When a parameter is **passed by reference**, the caller and the callee **use the same variable** for the parameter. If the callee modifies the parameter variable, the effect is visible to the caller's variable.

When a parameter is **passed by value**, the caller and callee have **two independent variables** with the same value. If the callee modifies the parameter variable, the effect is not visible to the caller.

Q. Is Java pass by value of reference?

Ans: For simplicity ***Objects are passed by reference, and primitives are passed by value***

But correct ***Primitives and Object references are passed by value***

Java is always **pass-by-value**. Unfortunately, they decided to call the location of an object a "reference". When we pass the value of an object, we are passing the *reference* to it. This is confusing to beginners.

It goes like this:

public static void main( String[] args ) {

Dog aDog = new Dog("Max");

// we pass the object to foo

foo(aDog);

// aDog variable is still pointing to the "Max" dog when foo(...) returns

aDog.getName().equals("Max"); // true, java passes by value

aDog.getName().equals("Fifi"); // false

}

public static void foo(Dog d) {

d.getName().equals("Max"); // true

// change d inside of foo() to point to a new Dog instance "Fifi"

d = new Dog("Fifi");

d.getName().equals("Fifi"); // true

}

In this [example](http://ideone.com/oUEPbg) aDog.getName() will still return "Max". The value aDog within main is not changed in the function foo with the Dog "Fifi" as the object reference is passed by value. If it were passed by reference, then the aDog.getName() in main would return "Fifi" after the call to foo.

Likewise:

public static void main( String[] args ) {

Dog aDog = new Dog("Max");

foo(aDog);

// when foo(...) returns, the name of the dog has been changed to "Fifi"

aDog.getName().equals("Fifi"); // true

}

public static void foo(Dog d) {

d.getName().equals("Max"); // true

// this changes the name of d to be "Fifi"

d.setName("Fifi");

}

In the above example, FiFi is the dog's name after call to foo(aDog) because the object's name was set inside of foo(...). Any operations that foo performs on d are such that, for all practical purposes, they are performed on aDog itself (except when d is changed to point to a different Doginstance like d = new Dog("Boxer")).