Read more: <http://www.java67.com/2012/09/top-10-tricky-java-interview-questions-answers.html#ixzz4GjNWMgXh>

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](http://java67.blogspot.sg/2012/12/difference-between-runtimeexception-and-checked-exception.html) 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](http://java67.blogspot.sg/2012/09/top-10-tough-core-java-interview-questions-answers.html) 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](http://javarevisited.blogspot.com/2015/12/5-good-books-for-java-jee-programming.html), 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](http://www.java67.com/2016/06/12-must-read-advance-java-books-for-intermediate-programmers.html) 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](http://java67.blogspot.sg/2013/03/how-to-convert-java-string-to-int-or.html), 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](http://java67.blogspot.com/2016/06/difference-between-final-vs-finally-vs-finalize-in-java.html) 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**](http://java67.blogspot.sg/2012/08/can-we-override-static-method-in-java.html)**,** 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](http://java67.blogspot.sg/2012/08/can-we-override-private-method-in-java.html) 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 involving them. The simple answer to this question is that **it will not throwArithmeticExcpetion 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](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20) 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 than why not Java is the argument Interviewer often give. Answer of this question is much more subtle then it looks like, because J**ava 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](http://javarevisited.blogspot.sg/2011/07/why-multiple-inheritances-are-not.html) 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](http://java67.blogspot.sg/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html) 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, t[o convert a string to a byte array](http://javarevisited.blogspot.sg/2014/08/2-examples-to-convert-byte-array-to-String-in-Java.html), 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 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](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html) 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, three 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](http://javarevisited.blogspot.sg/2011/11/how-to-override-compareto-method-in.html) 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](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html) 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](http://java67.blogspot.sg/2012/08/difference-between-countdownlatch-and-cyclicbarrier-java.html) for more differences.

**What is the difference between StringBuffer and StringBuilder in Java?**

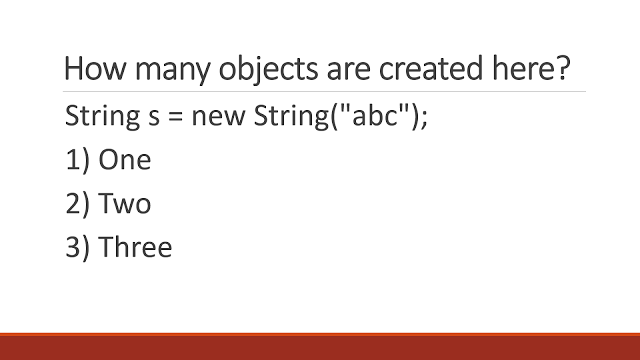
Classic Java questions which some people think tricky and some consider very easy. StringBuilder in Java was introduced in JDK 1.5 and the only difference between both of them is that StringBuffer methods e.g. length(), capacity() or append() are [synchronized](http://javarevisited.blogspot.sg/2011/04/synchronization-in-java-synchronized.html) 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 99% scenario, you perform string concatenation on the same thread. See [StringBuilder vs StringBuffer](http://javarevisited.blogspot.sg/2011/07/string-vs-stringbuffer-vs-stringbuilder.html) 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](http://javarevisited.blogspot.sg/2012/02/why-non-static-variable-cannot-be.html) to learn more about this tricky Java questions.

**How many String objects are created by the following code?**



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 != 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?
13. 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.

Read more: <http://www.java67.com/2012/09/top-10-tricky-java-interview-questions-answers.html#ixzz4GjNi9beS>

I’ll try to compile here a list of maybe the most / best java interview questions which are asked at java interviews, differentiated by the level of the candidate applying:

**1. In general first questions should be regarding basic java knowledge** (syntax, types,most used methods, synchronization, garbage collection, collections, simple design paterns, jdbc, SQL – maybe).

**2. The next pack of questions should test the higher abilities of the developer**, targeting more complicated languge related functionality(trick questions which imply a good language understanding, some exotic keywords like volatile, some exotic methods like intern, some more complex design patterns, difference between last versions of java, virtual machine implementations etc, rmi, idl, corba)

**3. The 3rd pack sould contain questions about web, distributed and database development** (jsp, servlets, struts, jsf, spring, hibernate, orm, differences between them – where is the case, design patterns, projects involved, soa, web services)

**4. I add in the last pack general questions** (like what IDEs have you used, what application servers have you used, comparison between them – where is the case, best and worst projects you’ve been into and why, motivation for changing the job for the present one etc)

\*\*UPDATE\*\*: If you are a junior java developer or an aspiring one, I strongly recommend you to take first the OCJP certification before going to an interview. This can increase your chances to succes big time, as well as your entry salary – proven and tested by myself! :)

1) Junior java developer

a) Basic ocjp (former scjp) questions:

– What does static, final mean, purposes;

– How many accesibility modifiers exist? Please describe them.

– Why do you need a main method?

– How many constructors can you have?

– Define overwriting and overloading

– Give java API implementations for overwriting and overloading

– Describe the String class – unique properties

– StringBuilder vs StringBuffer

– Collections : please describe, give some examples and compare them to eachother

– ArrayList vs Vector

– HashMap vs HashTable

– What’s a tree

– What’s a map

– Multithreading: describe the management in java

– What’s a semaphone?

– How many states are there for threads?

– Describe the usage for synchronized word (2)

– Serialization in java – a descrition and usage

– Garbage collection in java – description and usage

– Can you guarantee the garbage collection process?

b) Simple design pattern questions:

– Singleton please describe main features and coding

– Factory please describe main features and coding

– Have you used others? please describe them

2) Intermediate and Senior level – depending on rate of good responses, additional questions to 1):

a) Some other scjp:

– Describe transient

– Which other fields besies transient are not persisted?

– Describe volatile

b) Some logical assesments

– How can you isolate some variables in a multithreading environment without synchronized?

– Can you extend a class which is implementing the Singleton design pattern?

– Describe the Singleton pattern for a eager/laizy impelmentation and describe differences, pluses and minuses

– Give a thread safe version of Singleton

– Can you have more instances per one Singleton in a jvm?

– Diference between stateless and statefull – in general (not ejb-related)

– Difference between synchronous and asynchronous call in general (not ejb related)

c) Design patterns – additional questions

– Adaptor vs Wrapper design pattern

– Factory method design pattern

– Strategy design pattern

– Observer design pattern

d) Web development and enterprise knowledge

– D/HTML general questions

– What’s an EJB – how many typers are there

– How many types of JDBC drivers exist

– Describe JAAS

– Describe JAXB

– Descrive JAXP – difference to JAXB

– Describe Java Security in general

– What’s a webservice, how many ways of implementing it do you know?

– Describe SOA

– Why should you use SOA

– Descibe CORBA

– Describe SOAP

– Describe RMI

– What’s a stub? And a skeleton? Differences between them?

– What’s a Servlet?

– What’s a jsp?

– Difference between jsp and servlet

– Describe JSF

– What’s ORM

– Describe JPA

e) 3rd party frameworks

– What 3rd party frameworks do you know?

– Describe Struts 1

– Describe Struts 2

– Differences between Struts1, Struts 2 and JSF

– Describe Hibernate

– Describe Spring

f) others

– Tell me about teamwork

– What IDE’s have you used? Please describe and compare.

– What Applicaion Servers have you used? Please describe and compare.

– What companies have you worked for before? On which projects

– Please tell me the best project you were working at

– Please tell me the worst project you were woring at

– Please tell me why do you want to be a member of this team

You can find more questions and also the answers to these ones(in progress) at<http://centraladvisor.com/tag/interview>

and/or at <http://centraladvisor.com/tag/java>