**Call by Value and Call by Reference in Java**

1. Methods can be accessed in two ways: by value and by reference. Calling a method by value means passing a value as a parameter.
2. Calling a method by reference means passing a reference (the variable’s location) as a parameter.
3. Primitive data types (int, float, double, boolean, char, and so on) are always passed as values in Java, while [non-primitive data types](https://www.prepbytes.com/blog/java/non-primitive-data-types-in-java/) (class, object, array, string, and interface) are always passed as references. This article will go over call by value in Java and call by reference in Java.

## **Call by Value in Java**

1. In Java, call by value refers to calling a method by passing the value in the parameter.
2. In Java, call by value passes a copy of the variable to the method, so all changes are reflected only in that method; thus, no changes are reflected in the main method.
3. This is the main difference between call by value in java and call by reference in java. When we pass a variable with primitive data types, it is considered a call by values in Java, so any changes to the variable will not be reflected in the caller’s scope.

### **Example of Call by Value in Java**

**class** Prepbytes{

**public** **static** **void** increment(**int** number){

number = number+1; // increment variable by 1

System.out.println("value in method: "+number);

}

**public** **static** **void** main(**String**[] args) {

**int** number=10;

System.out.println("value before method call : "+number);

increment(number);

System.out.println("value after method call: "+number);

}

}

**Output**

value before method call: 10

value in method: 11

value after method call: 10

We created a method increment with an int data type parameter number in the preceding example. In this method, we increase the variable number’s value by one. We created a variable number with an int data type and assigned it the value 10 in the main method. Now we called method increment and passed the variable number to it. We print the value of the number before calling the method, during the method increment, and after the method call is completed. The value of the variable only changed (value = 11) in the method increment before the method call, and the value after the method call is the same (value = 10). This is how call by value works in Java.

## **Call by Reference in Java**

1. Though Java is strictly call by value when we pass the reference of the object it creates a copy of the reference and then passes it as value to the method.
2. The copy reference also points to the same address so all the changes also reflect in the main method this is the main difference between call by value in java and call by reference in java.
3. Let’s see examples of how we can achieve call by reference in java.

### Example of Call by Reference in Java using Object

**class** Prepbytes{

**int** number=10;

// pass object as parameter

**public** **static** **void** increment(Prepbytes pb){

pb.number = pb.number+1; // increment variable by 1

System.out.println("value in method: "+pb.number);

}

**public** **static** **void** main(**String**[] args) {

Prepbytes pb=new Prepbytes(); // pb is an object of class Prepbytes

System.out.println("value before method call: "+pb.number);

increment(pb); // pass object of the class prepbytes

System.out.println("value after method call: "+pb.number);

}

}

**Output**

value before method call: 10

value in method: 11

value after method call: 11

In this example, we created an object pb of the class Prepbytes. We’ve now passed the class object to the method call. If we increase the value of the variable number, the main method will also change. So now if we print the value of the variable number it will be modified (value=11) after the increment method call. This is one way to achieve call by reference in java.