b,**int** c){**return** a+b+c;}
4. }
5. **class** TestOverloading1{
6. **public** **static** **void** main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(11,11,11));
9. }}

**Output:**

22

33

#### 2) Method Overloading: changing data type of arguments

In this example, we have created two methods that differs in [data type](https://www.javatpoint.com/java-data-types). The first add method receives two integer arguments and second add method receives two double arguments.

1. **class** Adder{
2. **static** **int** add(**int** a, **int** b){**return** a+b;}
3. **static** **double** add(**double** a, **double** b){**return** a+b;}
4. }
5. **class** TestOverloading2{
6. **public** **static** **void** main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(12.3,12.6));
9. }}

**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:

1. **class** Adder{
2. **static** **int** add(**int** a,**int** b){**return** a+b;}
3. **static** **double** add(**int** a,**int** b){**return** a+b;}
4. }
5. **class** TestOverloading3{
6. **public** **static** **void** main(String[] args){
7. System.out.println(Adder.add(11,11));//ambiguity
8. }}

**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?

#### Method Overloading and Type Promotion

One type is promoted to another implicitly if no matching datatype is found. Let's understand the concept by the figure given below:



#### Example of Method Overloading with TypePromotion

1. **class** OverloadingCalculation1{
2. **void** sum(**int** a,**long** b){System.out.println(a+b);}
3. **void** sum(**int** a,**int** b,**int** c){System.out.println(a+b+c);}
5. **public** **static** **void** main(String args[]){
6. OverloadingCalculation1 obj=**new** OverloadingCalculation1();
7. obj.sum(20,20);//now second int literal will be promoted to long
8. obj.sum(20,20,20);
10. }
11. }

**Output:**

40

60

Bottom of Form

Top of Form

#### 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

* 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. }

**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");}
12. **public** **static** **void** main(String args[]){
13. Bike2 obj = **new** Bike2();//creating object
14. obj.run();//calling method
15. }
16. }

**Output:**

Bike is running safely

**Another Example:**



#### Can we override static method?

No, a static method cannot be overridden. It can be proved by runtime polymorphism.

#### 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.

Bottom of Form

Top of Form

#### ****Difference between method Overloading and Method Overriding in java****

Bottom of Form

Top of Form

Bottom of Form

Top of Form

**Video Lectures :**

[**https://www.youtube.com/watch?v=RaAp-f2Np1U**](https://www.youtube.com/watch?v=RaAp-f2Np1U)

[**https://youtu.be/9OXXbQBesHg**](https://youtu.be/9OXXbQBesHg)

**Reference Links:**

[**https://www.javatpoint.com/method-overloading-in-java**](https://www.javatpoint.com/method-overloading-in-java)

[**https://www.javatpoint.com/method-overriding-in-java**](https://www.javatpoint.com/method-overriding-in-java)

[**https://www.javatpoint.com/method-overloading-vs-method-overriding-in-java**](https://www.javatpoint.com/method-overloading-vs-method-overriding-in-java)

[**https://www.geeksforgeeks.org/overloading-in-java/**](https://www.geeksforgeeks.org/overloading-in-java/)

[**https://www.journaldev.com/16807/method-overloading-in-java**](https://www.journaldev.com/16807/method-overloading-in-java)

Bottom of Form