

[JAVA TUTORIAL](#)[#INDEX POSTS](#)[#INTERVIEW QUESTIONS](#)[RESOURCES](#)[HIRE ME](#)[DOWNLOAD ANDROID APP](#)[CONTRIBUTE](#)

Subscribe to Download Java Design Patterns eBook

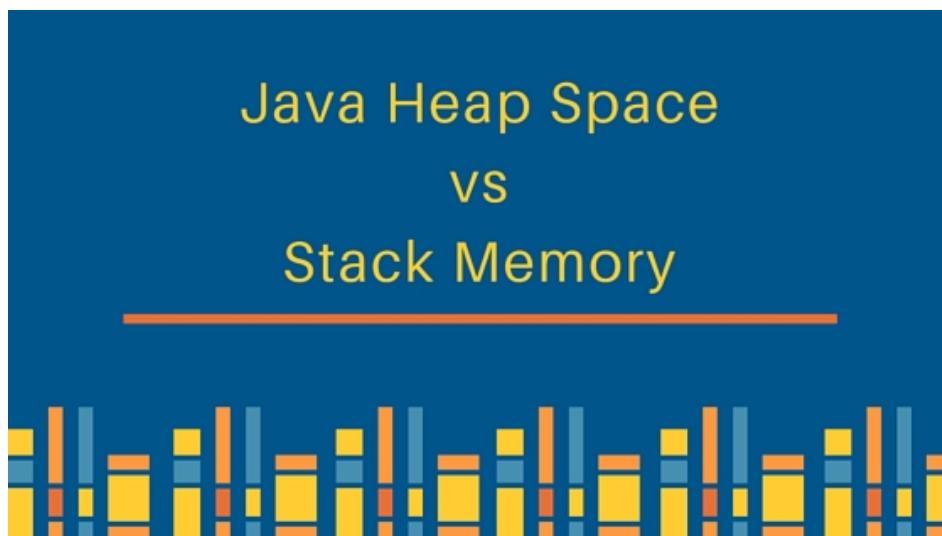
DOWNLOAD NOW

[HOME](#) » [INTERVIEW QUESTIONS](#) » [JAVA HEAP SPACE VS STACK – MEMORY ALLOCATION IN JAVA](#)

# Java Heap Space vs Stack – Memory Allocation in Java

APRIL 2, 2018 BY [PANKAJ](#) — [135 COMMENTS](#)

Sometime back I wrote a couple of posts about [Java Garbage Collection](#) and [Java is Pass by Value](#). After that I got a lot of emails to explain about **Java Heap Space**, **Java Stack Memory**, **Memory Allocation in Java** and what are the differences between them.



You will see a lot of reference to Heap and Stack memory in Java, Java EE books and tutorials but hardly complete explanation of what is heap and stack memory in terms of a program.

#### Table of Contents [\[hide\]](#)

- 1 Java Heap Space
  - 1.1 Java Stack Memory
  - 1.2 Heap and Stack Memory in Java Program
- 2 Difference between Java Heap Space and Stack Memory

## Java Heap Space

Java Heap space is used by java runtime to allocate memory to Objects and JRE classes. Whenever we create any object, it's always created in the Heap space.

Garbage Collection runs on the heap memory to free the memory used by objects that doesn't have any reference. Any object created in the heap space has global access and can be referenced from anywhere of the application.

## Java Stack Memory

Java Stack memory is used for execution of a thread. They contain method specific values that are short-lived and references to other objects in the heap that are getting referred from the method.

Stack memory is always referenced in LIFO (Last-In-First-Out) order. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method.

As soon as method ends, the block becomes unused and become available for next method. Stack memory size is very less compared to Heap memory.

## Heap and Stack Memory in Java Program

Let's understand the Heap and Stack memory usage with a simple program.

```
package com.journaldev.test;

public class Memory {

    public static void main(String[] args) { // Line 1
        int i=1; // Line 2
        Object obj = new Object(); // Line 3
        Memory mem = new Memory(); // Line 4
        mem.foo(obj); // Line 5
    }
}
```

```

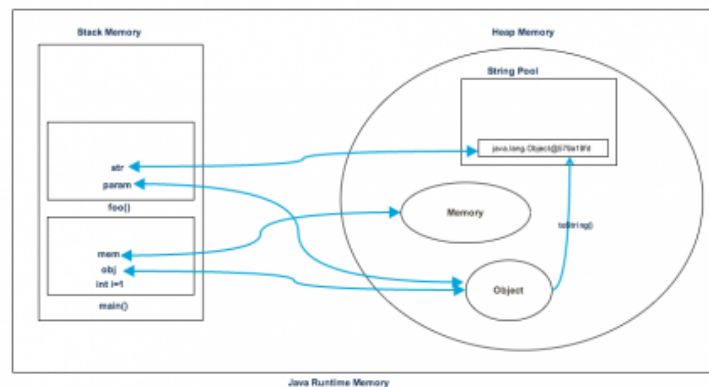
    } // Line 9

    private void foo(Object param) { // Line 6
        String str = param.toString(); //// Line 7
        System.out.println(str);
    } // Line 8

}

```

Below image shows the Stack and Heap memory with reference to above program and how they are being used to store primitive, Objects and reference variables.



## APIs for Java Developers

DOC, XLS, PDF, PPT, Image and more Try a free evaluation version [aspose.com](https://www.aspose.com)



Let's go through the steps of execution of the program.

- As soon as we run the program, it loads all the Runtime classes into the Heap space. When `main()` method is found at line 1, Java Runtime creates stack memory to be used by `main()` method thread.
- We are creating primitive local variable at line 2, so it's created and stored in the stack memory of `main()` method.
- Since we are creating an Object in line 3, it's created in Heap memory and stack memory contains the reference for it. Similar process occurs when we create `Memory` object in line 4.
- Now when we call `foo()` method in line 5, a block in the top of the stack is created to be used by `foo()` method. Since Java is pass by value, a new reference to `Object` is created in the `foo()` stack block in line 6.
- A string is created in line 7, it goes in the **String Pool** in the heap space and a reference is created in the `foo()` stack space for it.
- `foo()` method is terminated in line 8, at this time memory block allocated for `foo()` in stack becomes free.
- In line 9, `main()` method terminates and the stack memory created for `main()` method is destroyed. Also the program ends at this line, hence Java Runtime frees all the memory and end the execution of the

program.

## Difference between Java Heap Space and Stack Memory

Based on the above explanations, we can easily conclude following differences between Heap and Stack memory.

1. Heap memory is used by all the parts of the application whereas stack memory is used only by one thread of execution.
2. Whenever an object is created, it's always stored in the Heap space and stack memory contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space.
3. Objects stored in the heap are globally accessible whereas stack memory can't be accessed by other threads.
4. Memory management in stack is done in LIFO manner whereas it's more complex in Heap memory because it's used globally. Heap memory is divided into Young-Generation, Old-Generation etc, more details at [Java Garbage Collection](#).
5. Stack memory is short-lived whereas heap memory lives from the start till the end of application execution.
6. We can use **-Xms** and **-Xmx** JVM option to define the startup size and maximum size of heap memory. We can use **-Xss** to define the stack memory size.
7. When stack memory is full, Java runtime throws `java.lang.StackOverflowError` whereas if heap memory is full, it throws `java.lang.OutOfMemoryError: Java Heap Space` error.
8. Stack memory size is very less when compared to Heap memory. Because of simplicity in memory allocation (LIFO), stack memory is very fast when compared to heap memory.

That's all for **Java Heap Space vs Stack Memory** in terms of java application, I hope it will clear your doubts regarding memory allocation when any java program is executed.

I have made a video tutorial for java heap space and stack memory, you should watch it to clarify any doubts.

Subscribe to my YouTube Channel

YouTube

6K

#### References:

[https://en.wikipedia.org/wiki/Java\\_memory\\_model](https://en.wikipedia.org/wiki/Java_memory_model)

<https://blogs.oracle.com/jonthecollector/presenting-the-permanent-generation>

#### « PREVIOUS

40 Common Job Interview Questions and Answer Tips

#### NEXT »

5 Best Core Java Books for Beginners

#### 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

**Maryam says**

JULY 20, 2018 AT 1:34 AM

That was really good and clear. Thanks!

[Reply](#)

**Mouaad says**

JULY 3, 2018 AT 1:10 PM

Really interesting and very good and clear explanation. Thanks a lot.

[Reply](#)

**Tom says**

JUNE 28, 2018 AT 8:17 AM

Hi! I tried a little example, however, I get zero, and not  $9 \times 4$ . Could you explain why?

```
Integer v = 4;
```

```
Integer b = 9;
```

```
Integer x = 0;
```

```
integerMet (v, b, x);
```

```
System.out.println(" x after integerMet:" + x);
```

```
static public void met (int a, int b, int c){
```

```
    c = a*b;
```

```
}
```

[Reply](#)

**Bangerboi says**

JULY 7, 2018 AT 3:04 AM

You declare c as an int in met()'s signature, resulting in x being casted to a primitive int.

The value of the integer x is being given to c, but not the reference.

[Reply](#)

**Abhi1111 says**

JULY 14, 2018 AT 4:26 AM

Here u passing the copy of variable x to c, it will remain same in parent method.  
Since integer X reference is not changed , whatever your assigning you are assigning to int c.

[Reply](#)**Atul says**

JUNE 25, 2018 AT 6:32 AM

really, good Artical

[Reply](#)**Neetu Kumari says**

JUNE 21, 2018 AT 6:28 AM

Easy going approach of explanation...fantastic really!! cleared the doubt..

[Reply](#)**Saishav says**

JUNE 16, 2018 AT 6:15 AM

Concise and well explained. Good job!

[Reply](#)**Sathvik says**

JUNE 13, 2018 AT 2:56 AM

Good Article .Thank you ):

[Reply](#)**Samrat says**

MAY 21, 2018 AT 2:19 AM

Wonderful article with detailed way of explanation. Easy to understand.

[Reply](#)

**nileshD says**

APRIL 25, 2018 AT 11:57 PM

thank you very much for such wonderful article.

[Reply](#)**yubo says**

APRIL 22, 2018 AT 8:48 AM

Hi PANKAJ,

thanks for your article! One question, in the fourth bullet point walking through the example:

"Now when we call foo() method in line 5, a block in the top of the stack is created to be used by foo() method. Since Java is pass by value, a new reference to Object is created in the foo() stack block in line 6."

For non-primitive data type, java pass it by reference instead of pass by value right?

Thanks

[Reply](#)**Prateek says**

MAY 25, 2018 AT 1:55 PM

Even for non-primitive data types, value of (reference)variable (i.e the address/pointer to object on Java heap) is copied between method calls .

[Reply](#)**bulent says**

APRIL 19, 2018 AT 5:08 AM

thank you

[Reply](#)**user says**

MARCH 23, 2018 AT 12:29 AM

Thank you!

[Reply](#)



**Bhargava says**

MARCH 14, 2018 AT 10:12 AM

Nicely explained

[Reply](#)**LW says**

MARCH 6, 2018 AT 10:36 AM

The video is excellent, thank you!

[Reply](#)**Satheesh R says**

DECEMBER 5, 2017 AT 1:32 AM

Looking like a charm.

[Reply](#)**mdr says**

DECEMBER 2, 2017 AT 5:03 AM

Sir propably i understand you wrong but you said:

"Any object created in the heap space has global access and can be referenced from anywhere of the application"

but when i run this code:

```
{  
Cat cat1 = new Cat();  
cat1.meow();  
}  
cat1.meow();
```

I get out of scope error

[Reply](#)**Deepak Dhaka says**

JANUARY 13, 2018 AT 11:11 PM

HI Mdr,

In this case, the error out of scope came because the scope of cat1 is now lost { the property of stack}

and the object of cat is created in Heap, is now eligible for Garbage Collection,

By Saying "Any object created in the heap space has global access and can be referenced from anywhere of the application" means anything that is created in Heap is not per Thread bases.

[Reply](#)

**Tal says**

NOVEMBER 28, 2017 AT 10:30 AM

Hi. Thank you for the great explanation.

I have a question please:

Given the following class:

```
Class MyClass{  
    public int count;  
    public int getCount();  
    public void incCountBy1();  
}
```

Suppose we have only one thread that uses increment method and many threads that uses get method.

How much time can it take from setting the value to first read?

Can it take even 1 sec??

Thanks a lot!

[Reply](#)

**Pankaj says**

NOVEMBER 28, 2017 AT 11:27 PM

That depends on what is going on inside the method, also is it synchronized to make sure no dirty-read operation happens?

[Reply](#)

**Vijay Jetti says**

NOVEMBER 2, 2017 AT 7:09 AM

Simply Superb,

[Reply](#)

**Mohamad Ali Al-aaraji says**

OCTOBER 29, 2017 AT 7:04 AM

Thanks a lot,you just saved me from getting lost \* \_ \*

[Reply](#)**Alex says**

OCTOBER 17, 2017 AT 12:35 PM

Please, can u explain in more detail item 8 , that says Java stack is faster than heap

[Reply](#)**sonal balekai says**

OCTOBER 10, 2017 AT 12:10 AM

super explanation than q

[Reply](#)**Wing says**

OCTOBER 8, 2017 AT 9:48 PM

Very nice and clear! Thanks a lot!

[Reply](#)**Helal Uddin says**

SEPTEMBER 23, 2017 AT 10:27 PM

very well explained

[Reply](#)**Sana Shaikh says**

AUGUST 9, 2017 AT 10:31 PM

Wooooow.....well explained...Short and sweet...Thanks alot

[Reply](#)**Harsh Mudgal says**

AUGUST 3, 2017 AT 7:20 AM

Nice Article and you explained it well.

[Reply](#)

**Kamesh B says**

MAY 25, 2017 AT 4:48 AM

Great article.

[Reply](#)**Java Reader says**

JULY 10, 2017 AT 7:31 PM

Great post .Thanks

[Reply](#)**chandrashekhar says**

MAY 24, 2017 AT 11:47 AM

class Demo{

public static void main(String[] args)

{

Demo d=new Demo();

}

my question is how many memory allocate for object in heap

[Reply](#)**Krishna Choudhary says**

JUNE 20, 2017 AT 9:13 PM

Just one object in heap, and one reference variable in stack.

[Reply](#)**Needa says**

MAY 1, 2017 AT 8:49 AM

This is super helpful!! Thank you so much for the video!

Really good explanation sir!!

[Reply](#)

**lova says**

APRIL 20, 2017 AT 6:46 PM

Very Nice Explanation .. Thank you

[Reply](#)**Jitendra Nandiya says**

APRIL 5, 2017 AT 10:52 PM

Very nice Example.....good

[Reply](#)**chandan says**

MARCH 15, 2017 AT 11:39 AM

nice tutorial

thanks

[Reply](#)**Parjanya Roy says**

MARCH 2, 2017 AT 2:05 AM

Great explanation .

I feel the following link would complete this topic.

<https://www.mkyong.com/java/find-out-your-java-heap-memory-size/>[Reply](#)**Hardik Doshi says**

JUNE 15, 2017 AT 3:47 AM

Good reference.

[Reply](#)**Lukasz says**

FEBRUARY 25, 2017 AT 1:40 PM

wow, I'm impressed, looks like one of the best examples I've seen so far (and I've seen quite a few).

[Reply](#)

**Rajat says**

FEBRUARY 6, 2017 AT 7:51 AM

nice!

[Reply](#)**sanjay says**

FEBRUARY 2, 2017 AT 1:18 AM

Thanks pankaj that was very useful. I have related question if you could help. I have a PatriciaTrie list that holds item but there is not limit to the no of item . so when the item size is huge it throws Java heap error . there no way to increase it any more using -Xms am-Xmx . the only thing is that should be cached and proper message to be shown to the user (now it gives me a internal error which is not useful to the user). But the problem is i am not able to catch this exception. Any suggestions.

[Reply](#)**Murugesh says**

FEBRUARY 1, 2017 AT 4:46 AM

if we declare a variables in static method then wether it stores on heap or stack. Please explain me.

[Reply](#)**Narendar says**

JANUARY 30, 2017 AT 11:05 PM

Its very nice explanation.

[Reply](#)**Eric Wang says**

JANUARY 20, 2017 AT 12:25 AM

Thank you !

[Reply](#)**Santhosh says**

JANUARY 17, 2017 AT 12:09 AM

Hi Pankaj,

Its very nice explanation you have given. But I have small query about below code,

```
public class MemoryMgmt {  
    static Integer a = new Integer(12);  
    public static void main(String[] args) {  
        Integer b = new Integer(12);  
        int c = 12;  
        System.out.println("a==b "+(a==b));  
        System.out.println("b==c "+(b==c));  
        System.out.println("a==c "+(a==c));  
    }  
}
```

Can you please explain the memory allocation, where those variables are stored in memory.

[Reply](#)

**Tamil says**

APRIL 21, 2017 AT 6:47 AM

It's very easy to understand. can you please notify which one is best? Stack or Heap

[Reply](#)

**fred says**

JANUARY 11, 2017 AT 5:52 AM

Very good!

[Reply](#)

**Sanjeev says**

JANUARY 18, 2017 AT 9:42 PM

Stack memory is static, and heap memory is dynamic.(then read properly what is written)

Second - \_ why a method always contain 2 stack, if that method has no body.

Third- difference between iload and aload. How it is used in stack.

If any one plz explain.

[Reply](#)

**Sanjeev says**

JANUARY 18, 2017 AT 9:44 PM

In Java variable is always pass by reference not value.

[Reply](#)

**Miroslav Manchev says**

FEBRUARY 7, 2017 AT 6:06 AM

No, it is always pass by value. In some cases value is a primitive and in some cases value is a reference to an object.

[Reply](#)**Ajay says**

DECEMBER 2, 2016 AT 5:23 PM

Thanks Pankaj. You explained a lot in less time and easy manner.

[Reply](#)**Sailaja Gudala says**

NOVEMBER 21, 2016 AT 10:23 PM

Wonderful explanation

[Reply](#)**Saurabh says**

NOVEMBER 3, 2016 AT 3:55 AM

Superb Explain Pankaj..... After a long time now i got the proper answer HERE.... Hats Off to you

[Reply](#)**Jun Hua says**

NOVEMBER 2, 2016 AT 8:03 PM

Great explanation!

[Reply](#)**Gafoor says**

NOVEMBER 2, 2016 AT 5:13 AM



The way explained is easily grasp difference between Heap Space vs Stack – Memory.  
thanks for publishing this article.

[Reply](#)

**LittleLion says**

OCTOBER 12, 2016 AT 1:06 PM

String s1=new String("hello");

As I know two objects will be created..but interviewer asked to prove..Pls help me out @pankaj..

[Reply](#)

**Rakesh says**

OCTOBER 18, 2016 AT 5:39 AM

java creates an object xyz in memory and then a x object which has the ingredients of object xyz.  
Then xyz is destroyed (lost). So at the end we created 2 object but now we have one !

[Reply](#)

**Roy says**

OCTOBER 26, 2016 AT 9:09 AM

Rakesh, thank you for the excellent article.

For LittleLion's example, is the way bellow explained correctly or not?

String s1 = new String("Hello");

"Hello" is a String literal, so JVM creates "Hello" Object and put into the String pool (String interning).

Then use "Hello" to create another String Object and put into Heap. (String Object).

Thats why two object created.

Is this right?

[Reply](#)

**Sridam biswas says**

FEBRUARY 22, 2017 AT 9:31 AM

yes! Roy you are right!...there is a pool of strings in JAVA by named PermGen .first JVM check that pool for literal(one object) if it is present then it will return to the object(another) by creating new operator..that is the way of string creating object...

[Reply](#)

**Prabhat Kumar says**

SEPTEMBER 29, 2016 AT 3:41 AM

Hi

Thanks for this awesome brief about heap and stack memory.

[Based on the above explanations, we can easily conclude following differences between Heap and Stack memory.

Heap memory is used by all the parts of the application whereas stack memory is used only by one thread of execution.]

Question: Block for main thread always exists. So as per me stack memory is used only by one thread of execution is not totally correct. Please explain me if I am wrong.

[Reply](#)**Nagaraju says**

SEPTEMBER 25, 2016 AT 12:44 AM

Good work boss..

[Reply](#)**Reddeppa Kande says**

SEPTEMBER 21, 2016 AT 6:06 PM

Nice article. a small suggestion in diagram.. can you write steps 1,2,3... so that when step 7 occurs where is the flow can be easily referred for beginners

[Reply](#)**rungrengreng says**

AUGUST 8, 2016 AT 9:29 PM

Thank for your great work! This is interesting!

[Reply](#)**Jim Mehta says**

AUGUST 7, 2016 AT 9:16 PM

I watched the video. It's excellent. But I have a question about who collects Stack Memory? References will always be in the Stack and as soon as the reference goes out of scope object is available for the Garbage collection but what happens to the reference memory in the stack?

[Reply](#)

**andrey says**

AUGUST 28, 2016 AT 5:50 AM

Stack frame memory gets deallocated automatically when method exists, because it's get popped from the stack.

[Reply](#)**Dheeraj says**

AUGUST 5, 2016 AT 1:26 AM

Hi ,

What is about

String args[] variable.

Will this reference variable will not be listed in stack of main method.

[Reply](#)**Mrs Afzal says**

JULY 4, 2016 AT 10:10 PM

I want to ask how we can elaborate using stack memory and heap memory that two methods are accessing to the same object means (This Reference and Other References to the same object at method level). The purpose is that we want to see the dependencies between two methods using same data.

[Reply](#)**Melwin says**

JUNE 20, 2016 AT 8:47 AM

Where would the references to an Object which is an instance variable be stored. For eg.

Class Memory{

Memory mem = new Memory();

public Memory(){

}

}

So, where would be the "mem" reference be stored as it is not tied to any code/method.

[Reply](#)**Satishchandra Singh says**

AUGUST 4, 2016 AT 10:57 PM

It would be stored in the heap along with all the other contents (instance primitive variables) in the heap.

[Reply](#)

**Sumant Dharkar says**

JUNE 3, 2016 AT 12:19 AM

where an object and its reference will create for all type of inner classes ?

[Reply](#)

**Selvakumar Esra says**

MAY 15, 2016 AT 7:01 AM

Java is both "Pass by value" and "Pass by reference"  
why

balloon.setColor("Red");

changed the value of color to "Red" when printed in main() method?

[Reply](#)

**Sarvesh Singh says**

APRIL 24, 2016 AT 7:20 PM

Good work! 👍

Thank you.... 😊

[Reply](#)

**Farhan says**

APRIL 13, 2016 AT 6:50 AM

This is good explanation.

Can you give explanation on this involving array manipulation?

[Reply](#)

**Sid says**

APRIL 12, 2016 AT 5:14 AM

Loved the way you have written this article. Excellent. Understood the concept.

Thanks,

Cheers

[Reply](#)

**Ram says**

APRIL 11, 2016 AT 4:15 AM

Excellent Boss !!! clear explanation

[Reply](#)

**CS Student says**

APRIL 1, 2016 AT 6:35 AM

Well done! This really helped me.

[Reply](#)

**Pratap says**

MARCH 3, 2016 AT 5:43 AM

How much memory is allocated in heap for String ?

for example if i define String name; in class

[Reply](#)

**Sushant says**

JANUARY 28, 2016 AT 8:24 AM

Hello Pankaj,

Above explanation of memory allocation is very nice and useful. But I am still confused for following scenario :

```
Class A {  
    public A(){...}  
    public void test(){...}  
    public void test1(){...}  
}  
  
class B extends A {  
    public B(){...}  
    public void test(){...}  
    public void test2(){...}  
}
```

now in main method I am creating object in this way :

```
A a = new B();  
a.test();  
a.test1();  
a.test2();
```

so how memory allocation will work ?  
Thanks in Advance.

[Reply](#)**mista says**

FEBRUARY 5, 2016 AT 7:19 AM

that is not going to compile cause class A doesn't have test2() method.

[Reply](#)**Lucian says**

JANUARY 16, 2016 AT 10:36 AM

Very good explanation . Keep doing a great job !

[Reply](#)**Jinesh says**

JANUARY 9, 2016 AT 2:31 AM

Thanks a lot, very well framed. Easy & Simple to understand

[Reply](#)**Swapnil Suhane says**

JANUARY 8, 2016 AT 9:21 AM

Great work..written in simple way to understand it ☐

You can add some advanced level theory also later to make this article perfect like...static/non static context etc

[Reply](#)**arun says**

DECEMBER 13, 2015 AT 12:18 AM

Hi,

I have one question. i faced this question in one of the interview,

```
String s1=new String("abc");
```

```
String s2=new String("abc");
```

```
int a=10;
```

```
int b=10;
```

if a==b is true then why

s1==s2 is false

[Reply](#)

**Pankaj says**

DECEMBER 13, 2015 AT 9:31 AM

1. int is not an object, java handles primitive data types differently, reducing memory footprint by caching the variables, something like String Pool.
2. String is an object, when new operator is used then a new variable is created rather than checking in String Pool to find some already existing one.

Based on above points, I think you can clearly see the reason behind the program output.

[Reply](#)

**Niraj says**

JANUARY 7, 2016 AT 9:40 PM

```
StringBuilder sb1 = new StringBuilder("Niraj");
```

```
StringBuilder sb2 = new StringBuilder("Niraj");
```

```
System.out.println("sb1.equals(sb2) "+(sb1.equals(sb2)));
```

Then why this is give "false"

[Reply](#)

**Alex says**

JANUARY 15, 2016 AT 10:25 AM

That gives 'false' because you are comparing two unique StringBuilder objects, not the actual String values. If you changed your code to this:

```
sb1.toString().equals(sb2.toString())
```

That should give you 'true'. Remember, when you have an object (like String) it has an object reference (or ID) as well as an actual "value". If you compare the object references, you will find they are different, even if the object values are the same.

[Reply](#)

**varun says**

DECEMBER 8, 2015 AT 10:54 PM

very good explanation. keep it up!

[Reply](#)**Rangrao Patil says**

DECEMBER 3, 2015 AT 7:07 AM

What is the stack memory allocated to each java program?.

What is the size of stack memory?

[Reply](#)**Akanksha says**

OCTOBER 22, 2015 AT 6:40 PM

Very well explained

[Reply](#)**Krishna says**

OCTOBER 9, 2015 AT 10:25 AM

Awesome explanation, thank you so much.

[Reply](#)**selvi says**

OCTOBER 7, 2015 AT 11:39 AM

thnks a lot

[Reply](#)**satyam says**

SEPTEMBER 19, 2015 AT 8:45 PM

simplicity makes this article awesome

[Reply](#)



**Shambhu says**

SEPTEMBER 14, 2015 AT 3:01 AM

You mentioned stand alone program. So My question is once application finish executing, what will happen to String pool, it will be memory or it will remove from heap?

[Reply](#)**Pankaj says**

SEPTEMBER 14, 2015 AT 4:56 AM

Once application is finished executing, JVM terminates and all the memory is released to OS.

[Reply](#)**Amit says**

SEPTEMBER 1, 2015 AT 4:33 AM

Nice Article

my doubt is

```
class Memory{
```

```
Memory memory=new Memory();
```

```
}
```

where memory is get stored....on heap??

[Reply](#)**satyam says**

SEPTEMBER 19, 2015 AT 8:40 PM

stack

[Reply](#)**cc10123 says**

NOVEMBER 22, 2015 AT 6:37 PM

Heap.

The Memory object would always be created on the heap, regardless of where its reference lives. In your example, the memory (small 'm') reference is also on the heap, because it exists as part of the Memory class which is on the heap.

[Reply](#)

**z6qc@yahoo.com says**

AUGUST 4, 2015 AT 2:41 AM

not all objects allocated in Heap ! but static ones ,  
you created Memory and Object instances in a static context so the variables in that context created in heap !  
if you create them in foo method stack memory will be used ,  
please correct this big wrong article!!!

[Reply](#)**Pankaj says**

AUGUST 5, 2015 AT 5:52 PM

i have no idea what you are talking here.

[Reply](#)**PriyaYT says**

DECEMBER 3, 2015 AT 11:56 PM

We mean that in foo(Object param)  
here param is a local variable of type Object in foo() so it won't get space in heap but according to ESCAPE ANALYSIS(by java 6e14) that param will be treated similar to those local variables of primitive types in foo() and gets space in stack and not in heap.!!  
Will you please clear this out Sir.  
Thank You.

[Reply](#)**Anirban Pal says**

NOVEMBER 10, 2016 AT 7:32 AM

You are right...but escape analysis is not always possible, In this trivial case it was easy for the compiler to find out.

[Reply](#)**shneha says**

JULY 23, 2015 AT 11:11 PM

easily i got understand by reading this... thanks to clear my doubt

[Reply](#)

**Subrahmanyam says**

JUNE 29, 2015 AT 3:42 AM

Very nice and useful Information.. Thanks a lot ☐

[Reply](#)**YangFang says**

JUNE 12, 2015 AT 8:12 PM

Very well! I learn o lot form this passage!

[Reply](#)**Arjun says**

MAY 26, 2015 AT 9:24 AM

Thanks dude. Thanks for explaining it neat and clear.. (y)  
cheers!

[Reply](#)**Huy says**

MAY 21, 2015 AT 9:25 PM

Thanks man, you made my day ☐

[Reply](#)**Suraj Shrestha says**

MAY 16, 2015 AT 1:01 PM

Nice Article, I have one question. Do we have separate stack memory for each Thread we create?

[Reply](#)**Alexander Orlov says**

FEBRUARY 26, 2015 AT 11:49 PM

Java is not only "Pass by Value", it's also "Pass by Reference" which depends if primitives are passed or reference objects (the name implies it). If Java would be purely "Pass by Value", the following program

```

public class Test {
    public static void main(String[] args) {
        Integer i = 2;
        System.out.println("i initialized = " + i);
        List u = new ArrayList(Arrays.asList("1", "2", "3"));
        System.out.println("u before method = " + u);
        test(i);
        test1(u);
        System.out.println("i outside method = " + i);
        System.out.println("u AFTER method call = " + u);
    }

    private static void test1(List u) {
        u.add("blah");
        System.out.println("u inside method = " + u);
    }

    private static void test(int i) {
        i = i + 1;
        System.out.println("i inside method = " + i);
    }
}

```

...would not print  
u inside method = [1, 2, 3, blah]  
[...]  
u AFTER method call = [1, 2, 3, blah]

[Reply](#)**Purnendu Dutta says**

AUGUST 23, 2015 AT 2:39 PM

In the test(int i) method, when i is being printed its the local variable value which is a copy of i in main(), coming back in main() when i is printed the value still remains same hence its PassByValue.

[Reply](#)**Miroslav Manchev says**

FEBRUARY 7, 2017 AT 6:27 AM

In both cases it's pass by value. In case of 'i', value of 'i' is copied in the method 'test()'. In case of 'u', value of 'u' which is a reference, is copied in method 'test1()'. Check this code:

```

public static void main(String[] args) {
    List list = Arrays.asList("One", "Three", "Five");
    passByReferenceTest(list);
    System.out.println(list); // n1: What is printed here?
}

```

```
private static void passByReferenceTest(List list) {  
list = Arrays.asList("Two", "Four", "Six"); // n2  
}
```

If pass by reference was possible then the assignment at line n2 would have changed the list reference in main() and therefore "Two", "Four", "Six" would have been printed at line n1. Pointers in C behave like that.

[Reply](#)

**Upir says**

FEBRUARY 17, 2015 AT 1:10 AM

Nice video, thanks !

[Reply](#)

**Pieter says**

JANUARY 12, 2015 AT 11:36 PM

Very well explained, thanks!

[Reply](#)

**Prabhath says**

JANUARY 4, 2015 AT 8:22 AM

Nice Article.

[Reply](#)

**Vel says**

DECEMBER 16, 2014 AT 4:50 AM

I have a doubt in static reference and static primitive.

```
public static A a = new A();
```

```
public static int b = 5;
```

where 'a' reference and 'b' primitive live (stack or heap). When it will eligible for garbage collection.

[Reply](#)

**Sudhakar says**

DECEMBER 14, 2014 AT 5:36 AM

Crisp, Clear and to the point...great article. keep rocking!!!

[Reply](#)

### Alalonks says

DECEMBER 5, 2014 AT 4:49 PM

If it were truly "pass by value" as in the case with c++ then it would be a copy/clone of the object. Calling the foo() method would not affect the blue balloon color value in the main method. But in the example it does because it is "referencing" the memory space in the heap and changing the value. Much like you would when "passing by reference".

However, I also see the point of how the value is used (as if a copy/clone were passed in) in the swap method. The manipulation of the balloon colors in the swap method does not affect the balloons variables in the main method. Its almost like its "pass by reference or value".

In either case the presentation was awesome as well as the write up. Thanks for your efforts.

[Reply](#)

### Arfeen says

DECEMBER 2, 2014 AT 12:11 PM

Thank you for sharing this article.

One point i have to raise here. regarding "int i = 1". Not sure if this is because of "escape analysis" or something else like "interning". But what i tested it goes to "Heap".

Following is my test:

```
package immutable;
public class Immutable_One {
private int i = 2;
public int get() {
return i;
}
}
```

```
package immutable;
public class Immutable_Two {
private int j = 2;
public int get() {
return j;
}
}
```

```
package immutable;
```

```
public class ImmutableTest {  
    public static void main(String[] args) {  
        Immutable_One immutable_One = new Immutable_One();  
        int i = immutable_One.get();  
        Immutable_Two immutable_Two = new Immutable_Two();  
        int j = immutable_Two.get();  
        if (i == j) {  
            System.out.println("Equal");  
        } else {  
            System.out.println("NOT Equal");  
        }  
    }  
}
```

The result comes "Equal".

[Reply](#)

#### Tanzy says

DECEMBER 4, 2014 AT 2:43 PM

Its basically because of the caching.

If we open the Integer class, you can see that value between -128 to 127 are cached.

In the above program if you put any value outside of the range, you will the desired behavior.

So as per my thought- What Pankaj mentioned in picture "int i = 1" as part of stack, that must be re-corrected and move it to heap area which can be shared among all thread.

[Reply](#)

#### cc10123 says

NOVEMBER 22, 2015 AT 6:51 PM

I don't see how this is a valid test. You're comparing the value of two primitives, which is a straight up comparison of their values. No references involved at that point. For a better test, make i and j Integer objects, and return Integer objects. Than you'll see the caching behavior that Tanzy mentions.

[Reply](#)

#### Michael says

MARCH 27, 2016 AT 1:44 AM

Agree. He is just comparing primitive values, 2 == 2 returns true.

He should try something like this:

```
Integer a = Integer.valueOf(2);
```

```
Integer b = Integer.valueOf(2);
```

```
System.out.println(a.equals(b)); // comparing values
System.out.println(a==b); // comparing references
Integer c = new Integer(2);
Integer d = new Integer(2);
System.out.println(c.equals(d)); // comparing values
System.out.println(c==d); // comparing references
```

[Reply](#)**Vishnu says**

NOVEMBER 27, 2014 AT 8:40 AM

Nice artical , greate and simple explanation .

[Reply](#)**Abhishek Thakur says**

NOVEMBER 20, 2014 AT 4:14 AM

This is a very nice explanation of Stack and Heap Memory and their differences. I wish you could write a bit about instance variables and static variables. Well Done !

[Reply](#)**i\_know\_nothing says**

OCTOBER 28, 2014 AT 7:55 AM

Thanks a lot Pankaj.

This article is very great!

Can You write a article about Thread-Safe? Because I don't know Why a global variable is not-thread-safe.

I reseach very much about it. I use "new" operation but a global variable in that class, it's not Thread-Safe.

Thanks in advance!

[Reply](#)**Dmitry says**

SEPTEMBER 4, 2014 AT 12:09 PM

Thanks for your articles. They are very usefull!

[Reply](#)



**Shaik says**

AUGUST 31, 2014 AT 9:40 PM

On the public static void main(String[] args) – Does the memory for args[] allocated in heap space but the reference is allocated in Stack – Is my understanding correct?

[Reply](#)**Pankaj says**

AUGUST 31, 2014 AT 11:55 PM

Yes, it's String array.. hence Object, so stored in Heap memory. However main method contains the references in stack memory.

[Reply](#)**Sorrowfull Blinger says**

AUGUST 28, 2014 AT 2:03 AM

Awesome ! best place to brush up concepts !

[Reply](#)**kanaga says**

AUGUST 20, 2014 AT 2:18 AM

This is very useful. Thanks for your great work. public static void main(String[] args) – what about args? It is stored in stack right?

[Reply](#)**pk says**

AUGUST 19, 2014 AT 11:58 AM

what about static methods and static variables? are they stored in stack or heap memory?

[Reply](#)**Sankar says**

JULY 31, 2016 AT 2:42 AM

class structure and static variables are stored in a separate memory pool area known as Method Area.

[Reply](#)**johnsonjeven says**

MARCH 30, 2017 AT 11:27 PM

Static Variables are stored in HEAP. Classes and all of the data applying to classes (not instance data) is stored in the Permanent Generation section of the heap.

<http://net-informations.com/java/cjava/memory.htm>

John

[Reply](#)

## Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment

Name \*

Email \*

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

---

[DOWNLOAD ANDROID APP](#)

---



---

## CORE JAVA TUTORIAL

---

[Java 10 Tutorials](#)   [Java 9 Tutorials](#)  
[Java 8 Tutorials](#)   [Java 7 Tutorials](#)   [Core Java Basics](#)   [OOPS Concepts](#)   [Data Types and Operators](#)   [String Manipulation](#)  
[Java Arrays](#)   [Annotation and Enum](#)  
[Java Collections](#)   [Java IO Operations](#)  
[Java Exception Handling](#)  
[MultiThreading and Concurrency](#)  
[Regular Expressions](#)   [Advanced Java Concepts](#)

---

---

## IMPORTANT INTERVIEW QUESTIONS

---

# Java Interview Questions

- > [Core Java Interview Questions](#)
- > [String Interview Questions](#)
- > [Multithreading Interview Questions](#)
- > [Collections Interview Questions](#)
- > [Exception Interview Questions](#)
- > [Java Programming Interview Questions](#)
- > [Java 8 Interview Questions Part 1](#)
- > [Java 8 Interview Questions Part 2](#)
- > [Servlet Interview Questions](#)
- > [JSP Interview Questions](#)
- > [Struts 2 Interview Questions](#)
- > [JDBC Interview Questions](#)
- > [Spring Interview Questions](#)
- > [Hibernate Interview Questions](#)
- > [JSF Interview Questions](#)
- > [Web Services Interview Questions](#)
- > [Scala Basic Interview Questions](#)
- > [Scala Intermediate Interview Questions](#)
- > [Scala Advanced Interview Questions](#)

- > [Scala Interview Questions Summary](#)
- > [Common Job Interview Questions](#)

Miscellaneous

- > [Java ClassLoader](#)
- > [String StringBuffer StringBuilder](#)
- > [Java is Pass By Value](#)
- > [Java Heap vs Stack Memory](#)

RECOMMENDED TUTORIALS

Java Tutorials

- > [Java IO](#)
- > [Java Regular Expressions](#)
- > [Multithreading in Java](#)
- > [Java Logging](#)
- > [Java Annotations](#)
- > [Java XML](#)
- > [Collections in Java](#)
- > [Java Generics](#)
- > [Exception Handling in Java](#)
- > [Java Reflection](#)
- > [Java Design Patterns](#)
- > [JDBC Tutorial](#)


Java EE Tutorials

- > [Servlet JSP Tutorial](#)
- > [Struts2 Tutorial](#)
- > [Spring Tutorial](#)
- > [Hibernate Tutorial](#)
- > [Primefaces Tutorial](#)
- > [Apache Axis 2](#)
- > [JAX-RS](#)
- > [Memcached Tutorial](#)

© 2018 · Privacy Policy · Don't copy, it's Bad Karma · Powered by W


JAVA  
FILE APIs

- CONVERT
- PRINT
- CREATE
- COMBINE
- MODIFY




Files from  
your Code

100% Native  
Java



Try FREE Today!

 **ASPOSE**  
File Format APIs