

Search

[Home](#) [core java](#) [coding](#) [thread](#) [sql](#) [java 8](#) [books](#) [array](#) [string](#) [j2ee](#) [oop](#) [debugging](#) [collections](#) [data structure](#)**OCAJP 7 or OCAJP 8?
Which Java Certification
will you take?**

- ☐ OCAJP7
☐ OCAJP8

 [Show results](#)

Votes so far: 694
Days left to vote: 328

Categories

- [core java](#) (296)
- [programming](#) (184)
- [core java interview question answer](#) (86)
- [Java collection tutorial](#) (71)
- [interview questions](#) (56)
- [coding](#) (51)
- [java](#) (50)
- [Coding Problems](#) (40)
- [Java programming Tutorial](#) (32)
- [error and exception](#) (30)
- [data structure and algorithm](#) (23)
- [homework](#) (21)
- [Java 8](#) (19)
- [collection tutorial example](#) (19)
- [books](#) (15)
- [J2EE](#) (14)
- [String](#) (14)
- [date and time](#) (14)
- [thread](#) (14)
- [sql](#) (13)
- [SQL interview Question](#) (12)
- [Java 5 tutorial](#) (11)
- [array](#) (11)
- [java concurrency tutorial](#) (11)
- [java io tutorial](#) (11)
- [JSP](#) (10)
- [Java Multithreading Tutorial](#) (10)
- [Servlet](#) (10)
- [thread interview questions](#) (10)
- [database](#) (8)
- [object oriented programming](#) (8)
- [JDBC](#) (6)
- [Eclipse](#) (5)
- [basics](#) (5)
- [troubleshooting](#) (5)
- [JEE Interview Questions](#) (4)
- [JavaScript](#) (4)
- [MySQL tutorial example](#) (4)
- [design pattern](#) (4)
- [java design pattern](#) (4)
- [JSP Interview Question](#) (3)
- [Java Enum](#) (3)
- [Linux](#) (3)
- [OCAJP](#) (3)
- [OOPS](#) (3)
- [Struts](#) (3)
- [Web Service](#) (3)
- [debugging](#) (3)
- [enum](#) (3)
- [Hibernate interview Question](#) (2)

Top 10 Tricky Java interview questions and Answers

What is a tricky question? Well, tricky Java interview questions are those questions which have some surprise element on it. If you try to answer a tricky question with common sense, you will most likely fail because they require some specific knowledge. Most of the tricky Java questions comes from confusing concepts like function overloading and overriding, Multi-threading which is really tricky to master, character encoding, **checked vs unchecked exceptions** and subtle Java programming details like Integer overflow. Most important thing to answer a tricky Java question is attitude and analytical thinking, which helps even if you don't know the answer. Anyway in this Java article we will see 10 Java questions which are real tricky and requires more than average knowledge of Java programming language to answer them correctly. As per my experience, there is always one or two tricky or **tough Java interview question** on any core Java or J2EE interviews, so it's good to prepare tricky questions from Java in advance.

If I take an interview, I purposefully put this kind of question to gauge the depth of candidate's understanding in Java. Another advantage of asking such question is the surprising element, which is a key factor to put the candidate on some pressure during interviews.

Since these questions are less common, there is good chance that many Java developer doesn't know about it. You won't find these questions even on popular Java interview books like **Java Programming Interview exposed**, which is nevertheless an excellent guide for Java interviews.

Btw, if you don't find these question tricky enough, then you should check Joshua Bloch's another classic book, **Java Puzzlers** for super tricky questions. I am sure you will find them challenging enough.

10 Tricky Java interview question - Answered

Here is my list of 10 tricky Java interview questions, Though I have prepared and shared lot of difficult core Java interview question and answers, But I have chosen them as Top 10 tricky questions because you can not guess answers of this tricky Java questions easily, you need some subtle details of Java programming language to answer these questions.

Question: What does the following Java program print?

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println(Math.min(Double.MIN_VALUE, 0.0d));  
    }  
}
```

Answer: This question is tricky because unlike the **Integer**, where **MIN_VALUE** is negative, both the **MAX_VALUE** and **MIN_VALUE** of the **Double** class are positive numbers. The **Double.MIN_VALUE** is 2^{-1074} , a double constant whose magnitude is the least among all double values. So unlike the obvious answer, this program will print 0.0 because **Double.MIN_VALUE** is greater than 0. I have asked this question to Java developer having experience up to 3 to 5 years and surprisingly almost 70% candidate got it wrong.

What will happen if you put return statement or System.exit () on try or catch block? Will finally block execute?

This is a very popular tricky Java question and it's tricky because many programmers think that no matter what, but the **finally block** will always execute. This question challenge that concept by putting a return statement in the try or catch block or calling **System.exit ()** from try or catch block. Answer of this tricky question in Java is that finally block will execute even if you put a return statement in the try block or catch block but finally block won't run if you call **System.exit ()** from try or catch block.

Question: Can you override a private or static method in Java?

Another popular Java tricky question, As I said method overriding is a good topic to ask trick questions in Java. Anyway, **you can not override a private or static method in Java**, if you create a similar method with same return type and same method arguments in child class then it will hide the superclass method, this is known as method hiding.

Similarly, you cannot override a private method in sub class because it's not accessible there, what you do is create another private method with the same name in the child class. See **Can you override a private method in Java** or more details.

Question: What do the expression 1.0 / 0.0 will return? will it throw Exception? any compile time error?

Answer: This is another tricky question from Double class. Though Java developer knows about the double primitive type and Double class, while doing floating point arithmetic they don't pay enough attention to **Double.INFINITY**, **NaN**, and **-0.0** and other rules that govern the arithmetic calculations

Interview Questions

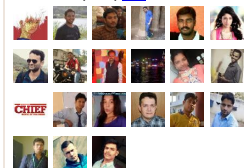
- [core java interview questions](#)
- [programming interview questions](#)
- [SQL interview questions](#)
- [data structure interview question](#)
- [coding interview questions](#)
- [java collection interview questions](#)
- [java design pattern interview questions](#)
- [thread interview questions](#)
- [hibernate interview questions](#)
- [jdbc interview questions](#)
- [array interview questions](#)
- [j2ee interview questions](#)
- [spring interview questions](#)

Books and Resources

- [6 Website to Learn JavaScript Online](#)
- [Top 10 Java 8 Tutorials](#)
- [5 Free JavaScript Books for Web Developers](#)
- [Best Book to Learn Java Programming](#)
- [Top 3 Free Struts Books for Java EE developers](#)
- [5 Books to Improve Your Coding Skill](#)
- [10 Books Every Programmer Should Read](#)
- [10 Computer Algorithm Books for Programmers](#)
- [10 Free Java Programming Books](#)
- [5 Books to Learn Java 8 Better](#)

Follow by Email **Search This Blog** **Recommended Reading**

- [The Best Book to Learn Java in 30 days](#)
- [10 Java Web Service Interview Questions](#)
- [Top 10 Android Interview Questions for Java Programmers](#)
- [How to use an ArrayList in Java?](#)
- [10 Books Every Programmer Should Read](#)
- [5 Great Books to Learn Java 8](#)

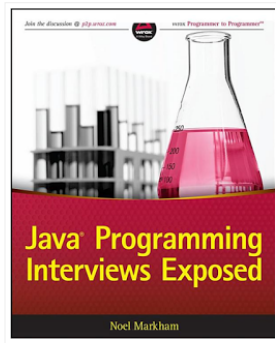
Followers**Followers (1001)** [Next](#)**Blog Archive**

- **2016** (154)
- ▼ **2015** (102)
 - **December** (8)
 - **November** (4)
 - ▼ **October** (13)
 - [2 Ways to find duplicate elements in an Array - Ja...](#)
 - [How to Sort List into Ascending and Descending Ord...](#)
 - [Top 10 Tricky Java interview questions and Answers...](#)
 - [How to Print Pyramid Pattern in Java? Program Exam...](#)
 - [Java ArrayList Tutorial - The MEGA List](#)
 - [How to solve FizzBuzz in Java?](#)
 - [How to convert float to int in Java? Examples](#)
 - [Top 5 FREE JavaScript Books - Download PDF or Read...](#)
 - [How to declare ArrayList with values in Java? Exam...](#)
 - [Java - How to convert from Integer to String?](#)

involving them. The simple answer to this question is that it will not throw `ArithmeticException` and return `Double.INFINITY`.

Also, note that the comparison `x == Double.NaN` always evaluates to false, even if `x` itself is a `NaN`. To test if `x` is a `NaN`, one should use the method call `Double.isNaN(x)` to check if given number is `NaN` or not. If you know SQL, this is very close to `NULL` there.

Btw, If you are running out of time for your interview preparation, you can also check out [Java Programming Interviews exposed](#) for more of such popular questions,



Does Java support multiple inheritances?

This is the trickiest question in Java if C++ can support direct multiple inheritances then why not Java is the argument Interviewer often give. Answer of this question is much more subtle then it looks like, because Java does support multiple inheritances of Type by allowing an interface to extend other interfaces, what Java doesn't support is multiple inheritances of implementation. This distinction also gets blur because of default method of Java 8, which now provides Java, multiple inheritances of behavior as well. See why multiple inheritances are not supported in Java to answer this tricky Java question.

What will happen if we put a key object in a HashMap which is already there?

This tricky Java question is part of another frequently asked question, How HashMap works in Java. HashMap is also a popular topic to create confusing and tricky question in Java. Answer of this question is if you put the same key again then it will replace the old mapping because HashMap doesn't allow duplicate keys. The Same key will result in the same hashCode and will end up at the same position in the bucket.

Each bucket contains a linked list of `Map.Entry` object, which contains both Key and Value. Now Java will take the Key object from each entry and compare with this new key using `equals()` method, if that return true then value object in that entry will be replaced by new value. See [How HashMap works in Java](#) for more tricky Java questions from HashMap.

Question: What does the following Java program print?

```
public class Test {
    public static void main(String[] args) throws Exception {
        char[] chars = new char[] {'\u0097'};
        String str = new String(chars);
        byte[] bytes = str.getBytes();
        System.out.println(Arrays.toString(bytes));
    }
}
```

Answer: The trickiness of this question lies on character encoding and how String to byte array conversion works. In this program, we are first creating a String from a character array, which just has one character `'\u0097'`, after that we are getting the byte array from that String and printing that byte. Since `\u0097` is within the 8-bit range of byte primitive type, it is reasonable to guess that the `str.getBytes()` call will return a byte array that contains one element with a value of -105 (byte) `0x97`.

However, that's not what the program prints and that's why this question is tricky. As a matter of fact, the output of the program is operating system and locale dependent. On a Windows XP with the US locale, the above program prints `[63]`, if you run this program on Linux or Solaris, you will get different values.

To answer this question correctly, you need to know about how Unicode characters are represented in Java char values and in Java strings, and what role character encoding plays in `String.getBytes()`.

In simple word, to convert a string to a byte array, Java iterate through all the characters that the string represents and turn each one into a number of bytes and finally put the bytes together. The rule that maps each Unicode character into a byte array is called a character encoding. So It's possible that if same character encoding is not used during both encoding and decoding then retrieved value may not be correct. When we call `str.getBytes()` without specifying a character encoding scheme, the JVM uses the default character encoding of the platform to do the job.

The default encoding scheme is operating system and locale dependent. On Linux it is `UTF-8` and on

Error: could not open
'C:\Java\jre8\lib\amd64\jvm....

Avoid
ConcurrentModificationExcepti
on while loopin...

How to find highest repeating
word from a text Fil...

- September (10)
- August (10)
- July (18)
- June (21)
- May (5)
- April (1)
- March (3)
- February (2)
- January (7)

- 2014 (67)
- 2013 (44)
- 2012 (119)

The default encoding scheme is operating system and locale dependent. On Linux, it is UTF-8 and on Windows with a US locale, the default encoding is Cp1252. This explains the output we get from running this program on Windows machines with a US locale. No matter which character encoding scheme is used, Java will always translate Unicode characters not recognized by the encoding to 63, which represents the character U+003F (the question mark, ?) in all encodings.

If a method throws NullPointerException in the superclass, can we override it with a method which throws RuntimeException?

One more tricky Java questions from the overloading and overriding concept. The answer is you can very well throw superclass of RuntimeException in overridden method, but you can not do same if its checked Exception. See [Rules of method overriding in Java](#) for more details.

What is the issue with following implementation of compareTo() method in Java

```
public int compareTo(Object o){
    Employee emp = (Employee) o;
    return this.id - e.id;
}
```

where an id is an integer number.

Well, there is nothing wrong in this Java question until you guarantee that id is always positive. This Java question becomes tricky when you can't guarantee that id is positive or negative. the tricky part is, If id becomes negative than **subtraction may overflow** and produce an incorrect result. See [How to](#)

[override compareTo method in Java](#) for the complete answer of this Java tricky question for an experienced programmer.

How do you ensure that N thread can access N resources without deadlock?

If you are not well versed in writing multi-threading code then this is a real tricky question for you. This Java question can be tricky even for the experienced and senior programmer, who are not really exposed to deadlock and race conditions. The key point here is ordering, if you acquire resources in a particular order and release resources in the reverse order you can prevent deadlock. See [how to avoid deadlock in Java](#) for a sample code example.

Question: Consider the following Java code snippet, which is initializing two variables and both are not volatile, and two threads T1 and T2 are modifying these values as following, both are not synchronized

```
int x = 0;
boolean bExit = false;

Thread 1 (not synchronized)
x = 1;
bExit = true;

Thread 2 (not synchronized)
if (bExit == true)
System.out.println("x=" + x);
```

Now tell us, is it possible for Thread 2 to print "x=0"?

Answer: It's impossible for a list of tricky Java questions to not contain anything from multi-threading. This is the simplest one I can get. Answer of this question is Yes, It's possible that thread T2 may print x=0. Why? because without any instruction to compiler e.g. synchronized or volatile, bExit=true might come before x=1 in compiler reordering. Also, x=1 might not become visible in Thread 2, so Thread 2 will load x=0. Now, how do you fix it?

When I asked this question to a couple of programmers they answer differently, one suggests to make both threads synchronized on a common mutex, another one said make both variable volatile. Both are correct, as it will prevent reordering and guarantee visibility.

But the best answer is you just need to make bExit as volatile, then Thread 2 can only print "x=1". x does not need to be volatile because x cannot be reordered to come after bExit=true when bExit is volatile.

What is difference between CyclicBarrier and CountDownLatch in Java

Relatively newer Java tricky question, only been introduced from Java 5. The main difference between both of them is that you can reuse CyclicBarrier even if Barrier is broken, but you can not reuse CountDownLatch in Java. See [CyclicBarrier vs CountDownLatch in Java](#) for more differences.

What is the difference between StringBuffer and StringBuilder in Java?

Classic Java questions which some people think tricky and some consider very easy. StringBuilder in Java was introduced in JDK 1.5 and the only difference between both of them is that StringBuffer methods e.g. length(), capacity() or append() are **synchronized** while corresponding methods in StringBuilder are not synchronized.

Because of this fundamental difference, concatenation of String using StringBuilder is faster than StringBuffer. Actually, it's considered the bad practice to use StringBuffer anymore, because, in almost

Stun

ire
set
,

99% scenario, you perform string concatenation on the same thread. See [StringBuilder vs StringBuffer](#) for more differences.

Can you access a non-static variable in the static context?

Another tricky Java question from Java fundamentals. No, you can not access a non-static variable from

the static context in Java. If you try, it will give compile time error. This is actually a common problem beginner in Java face when they try to access instance variable inside the main method. Because main is static in Java, and instance variables are non-static, you can not access instance variable inside main. See, [why you can not access a non-static variable from static method](#) to learn more about this tricky Java questions.

How many String objects are created by the following code?

How many objects are created here?

String s = new String("abc");

- 1) One
- 2) Two
- 3) Three

Now, it's practice time, here are some questions for you guys to answer, these are contributed by readers of this blog, big thanks to them.

1. When doesn't Singleton remain Singleton in Java?
2. is it possible to load a class by two ClassLoader?
3. is it possible for equals() to return false, even if contents of two Objects are same?
4. Why compareTo() should be consistent to equals() method in Java?
5. When do Double and BigDecimal give different answers for equals() and compareTo() == 0.
6. How does "has before" apply to volatile work?
7. Why is $0.1 * 3 \neq 0.3$?
8. Why is (Integer) 1 == (Integer) 1 but (Integer) 222 != (Integer) 222 and which command arguments change this.
9. What happens when an exception is thrown by a Thread?
10. Difference between notify() and notifyAll() call?
11. Difference between System.exit() and System.halt() method?
12. Does following code legal in Java? is it an example of method overloading or overriding?

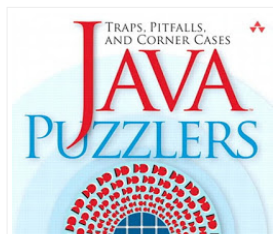
```
public String getDescription(Object obj){
    return obj.toString();
}
public String getDescription(String obj){
    return obj;
}
and
public void getDescription(String obj){
    return obj;
}
```

This was my list of Some of the most common tricky questions in Java. It's not a bad idea to prepare tricky Java question before appearing for any core Java or J2EE interview. One or two open-ended or tricky question is quite common in Java interviews.

Further Reading

If you are looking for super challenging trick coding questions then you should check out Joshua Bloch another classic book, the **Java Puzzlers**, I am sure you will find them really challenging to solve, I certainly

did.



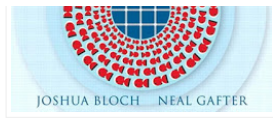
[Stun](#)

[a](#)

[ire](#)

[set](#)

[\)](#)



Hungry for more Java Interview Question and Answer post, check out these articles

- [18 Java design pattern question asked in interviews](#)
- [10 Java coding interview questions answer for 2 to 4 years experience](#)
- [Top 21 Most Frequently Asked Java Questions and Answers](#)
- [133 Core Java Interview Questions from last 5 years](#)
- [Top 50 Multithreading and Concurrency Interview Questions](#)
- [21 Frequently asked SQL queries from Java Interviews](#)

Related Java Tutorials

Posted by [Javin Paul](#) +55 Recommend this on Google

Labels: [core java interview question answer](#), [interview questions](#), [java](#)

64 comments:

Anonymous September 10, 2012 at 9:58 PM

Great questions, How about adding tricky questions related to programming exercise ?

[Reply](#)

Replies

Anonymous August 26, 2013 at 7:58 PM

One of the most tricky questions, I have face in a Java interview was, Does two object will always be equal, when there compareTo() method returns zero? I said, Yes, but that was not true. Though most of the classes will be equal if there compareTo() return true e.g. java.lang.String, but it's not mandatory. compareTo() may be inconsistent to equals(), which means compareTo() may return zero, but object will not be equal by equals() method. One of the prime example of this is java.math.BigDecimal class, whose equals() method return true if two BigDecimal object is equal in both value and scale e.g. 6.0 and 6.00 will not be equal, but compareTo() will return zero if both objects are compared. This was really tricky, until you had faced similar question previously. In another interview, my friend was asked this question little differently, Can we store BigDecimal class in TreeSet? obviously No, because of above reason, BigDecimal class can produce unexpected behavior when stored in SortedSet or SortedMap.



Alexander Vovolka September 12, 2016 at 6:46 AM

try here <http://quizful.com/quizzes/java>
as for me really tricky, I was really surprised with answers)



Emerikol December 15, 2016 at 1:05 PM

While I'm an certain these questions are getting asked, I find it ironic that being able to answer them in no way indicates any ability to be a successful programmer. That is why they are trick questions I guess.

[Reply](#)

Harish September 21, 2012 at 3:18 AM

Good work done! It will definitely help me in my interviews... :)
thanks .. :)

[Reply](#)



Summary September 27, 2012 at 2:20 AM

Hi ,

I am able to override public static method declared in base class in its subclass.
It didn't throw any compilation or runtime exception

[Reply](#)

Replies

Anonymous September 27, 2012 at 8:06 PM

It won't because compiler will treat it as different method. This is called method hiding and its indeed one of the tricky Java question. Remember that this method is not overriding super class method as static method bonded using type information. That's the reason this Java question is tricky.

[Reply](#)

Suvi October 12, 2012 at 12:56 AM

One of the tricky Java question I faced is "What is difference between Collection and Generics in Java", its not tricky because I don't know either collection or Generics but How can you compare Generics with Collection ?

[Reply](#)

[Stun](#)

[o](#)

[ire](#)

[set](#)

[\)](#)

**Ankur Kesar** October 29, 2012 at 4:36 AM

Hi

Is the line in codes correct.
It's one of ur question.

Can you access non static variable in static context?

Another tricky Java question from Java fundamentals. "No you can not access static variable in non static context in Java". Read why you can not access non-static variable from static method to learn more about this tricky Java questions.

[Reply](#)[Replies](#)**Javin Paul** October 29, 2012 at 6:05 AM

Hi Ankur, Thanks for pointing out, you should read opposite i.e non static variable can not accessible from static context. This always confuse if you don't remember it and that's why its one of the tricky question.

**shyam sunder** January 13, 2013 at 7:59 AM

Hi

This line is not correct.
"No you can not access static variable in non static context in Java".

Anonymous April 10, 2015 at 10:50 PM

you can not access a static variable inside the non-static one, other-way its so true.

[Reply](#)**rani** December 3, 2012 at 6:30 AM

all are tricky questions?

[Reply](#)**swapnil** December 6, 2012 at 3:32 AM

its really nice tricky question to read to face interview.

[Reply](#)**Anonymous** December 17, 2012 at 3:41 AM

these are not tricky rather easy questions ...

[Reply](#)**Anonymous** December 18, 2012 at 11:11 AM

i agree nice question to prepare before appearing for interview

[Reply](#)**Tafadzwa Kombe** January 5, 2013 at 10:17 AM

These are definitely a good set of tricky questions that a candidate may face during an interview. I certainly enjoyed going through them but I have a different opinion regarding the last question: Can you access non static variable in static context?

I think the answer to that question is a YES albeit one should note that you need an object reference of the associated class to access the variable (I assume this variable is an instance variable). And likewise you can directly access a static variable in non static context too. Please refer to this link for more information <http://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html>

[Reply](#)**Anonymous** February 11, 2013 at 8:36 AM

Is the compareTo code correct ? the references look to be wrong.

[Reply](#)**Ankita Dutta** February 20, 2013 at 12:51 AM

The explanation of the last question is little bit confusing. Actually static variables are always accessible in non static context however the reverse is not true.

[Reply](#)[Replies](#)**Anonymous** September 23, 2013 at 8:21 AM

In static context, you can access non-static variables - by creating a new instance of the object.
Hence the reverse is partly true.

[Reply](#)**Anonymous** February 23, 2013 at 1:05 PM

```
public int compareTo(Object o){
    Employee emp = (Employee) emp;
    return this.id - o.id;
}
```

I think this is what it should be...

```
public int compareTo(Object o){
    Employee emp = (Employee) o;
```

[Stun](#)[Stun](#)

```
return this.id - o.id;
}
```

[Reply](#)[Replies](#)**Patryk Szalanski** March 27, 2013 at 11:57 AM

Wrong, it should be:

```
public int compareTo(Object o){
    Employee emp = (Employee) o;
    return this.id - emp.id;
}
```

Anonymous February 17, 2014 at 3:31 AM

I should also mention that you shouldn't cast to Employee without knowing it's an Employee via instanceof, or just using Comparable generic interface.

[Reply](#)**saidi reddy** February 28, 2013 at 10:22 PM

Try some more new

[Reply](#)**Anonymous** March 12, 2013 at 12:43 AM

Q 1. about try and catch block

```
class Main
{
    public static void main(String args[])
    {

        try
        {
            int a=2/0;
            System.exit(0);
        }

        catch(Exception e)
        {

            System.out.println("i am in catch block");

        }

        finally
        {

            System.out.println("finally");
        }

    }
}
/*OUTPUT
i am in catch block
finally*/
```

finally block get executed if we place System.exit(0) in try block

[Reply](#)[Replies](#)**Anonymous** April 4, 2013 at 3:40 AM

Before calling System.exit(0) you raised exception "int a=2/0".
 Once system.exit(0) is called finally will never called.
 plz Don't confused others :)

Anonymous May 29, 2013 at 1:48 AM

2/0 is arithmetic exception so before going to system.exit it will jump to catch block and then finally but once system.exit(0) executes ...finally will not execute then .. try it with

```
class Main
{
    public static void main(String args[])
    {

        try
        {
            int a=2/1;
            System.exit(0);
        }

        catch(Exception e)
        {

            System.out.println("i am in catch block");

        }

        finally
        {

            System.out.println("finally");
        }

    }
}
```

[Stun](#)

```

[
]

```

```

ire
[
]
set
'
)

```

```

    }
}

```

Anonymous July 18, 2013 at 4:29 AM

as per my java knowledge,try,finally blocks will execute parallelly,because if single statement(inside try block first line or empty block) is executed corresponding catch may or may not execute but finally block will execute automatically,otherwise it wont execute,so without try&catch blocks we cant write finally block in java.

**venky dumpa** May 13, 2016 at 5:23 AM

System.exit(0) is unreachable statement here.

Reply**shuvam p** March 18, 2013 at 12:11 AM

I was asked during an interview the difference between "arraylist" and "linkedlist" and when I should use them

Also, found a site where you can give online Java mock interviews. Its www.prepastreet.com

Reply**Replies****Anonymous** October 22, 2015 at 3:55 AM

linklist is dynamic is called at runtime to save memory. where as arrylist is not dynamic whose memory locations are allocated in memory so memory is already stored for that so memory is waste. if u have shortage of memory can use linklist it is envoked at runtime and if u have enough memory can use arraylist..

**Alexey Zavyalov** November 23, 2015 at 2:01 AM

Wrong. LinkedList uses more memory, because, apart from values, it stores a reference to the previous and next entry. The difference between them in the ways of access to the elements of the list, and add / remove items in the list

Reply**Fawad** August 14, 2013 at 7:22 PM

What is so tricky about these question, to me they look most simplest question, which doesn't even qualify for Interviews. Tricky questions are those, who challenge your notion e.g.

When Singleton doesn't remain Singleton in Java?

is it possible to load a class by two ClassLoader?

is it possible for equals() to return false, even if contents of two Objects are same?

Why compareTo() should be consistent to equals() method in Java?

is Following code legal in Java? is it example of method overloading or overriding?

```

public String getDescription(Object obj){
    return obj.toString();
}

```

```

public String getDescription(String obj){
    return obj;
}

```

and

```

public void getDescription(String obj){
    return obj;
}

```

Anyone disagree with my questions not being tricky?

Reply**Replies****Anonymous** February 16, 2014 at 6:20 AM

Actually your questions are better than Javin's Questions...

**Peter Lawrey** May 5, 2014 at 9:17 AM

Trickier) When do Double and BigDecimal give different answers for equals() and compareTo() == 0.

Reply**Java Online** August 29, 2013 at 2:05 AM

Hi,

Good collection of tricky questions, However I feel that the questions in the interview becomes tricky because we need to answer the same as what interviewer in thinking is correct, this is the most tricky part. It is always better to clarify the question correctly by rephrasing it.

Reply**Peter Lawrey** May 5, 2014 at 9:13 AM

Some trickier questions.

1a) how do you prevent a return statement from calling finally.
1b) how do you make a System.exit() call the finally block.

3) Not sure multiple inheritance is the trickiest question in Java. How about;
2a) how does "break before" analysis relate to code?

Stun

```

    }
}

```

ire

set

}

3a) how does "nas before" apply to volatile work?

3b) why is $0.1 * 3 != 0.3$,

3c) why is `(Integer) 1 == (Integer) 1` but `(Integer) 222 != (Integer) 222` and which command arguments change this.

7) how can you release synchronized locks in an order which is not the reverse order? (You can do this BTW)

Good answers to some common questions, I look forward to some trickier questions. :)

[Reply](#)

Replies

Fawad June 27, 2014 at 10:29 PM

I would add :

What happens when exception is thrown in a Thread?

Difference between `notify` and `notifyAll` call?

Difference between `System.exit()` and `System.halt()` method?

do you agree these are much trickier question for an average Java programmers?



Unknown October 1, 2015 at 9:33 PM

Is there a `System.halt()` method exists in Java? I think it is `Runtime.getRuntime().halt(status)`, right?

[Reply](#)

Tom Henriksen October 6, 2014 at 5:32 AM

Thanks for sharing these. I need some good Java interview questions and a few of these might help me out.

[Reply](#)

Replies



Javin Paul October 6, 2014 at 5:35 AM

No problem Tom, thanks for dropping by. If you have not read already, you may find my article about [10 Questions to make Programming Interview Less Costly](#) useful as well.

[Reply](#)

Anonymous December 10, 2014 at 1:57 AM

Answer to the question "Is it possible to load a class by two ClassLoader?" is Yes, it quite possible if you are using a custom class loader or working on managed environment which uses classloader e.g. web and application server. This is one more reason why you should use `instanceof` instead of `getClass()` while overriding `equals()` method, otherwise `equals()` will return false even if object are same but classloader is different.

[Reply](#)



Hari Krishna December 24, 2014 at 3:25 AM

hi..

this is hari.can you any one send me jsp&servlets 1.6 year interview questions.

this is my id:harikrishnapulpati@gmail.com
please help me...

[Reply](#)

Anonymous January 9, 2015 at 8:59 PM

How would you describe `Enum<E> extends Enum<E> > ?`

[Reply](#)

Anonymous January 19, 2015 at 5:21 AM

About Q2: What will happen if you put `return` statement or `System.exit()` on `try` or `catch` block ? Will finally block execute?

Here is a code fragment (Java class) I found a long time ago on some C++ forum. It is called "JavaSucks" (sorry) and its content says it all:

```
public class JavaSucks {
    public static void main(String[] args) {
        for (;;) {
            try {
                System.out.println("Java sucks");
            } catch (Exception e) {
                System.exit(0);
            } finally {
                continue;
            }
        }
    }
}
```

It will compile without errors? If yes, what do you think it will produce on console as output?

Compile it and run it. Result: no compilation errors, just one warning. It will print infinitely the string "JavaSucks"...

[Reply](#)

Anonymous September 30, 2015 at 12:45 AM

Pass by value and pass by reference is also a good tricky Java question. Btw, Java is always pass by value, even for objects its pass by value, its just that reference or handle to the object is passed.

[Reply](#)

**Sandeep Yadav** October 12, 2015 at 11:23 AM

can we declare constructor as private??? if yes then for which purpose??

[Reply](#)**Javin Paul** October 13, 2015 at 5:50 AM

Hi Sandeep, Yes, you can declare a private constructor in Java. You can do to prevent instantiation of class outside the class, for example, Singleton pattern is one of the prime examples of the private constructor. In Singleton, the class itself is responsible for creating the instance and managing it so constructor is made private.

[Reply](#)**Anonymous** October 16, 2015 at 10:15 AM

A fiend who is a really good programmer was given some of these in an interview a few months ago.

His response was: "I'd fire who every wrote such lousy code. I never have to worry about such things, because I write defensive and don't allow such nonsense in my code base"

He didn't get the job, but he's right. Most of this "trick" questions are just academic and should never occur to in real life. It's lousy people think these type of questions some how gauge the productivity and ability of a programmer who avoids such pitfalls to begin with and hence doesn't know the answers to the way out of them.

[Reply](#)**Anonymous** October 19, 2015 at 8:56 PM

Some questions are really tricky especially "Why 01*3 != 0.3 in Java?". I doubt even experienced Java developers can answer with clarity and confidence. I have asked this question to my friends having 5 and 6 years of experience and my technical lead having 10 years of experience Java, but they couldn't provide me good answer. All they could say is, since some floating point numbers cannot be represented precisely in Java, hence 0.1*0.3 != 0.3

[Reply](#)**Anonymous** December 3, 2015 at 6:54 PM

I think in practice questions "How does "has before" apply to volatile work?", you mean how "happens before" concept work with volatile variables? right? as far as I know there are certain rules which decides which update will happen first. I remember reading about it on Java concurrency in Practice, which says that a volatile write will happen before volatile read.

[Reply](#)**Javin** December 5, 2015 at 8:06 AM

I have recently shared few more Java interview questions especially for developer with 1 to 4 years of experience, you can see it [here](#)

[Reply](#)**Modern Pathshala** December 17, 2015 at 8:38 AM

Is null key allowed to stored in HashMap? If yes, how it is handled?

What will happen if two different HashMap key objects have same hashCode?

Visit <http://modernpathshala.com/Learn/Java/Interview> for more java interview questions and answers.

[Reply](#)**niazi** February 7, 2016 at 8:38 AM

This blog awesome and i learn a lot about programming from here. The best thing about this blog is that you doing from beginning to experts level.

Love from

[Reply](#)**Anonymous** May 18, 2016 at 10:07 PM

```
public class Test {
    public static void main(String[] args) {
        System.out.println(Math.min(Double.MIN_VALUE, 0.0d));
    }
}
```

These type of questions will get you people in the industry who have crammed the language and don't know the design practices. No way to gauge a good programmer from a bad one

[Reply](#)**Anonymous** June 8, 2016 at 9:01 PM

What would the following program will print
public class Main {

```
    public static void main(String[] args) {
        Collection c = new HashSet();
        print(c);
    }
```

```
    public static void print(Collection c){
        System.out.println("Collection");
    }
```

```
    public static void print(Set s){
        System.out.println("Set");
    }
```

```
    public static void print(HashSet hs){
        System.out.println("HashSet");
    }
```

[Siun](#)

```

    }
}
```


```

ire |
set
}
```


}
I said "HashSet" but it was wrong.

Reply


Replies

 **Unknown** June 28, 2016 at 4:02 AM
Overriding static methods is called as method hiding. In method hiding method resolution is always based on reference type so it will print Collection.

Reply

 **Javin Paul** June 9, 2016 at 5:09 AM
It will print "Collection" because methods are static and so they are bonded during compile time and at that time only type information is available because object is created at runtime.


Reply

 **Jake Beasley** July 5, 2016 at 10:58 AM
Honestly, these questions are poor indicators of how effective they will be on the job. They may be entertaining, but they are really bad questions when screening job candidates for real work.


On the job you want people who are creative, capable of exploring available options, and capable of executing well on the tasks that they are assigned to them, large or small. These sort of "gotcha" questions do not actually tell you if they are good developers - just whether they have "happened" to stumbled upon obscure facts or details in their careers or have spend an inordinate amount of time reading java docs.

Reply

Replies


 **Javin Paul** July 26, 2016 at 5:52 PM
Hello @Jake, I don't believe the go/no-go decision is based upon these tricky questions. They are mainly used to add the surprise element or check the depth of candidates knowledge in any particular topic. It's a good indicator of candidate's deep and thorough understanding but as you said, don't expect every Java developer knows these subtle details.

Reply

 **Jang Bahadur** July 22, 2016 at 7:57 PM
Wow Really Nice Collections of Interview Questions . Thank you :)

Reply

Replies

 **Javin Paul** July 26, 2016 at 5:50 PM
Hello @Jang Bahadur, thanks you like these tricky Java questions, If you have any, please share with us as well :-)

Reply

Anonymous August 16, 2016 at 10:03 PM
One of these questions, [CyclicBarrier vs CountdownLatch](#) are asked to me on screening round, I managed to answer it well, thanks to internet and you guys.

Reply

Anonymous October 12, 2016 at 9:26 AM
web services consume or produce the web services if consume explain how

Reply

Enter your comment...

Comment as: amitabh sumar

Sign out

Publish

Preview

☐ Notify me

[Newer Post](#) [Home](#) [Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)

Stun

ire

set