# Java is Pass by Value and Not Pass by Reference

APRIL 2, 2018 BY PANKAJ — 67 COMMENTS

One of the biggest confusion in Java programming language is whether java is **Pass by Value** or **Pass by Reference**. I ask this question a lot in interviews and still see interviewee confused with it. So I thought to write a post about it to clarify all the confusions around it.

First of all we should understand what is meant by pass by value or pass by reference.

- **Pass by Value**: The method parameter values are copied to another variable and then the copied object is passed, that's why it's called pass by value.
- **Pass by Reference**: An alias or reference to the actual parameter is passed to the method, that's why it's called pass by reference.

Java is always Pass by Value and not pass by reference, we can prove it with a simple example.

Let's say we have a class `Balloon` like below.

And we have a simple program with a generic method to swap two objects, the class looks like below.

When we execute above program, we get following output.

If you look at the first two lines of the output, it's clear that swap method didn't worked. This is because Java is pass by value, this swap() method test can be used with any programming language to check whether it's pass by value or pass by reference.

Let's analyze the program execution step by step.

When we use **new** operator to create an instance of a class, the instance is created and the variable contains the reference location of the memory where object is saved. For our example, let's assume that "red" is pointing to 50 and "blue" is pointing to 100 and these are the memory location of both Balloon objects.

Now when we are calling swap() method, two new variables o1 and o2 are created pointing to 50 and 100 respectively.

So below code snippet explains what happened in the swap() method execution.

Notice that we are changing values of o1 and o2 but they are copies of “red” and “blue” reference locations, so actually there is no change in the values of “red” and “blue” and hence the output.

If you have understood this far, you can easily understand the cause of confusion. Since the variables are just the reference to the objects, we get confused that we are passing the reference so java is pass by reference. However we are passing a copy of the reference and hence it’s pass by value. I hope it clear all the doubts now.

Now let’s analyze foo() method execution.

The first line is the important one, when we call a method the method is called on the Object at the reference location. At this point, balloon is pointing to 100 and hence it’s color is changed to Red.

In the next line, ballon reference is changed to 200 and any further methods executed are happening on the object at memory location 200 and not having any effect on the object at memory location 100. This explains the third line of our program output printing **blue color=Red**.

I hope above explanation clear all the doubts, just remember that variables are references or pointers and it’s copy is passed to the methods, so **java is always pass by value**. It would be more clear when you will learn about Heap and Stack memory and where different objects and references are stored, for a detailed explanation with reference to a program, read Java Heap vs Stack.

**Update**: I am getting a lot of comments and it seems that still there is a lot of confusion, so I have made a video tutorial to explain this in detail.

« PREVIOUS
Hibernate Interview Questions and Answers

NEXT »
40 Common Job Interview Questions and Answer Tips

**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: INTERVIEW QUESTIONS, JAVA

## Comments

**Parth says**
JUNE 27, 2018 AT 12:38 AM
Thanks for the video tutorial. It really cleared the concept. Thanks a lot!!
Reply

**ramason says**
MAY 17, 2018 AT 7:06 PM
Hello.
I'm Java Developer in Korea □
Can I upload the post to my blog translating this?
The another code 'll be added of the post!
If you'll allow, please answer me.
Thank you.
Reply

**Pankaj says**
MAY 18, 2018 AT 1:52 AM
Rather than commenting here, it would be good if you contact me directly. Usually we don't allow republishing of our articles elsewhere.
Reply

**Progstacks says**
MAY 6, 2018 AT 7:25 AM
I would say it is both pass-by-reference and pass-by-value and somewhat confusing. I did a litter experiment and I posted in my blog.
http://programmingstacks.com/blog/java-is-both-pass-by-reference-and-pass-by-value-however/
Reply

**karatedog says**
MAY 3, 2018 AT 3:29 PM

You all fighting terminology here. Look at Assembly, where everything is a value, period. Any other fancy name (like “reference” or “pointer”) is the way you use this value.
Semantically, the sentence: “A copy of a reference is value” is false, If you changed “reference” to “dog”, and “value” to “cat” it is probably easier to see.
A copy of a pointer is, well… could be a pointer. Or my mother’s birthday, if it happens to be exactly 1941.
So Java is pass by value, because someone long ago stated, that a copy of a reference is a value. That’s all.
Anyway, check Stackoverflow for Is Java “pass-by-reference” or “pass-by-value”?, the question is there since 2012.
Reply

**Lokesh says**
APRIL 26, 2018 AT 11:00 AM
Hello Pankaj,
If you think the program is passing balloon objects as pass by value. Consider this program, it is swapping the values now.

```
package com.baloon;
public class Test {
public static void main(String[] args) {
Balloon red = new Balloon("Red"); //memory reference 50
Balloon blue = new Balloon("Blue"); //memory reference 100
swap(red, blue);
System.out.println("red color="+red.getColor());
System.out.println("blue color="+blue.getColor());
}
//Generic swap method
public static void swap(Balloon o1, Balloon o2){
/* Object temp = o1;
o1=o2;
o2=temp; */
o1.setColor("Blue");
o2.setColor("Red");
}
}
```

Output :

```
red color=Blue
blue color=Red
```

Reply

**David says**

MARCH 15, 2018 AT 5:49 PM

Do you really ask this question in job interviews? These are the type of questions that make common interview practices as useful as a coin flip.

Java passes variables by value (where object non primitive variables are actually references to an object). If someone can demonstrate they know how this works, rather than trying to answer a question that is known to be interpreted different ways. Have you ever known someone where answering this was the proving factor on their success? I guarantee there are people that have read that java passes object references by value, but would not be able to tell you the output of your tests.

Please stop asking questions like this. Everyone, please stop using interview questions like this. Asking "is java pass by value or pass by reference" when you know people are going to get confused looks on their faces only serves the purpose of feeding ones ego an sense of supiority. Instead try seeing if the candidate can actually write correct functioning code, that they can grok what code examples may do. More importantly, put an effort in the interview to determine if the candidate is bright, driven and a strong personality fit for your team. This will improve your rate of hiring strong team members. Itwill reduce the number of false negatives which come from candidates incorrectly answering pointed trivia questions that help the interviewer feel good about themselves. It will also reduce the number of missed job opportunities for candidates who were asked poor questions that prove nothing about potential for success.

The interview is. It about you (the I terviewer) it is about the candidate. Don't leave an interview feeling smart because you were able to trick a candidate. Instead, leave the interview feeling good because you helped the candidate really show what the have to offer.

Sorry to rant – I get frustrated with people perpetuating poor interview practices for the '80s. I know there are other good articles here that help provide deeper understanding of java. However, this article just feels disingenuous and a little smug to me.

Reply

**Pankaj says**

MARCH 16, 2018 AT 1:32 AM

Hi David,

If you have faced or taken a lot of interviews, you must know that it's not only about giving correct answers. Nobody expects to get 100% correct answers and any good interviewer don't judge if an answer is wrong. Some questions are confusing and they are still asked to see how interviewee responds to it, how he reacts when follow-up or counter questions are asked.

Technical knowledge can get you only half way, confidence, way of talking, problem solving approach, soft skills play a vital role in getting a job.

The article is written to help you understand the topic clearly, there is nothing "*disingenuous and a little smug*" part in it.

Reply

**Himanshu says**

MARCH 14, 2018 AT 12:09 PM

Appreciate your vedio

Reply

**William L. DeRieux IV says**

NOVEMBER 21, 2017 AT 2:28 PM

I believe that you don't fully understand pass by value and pass by reference.
primitive parameters are passed by value.
reference parameters are passed by reference.
pass by value means a copy of the primitive is sent to the method – any changes to the parameter will not change the original value.
pass by reference means that the address of the object is sent to the method – any changes to the parameter's instance members will result in that change being made in the original value..
What you are talking about in this article has nothing to do with pass by reference, instead you are talking about changing the address that the parameter points to, to some new location in memory — the address is constant but the data that the pointer refers to is not.

Reply

**miticosul says**

DECEMBER 1, 2017 AT 2:55 AM

+1! □ well done! my exactly the same thoughts regarding this article!

Reply

**Pankaj says**

DECEMBER 1, 2017 AT 9:43 AM

Sorry to say this but your understanding about "Pass by Reference" and "Pass by Value" is not correct. Below quote properly describes it:

```
Passing by reference means the called functions' parameter will be the same as the callers' passed argument (not the value, but the identity - the variable itself). Pass by value means the called functions' parameter will be a copy of the callers' passed argument. The value will be the same, but the identity - the variable - is different.
```

Also refer this StackOverflow Post: https://stackoverflow.com/questions/373419/whats-the-difference-between-passing-by-reference-vs-passing-by-value

Reply

**Anilal says**

DECEMBER 28, 2017 AT 4:33 AM

It is still confusing.
As you said in the beginning of the post,

"Pass by Reference: An alias or reference to the actual parameter is passed to the method, that's why it's called pass by reference"
And in the code example, exactly same happens when the method "foo" is called. Since the function argument is a reference to the original object, we can modify the content of the original object in method "foo". ( balloon.setColor("Red");)
So my understanding is
1. For Primitive types, parameters are passed by value.
2. For Object types, object reference is passed by value.
Reply

**szp says**
APRIL 18, 2018 AT 9:07 PM
As far as I know, the term "pass by value/reference" is comes form c/c++.
But java actual do not features related the c/c++ pointer concept.
So I suppose the point why people are debating here is that Mr Pankaj mixed the c/c++feature"pass by value/reference" with java in the title of the post.
But, even through the mixture is confusing for me, I still learned some useful details of the implementation of java programming language. Thanks.

**Govind Ostwal says**
MAY 4, 2018 AT 4:12 AM
+1 to William's answer.
Your test is not about pass by value / reference. Please replace it will valid test to avoid further confusion.
Java is pass by value for primitives and pass by reference for objects.
Please correct your misleading tutorial.
Reply

**Balwinder Singh says**
SEPTEMBER 4, 2017 AT 10:16 AM
Awesome explanation, thanks
Reply

**Pooya says**
AUGUST 28, 2017 AT 11:57 PM
it is the best explenation about pass by reference and pass by value.
everything is too clear right now
Reply

**Christopher says**
AUGUST 2, 2017 AT 10:12 AM
Pankaj:
I'm not a new programmer, but fairly new to Java. Once again your calm and helpful manner has come through for me. Thanks for the clarity.
Reply

**Pankaj says**
AUGUST 2, 2017 AT 10:23 AM
Thanks for the kind words, appreciate it.
Reply

**Anurag Singh says**
JULY 26, 2017 AT 8:05 AM
this post clear my all problem.
thank you
keep posting
Reply

**Rahul says**
MAY 18, 2017 AT 12:04 AM
Excellent explanation.
Reply

**Dharmendra says**
FEBRUARY 8, 2017 AT 3:54 AM
Hi Pankaj,

Thanks for such a wonderful article.
I have one question based on your article.

```
import java.util.ArrayList;
import java.util.List;
public class Test {
public static void main(String[] args) {
List a = new ArrayList();
a.add("Apple");
System.out.println("LIST SIZE"+a.size());
addInList(a, "Banana");
addInList(a, "COFFEE");
System.out.println("LIST SIZE"+a.size());
}
public static void addInList(List list, Object ojb){
list.add(ojb);
System.out.println("LIST SIZE::"+list.size());
}
}
```

output:
LIST SIZE1
LIST SIZE::2
LIST SIZE::3
LIST SIZE3
If java support pass by value then why list size is modified
In last print statement list size should be 1,
Could you clarify to me
Thanks
Dharmendra
Reply

**Victor says**
JUNE 30, 2017 AT 1:43 AM
List is an object. When you pass a object to a method, you are passing the value of the memory address of the object to the method. Thus you can modify the attributes of the object. However, you cannot reassign another object to this memory address.
Reply

**Disha says**
AUGUST 20, 2017 AT 1:05 AM
Hi Victor,
In Java, the new keyword returns a reference to a newly-constructed object. Therefore, in

String s = new String("foo");
the value of variable s is a reference to a String object
In the same way
List a = new ArrayList();
a should be reference to ArrayList Object
Isn't it?
Reply

**Bhagwati Lal says**
JANUARY 15, 2017 AT 5:55 AM
Very Urgent please help me.
Can you explain me how to draw memory diagram for following java code.

```
public class ArrayOfReferences {
public static void main(String[] args) {
String[] names; /* How many objects do we have? */
int numberOfNames = 3;
Scanner scanner = new Scanner(System.in);
/* Reading names */
names = new String[numberOfNames]; /* How many objects do we have? */
for (int idx = 0; idx < names.length; idx++) {
System.out.println("Enter the name of a friend: ");
names[idx] = scanner.next();
}
/* Printing names */
System.out.println("Your friends are: ");
for (int idx = 0; idx < names.length; idx++) {
System.out.println(names[idx]);
}
scanner.close();
}
}
```

Reply

**Arunkumar says**
JANUARY 8, 2017 AT 7:28 PM
Thanks a lot pankaj. That was a very good explanation.
Reply

**Jagadish says**
DECEMBER 29, 2016 AT 10:03 PM
Thanks Pankaj for explaining it more clearly in video. Now doubt is cleared.
Reply

**Vyacheslav says**
SEPTEMBER 21, 2016 AT 12:02 AM
Hello. Thanks for a good explanation.
Yet, I suppose, there is an issue what to mean passing by value or by reference. Look there – http://www.careerride.com/C++-pass-by-value-and-pass-by-reference.aspx. I like very much the idea, that the main difference is:
By val — The procedure code does not have any access to the underlying element in the calling code;
By ref — The procedure code gives a direct reference to the programming element in the calling code.
So, to my mind it is not clear that swap method is a correct way to obviously say if a language passes vars by value or by ref.
Reply

**Vyacheslav says**
SEPTEMBER 21, 2016 AT 12:22 AM
Well, also a good article: http://javadude.com/articles/passbyvalue.htm. Java doesn't support variable references, so, we can not speak at all about passing by reference in Java.
Reply

**Renat says**
SEPTEMBER 13, 2016 AT 3:55 AM
Man, you're so confused it's not funny. Stop writing stuff on the internet, people here believe all they see.
Consider this – it's an equivalent C program that passes a struct by reference and the behaviour is exactly the same as in Java:
#include
typedef struct s {
const char *colour;
} balloon;
void swap (balloon *b1, balloon *b2) {
balloon *tmp = b1;
b1 = b2;

```
b2 = tmp;
}
int main (int argc, char **argv) {
balloon b1, b2;
b1.colour = "red";
b2.colour = "blue";
swap (&b1, &b2);
printf ("red: %s, blue: %s\n", b1.colour, b2.colour);
return 0;
}
```

Reply

**Pankaj says**

SEPTEMBER 13, 2016 AT 6:03 AM

First of all here we are talking about Java. Even after reading and hopefully understanding above program, you don't understood then I feel for you. Read again, may be somewhere else… nothing else I can do for you.

Reply

**Vlad says**

SEPTEMBER 14, 2016 AT 7:14 AM

Your code does not pass by reference, it passes pointer by value. Moreover, there is no such thing as "reference" in C, it exists only in C++:

```
#include
typedef struct s {
const char *colour;
} balloon;
void swap (balloon &b1_r, balloon &b2_r) {
balloon tmp = b1_r;
b1_r = b2_r;
b2_r = tmp;
}
int main (int argc, char **argv) {
balloon b1, b2;
b1.colour = "red";
b2.colour = "blue";
swap (b1, b2);
printf ("red: %s, blue: %s\n", b1.colour, b2.colour);
return 0;
}
```

Reply

**Praveen says**
JULY 19, 2016 AT 6:01 AM
Damn it., That's one of the simplest way to explain it.
Thanks a ton Guru □
Reply

**Gadrov says**
JULY 18, 2016 AT 9:06 PM
I understand that Java is always pass by value, what I don't understand is what exactly is the difference between pass an object by reference and pass an object reference by value.
If you pass the reference value, isn't that essentially the same thing? What would change in the swap and balloon examples?
Currently I only understand the fact that calling the class's constructor allocates a different memory set for the variable, otherwise I can't see the difference. That's why a lot of people say that Java passes objects by reference, because the behavior, if not identical, is extremely similar.
Reply

**Mandeep Sharma says**
JULY 13, 2016 AT 12:28 AM
Hi,
Can you please declare an array of integer and initialize it with some values. then pass this array to some method (return type void) and add 1 to all of the elements in array. and come back in main and try to print array element.. this will print the incremented values. Can you do clear this thing why its happens.
Reply

**Albert says**
JUNE 23, 2016 AT 3:08 AM
Dear Mr. Pankaj,
I have a confusion with this method:
private static void foo(Balloon balloon) { //baloon=100 // line1
balloon.setColor("Red"); //baloon=100
balloon = new Balloon("Green"); //baloon=200
balloon.setColor("Blue"); //baloon = 200

}

In Line1,Could you please tell me why balloon points to “Blue” Object instead of “Red” Object ?

Reply

**Lakshman says**

JUNE 27, 2016 AT 2:09 AM

At line 1, object refers to Blue color Balloon object. From caller we passed Blue color Balloon object.

Reply

**Narendra Koli says**

JUNE 9, 2016 AT 12:13 PM

Thank you so much for creating video.

Your tutorials are really helpful.

Reply

**Pankaj says**

JUNE 9, 2016 AT 12:34 PM

You are welcome Narendra.

Reply

**qiangyu says**

MARCH 8, 2016 AT 7:23 PM

good post ,Thanks

Reply

**Esakki says**

FEBRUARY 4, 2016 AT 3:48 AM

Really good explanation (especially on the video content)

Reply

**Olga Guskova says**

JANUARY 20, 2016 AT 5:17 AM

Very useful explanation and tutorial in whole.

Thanks.

Reply

**bin says**

SEPTEMBER 5, 2015 AT 10:16 PM

good one. Aprreciated

Reply

**Brihati Jain says**

JUNE 17, 2015 AT 2:49 AM

It was a great post. In video you are using stack memory for static methods memory allocation.Can you please explain why so?.Static methods exist in pergen space of heap

Reply

**Rahul Jain says**

MAY 22, 2015 AT 2:13 AM

Very nicely explained

Reply

**Stefan Banu says**

APRIL 17, 2015 AT 12:32 AM

Hi Pankaj,

from the video i understood exactly, so the video solution is much much better,

thanks for all your work, your site is always a “reference” for my java development understanding.

thanks and keep doing your thing.

Stefan from Romania.

Reply

**Igorovics says**

JANUARY 22, 2015 AT 5:38 AM

Great explanation, thank you.

Reply

**konrad says**
DECEMBER 8, 2014 AT 6:37 AM
I think that the post topic is somehow misleading/confusing. Of course Java is pass by value for primitives and pass by reference for objects – you even proved that drawing arrows (references!, not values).
On the other hand those references (i.e. memory location addresses) are passed by value – that is how I think it should be understood.
Reply

**Bejond says**
MARCH 30, 2016 AT 5:14 PM
Totally agree with your opinion.
I believe "Pass by Value: The method parameter values are copied to another variable and then the copied object is passed, that's why it's called pass by value." is misunderstanding. Because right description should be "Pass by Value: The method parameter values are copied to another variable and then the copied reference is passed, that's why it's called pass by copied reference."
Bejond
Reply

**Pankaj Tandakar says**
AUGUST 22, 2014 AT 12:48 AM
But in case of pass by value ,since a copy is passed the original one doesnt get changed which occurs in case of primitive data types ,but same we do in case of Objects ,its original value gets changed
Reply

**Dan says**
AUGUST 21, 2014 AT 10:30 AM
These def