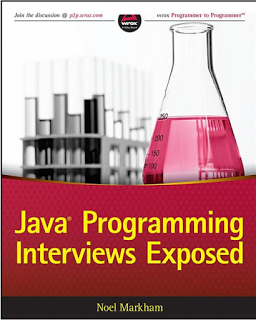
15 Technical Core Java Interview Questions for Experienced Developers

When the experience of a Java Programmer grows in the years e.g. when it goes from beginner years ( 2 to 4) to more experience or sort of senior level ( 5 to 7 years), [*Core Java Interview Questions*](http://java67.blogspot.com/2012/09/top-10-tough-core-java-interview-questions-answers.html) also changes a bit. Of course, basics like **data structure**, **algorithms,**and **object-oriented programming** remains same, but types of questions will become more advanced and their answers will definitely need to be more detailed and accurate. I often receive queries about core Java questions asked to a senior developer of 5 to 6-year experience, or, sometimes, I am going for an interview of senior Java developer, what kind of questions I should expect. This sometimes puzzles me, that once you become senior, you automatically starts taking part in the interview, and you should have an idea of what to expect on Interviews, but at the same time, I can understand that having an idea of questions before going on Interview, helps preparation. Of course, you are not going to get a question like the one you have faced on [2 to 3 years level Java Interviews](http://java67.blogspot.sg/2012/10/java-interview-questions-for-2-to-3-4-years-experienced.html), but It also depends upon different rounds of Interviews.   
  
I have not seen many changes in the questions asked on telephonic round, which almost remains same. You will find some [fact based](http://java67.blogspot.com/2014/07/21-frequently-asked-java-interview-questions-answers.html), some [coding questions](http://java67.blogspot.com/2012/08/10-java-coding-interview-questions-and.html) and few [tricky questions](http://java67.blogspot.com/2012/09/top-10-tricky-java-interview-questions-answers.html). On the other hand face-to-face, interviews have become more detailed and more tricky, especially with nasty follow-ups.

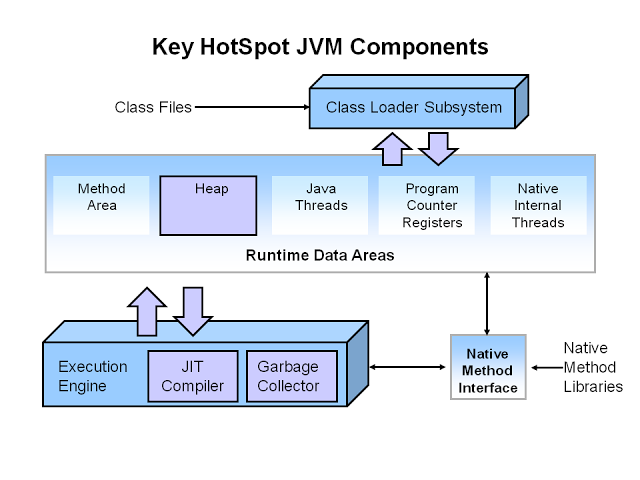
In this article, I am going to share some **15 technical core Java Interview Questions**, which I have seen asked senior and experienced developers of 4 to 6 years of experience in different interviews, mostly on [telephonic rounds](http://java67.blogspot.com/2015/03/top-40-core-java-interview-questions-answers-telephonic-round.html).  I am not posting answers as of now, but  you can find answers to most of the questions on here or Javarevisited blog.  
  
  
If you are in a hurry and actively looking for a Java Developer Job, you can also take help from some good books written to prepare you for Java J2EE interviews. Books like, [Java Programming Interview Exposed](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20) covers all important topics for both core Java and Java EE interviews, which includes basic Java questions, data structure and algorithms, JVM internals and GC tuning, Hibernate and Spring interview questions, JUnit ant unit testing questions, and some Java 8 stuff. It also covers knowledge of other JVM languages like Scala, Groovy and other platforms like Android. A perfect companion to do well in Java interviews.

[](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20)

If you want to prepare more, you can further check out [Cracking the Coding Interview](http://www.amazon.com/dp/098478280X/?tag=javamysqlanta-20), which contains almost 150 programming questions and solutions from technical interviews of big tech companies like Amazon, Facebook, Google, Twitter, and Microsoft.

**15 Core Java Questions For 5 to 6 Years Experienced**

[Java questions for 5 to 6 years experienced and senior programmers](http://1.bp.blogspot.com/-_GCqP1vu06Q/UBaTOZM869I/AAAAAAAAAaw/ykubu9U9kK4/s1600/java_logo_50_50.jpg)All these questions have been collected from quite senior developers, which has at least 5 years of experience. They have seen these questions on different rounds of their core Java interviews, including telephonic and face-to-face rounds on different companies, mostly on Investment banks like Barclays, Morgan, RBS and others.  
  
  
**1. What is Busy Spinning? Why Should You Use It in Java?**  
One of the interesting multithreading question to senior Java programmers, busy spinning is a *waiting strategy*, in which a thread just wait in a loop, without releasing the CPU for going to [sleep](http://java67.blogspot.sg/2012/08/what-are-difference-between-wait-and.html). This is a very advanced and specialized waiting strategy used in the high-frequency trading application when wait time between two messages is very minimal.  
  
By not releasing the CPU or [suspending the thread](http://java67.blogspot.com/2015/06/how-to-pause-thread-in-java-using-sleep.html), your thread retains all the cached data and instruction, which may be lost if the thread was suspended and resumed back in a different core of CPU.   
  
This question is quite popular in high-frequency low latency programming domain, where programmers are trying for extremely low latency in the range of micro to milliseconds. See here more [50+ advanced thread interview questions](http://javarevisited.blogspot.com/2014/07/top-50-java-multithreading-interview-questions-answers.html) for experienced programmers.   
  
  
  
**2. What is Read-Write Lock? Does ConcurrentHashMap in Java Uses The ReadWrite Lock?**  
ReadWrite Lock is an implementation of *lock stripping* technique, where two separate locks are used for read and write operation. Since read operation doesn't modify the state of the object, it's safe to allow multiple thread access to a shared object for reading without locking, and by splitting one lock into [read and write lock](http://javarevisited.blogspot.com/2014/10/how-to-use-locks-in-multi-threaded-java-program-example.html), you can easily do that.   
  
Java provides an implementation of read-write lock in the form of ReentrantReadWritLock class in the java.util.concurrent.lock package. This is worth looking before you decide to write your own read-write locking implementation.   
  
Also, the current implementation of java.util.ConcurrentHashMap doesn't use the ReadWriteLock, instead, it divides the Map into several segments and lock them separately using different locks. This means any given time, *only a portion of the ConcurrentHashMap is locked*, instead of the whole Map. See [how ConcurrentHashMap internally works in Java](http://javarevisited.blogspot.com/2013/02/concurrenthashmap-in-java-example-tutorial-working.html) for more detail.   
  
  
This core Java question is also very popular on senior and more experienced level Java interviews e.g. 4 to 6 years, where you expect Interviewer to go into more detail, e.g. by asking you to provided an implementation of the read-write lock with different policies. If you are an experienced Java programmer, consider reading [Java Concurrency in Practice](http://www.amazon.com/dp/0321349601/?tag=javamysqlanta-20) to gain more confidence about multithreading and concurrency in Java.   
  
  
  
**3. How to Make an Object Immutable in Java? Why Should You Make an Object Immutable?**  
Well, Immutability offers several advantage including thread-safety, ability to cache and result in more readable multithreading code. See [here](http://javarevisited.blogspot.sg/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html)to learn how to make object Immutable. Once again, this question can also go into more detail and depending on your answer, can bring several other questions e.g. when you mention Spring is Immutable, be ready with some reasons on [Why String is Immutable in Java](http://javarevisited.blogspot.sg/2010/10/why-string-is-immutable-in-java.html).  
  
  
**4. Which Design Patterns have You Used in Your Java Project?**  
Always expect some design patterns related question for Core Java Interview of senior developer position. It's a better strategy to mention any GOF design pattern rather than Singleton or MVC, which almost every other Java developer use it.   
  
Your best bet can be [Decorator pattern](http://java67.blogspot.sg/2013/07/decorator-design-pattern-in-java-real-life-example-tutorial.html) or may be [Dependency Injection Pattern](http://javarevisited.blogspot.sg/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html), which is quite popular in Spring Framework. It's also good to mention only the design patterns which you have *actually* used in your project and knows it's tradeoffs.   
  
It's common that once you mention a particular design pattern say Factory or Abstract Factory, Interviewer's next question would be, *have you used this pattern in your project?* So be ready with proper example and why you choose a particular pattern. You can also see [this](http://java67.blogspot.com/2012/09/top-10-java-design-pattern-interview-question-answer.html) article for more advanced design pattern questions from Java interviews.   
  
  
  
**5.  Do you know about Open Closed Design Principle or Liskov Substitution Principle?**  
Design patterns are based on [object-oriented design principles](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html), which I strongly felt every object-oriented developer and the programmer should know, or, at least, have a basic idea of what are these principles and how they help you to write better object oriented code. I  
  
f you don't know the answer to this question, you can politely say No, as it's not expected from you to know the answer to every question, but by answering this question, you can make your claim stronger as many experienced developers fail to answer basic questions like this. See [Clean Code](http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882?tag=javamysqlanta-20) to learn more about object-oriented and SOLID design principles.  
 **6. Which Design Pattern Will You Use to Shield Your Code From a Third Party library Which Will Likely to be Replaced by Another in Couple of Months?**  
This is just one example of  the scenario-based design pattern interview question. In order to test the practical experience of Java developers with more than 5 years experience, companies ask this kind of questions.  You can expect more real-world design problems in different formats, some with more detail explanation with context, or some with only intent around.   
  
One way to shield your code from third party library is to [code against an interface rather than implementation](http://javarevisited.blogspot.com/2014/11/why-use-interface-in-java-or-object-oriented-programming.html) and then use dependency injection to provide a particular implementation. This kind of questions is also asked quite frequently to experienced and senior *Java developers with 5 to 7 years of experience*.  
 **Question 7)  How  do you prevent SQL Injection in Java Code?**  
This question is more asked to J2EE and Java EE developers than core Java developers, but, it is still a good question to check the JDBC and Security skill of experienced Java programmers.  
  
You can [use PreparedStatement to avoid SQL injection](http://javarevisited.blogspot.com/2012/03/why-use-preparedstatement-in-java-jdbc.html) in Java code. Use of the PreparedStatement for executing SQL queries not only provides better performance but also shield your Java and J2EE application from SQL Injection attack.   
  
On a similar note, If you are working more on Java EE or J2EE side, then you should also be familiar with other security issues including *Session Fixation attack* or *Cross Site Scripting* attack and how to resolve them. These are some fields and questions where a good answer can make a lot of difference on your selection.   
 **Question 8) Tell me about different Reference types available in Java, e.g. WeakReference, SoftReference or PhantomReference? and Why should you use them?**  
Well, they are different reference types coming from java.lang.ref package and provided to assist Java Garbage Collector in a case of low memory issues. If you wrap an object with [WeakReference](http://javarevisited.blogspot.com/2014/03/difference-between-weakreference-vs-softreference-phantom-strong-reference-java.html) than it will be eligible for garbage collected if there are o strong reference. They can later be reclaimed by Garbage collector if JVM is running low on memory.  
  
The java.util.WeakHashMap is a special Map implementation, whose keys are the object of WeakReference, so if only Map contains the reference of any object and no other, those object can be garbage collected if GC needs memory. See [Java Performance The Definitive Guide](http://www.amazon.com/Java-Performance-The-Definitive-Guide/dp/1449358454?tag=javamysqlanta-20) to learn more about how to deal with performance issues in Java. 

[](http://java67.blogspot.com/2013/02/difference-between-jit-and-jvm-in-java.html)

**Question 9) How does get method of HashMap works in Java?**  
Yes, this is still one of the most popular core Java questions for senior developer interviews. You can also expect this question on telephonic round, followed by lot's of follow-up questions as discussed on my post [how does HashMap work in Java](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html).   
  
The short answer to this question is that HashMap is based upon hash table data structure and uses hashCode() method to calculate hash code to find the bucket location on underlying array and equals() method to search the object in the same bucket in case of a collision. See [here](http://java67.blogspot.com/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html) to learn a more about how does get() method of HashMap works in Java.   
  
  
  
**Question 10) Which Two Methods HashMap key Object Should Implement?**  
This is one of the follow-up questions I was saying about in previous questions. Since working of HashMap is based upon hash table data structure, any object which you want to use as key for HashMap or any other hash based collection e.g. Hashtable, or ConcurrentHashMap must implement [equals()](http://java67.blogspot.com/2012/11/difference-between-operator-and-equals-method-in.html) and [hashCode()](http://java67.blogspot.com/2013/04/example-of-overriding-equals-hashcode-compareTo-java-method.html) method.   
  
The hashCode() is used to find the bucket location i.e. index of underlying array and equals() method is used to find the right object in linked list stored in the bucket in case of a collision. By the way, from Java 8, HashMap also started using tree data structure to store the object in case of collision to reduce worst case performance of HashMap from O(n) to O(logN). See the article for learning more about [how does HashMap handless collision in Java](http://javarevisited.blogspot.com/2016/01/how-does-java-hashmap-or-linkedhahsmap-handles.html).   
  
  
  
**Question 11) Why Should an Object Used As the Key should be Immutable?**   
This is another follow-up of previous core Java interview questions. It's good to test the depth of technical knowledge of candidate by asking more and more question on the same topic. If you know about Immutability, you can answer this question by yourself. The short answer to this question is key should be immutable so that hashCode()method  always return the same value.

Since hash code returned by hashCode() method depends upon the content of object i.e. values of member variables. If an object is [mutable](http://java67.blogspot.com/2014/01/why-string-class-has-made-immutable-or-final-java.html) than those values can change and so is the hash code. If the same object returns different hash code once you inserted the value in HashMap, you will end up searching in different bucket location and will not able to retrieve the object. That's why a key object should be immutable. It's not a rule enforced by the compiler but you should take care of it as an experienced programmer. See the article for more [advanced Java Collection interview questions](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html).   
  
  
  
**Question 12) How does ConcurrentHashMap achieves its Scalability?**   
Sometimes this multithreading + collection interview question is also asked as, the difference between ConcurrentHashMap and Hashtable in Java. The problem with [synchronized HashMap](http://java67.blogspot.com/2015/02/how-to-synchronize-hashmap-in-java-with.html) or [Hashtable](http://java67.blogspot.com/2012/08/difference-between-hashmap-and-concurrentHashMap-java-collection.html) was that whole Map is locked when a thread performs any operation with Map.   
  
The java.util.ConcurrentHashMap class solves this problem by using *lock stripping* technique, where the whole map is locked at different segments and only a particular segment is locked during the write operation, not the whole map. The ConcurrentHashMap also achieves it's scalability by allowing lock-free reads as read is a thread-safe operation.  See [here](http://java67.blogspot.com/2012/08/5-thread-interview-questions-answers-in.html) for more advanced multi-threading and concurrency questions in Java.   
  
  
  
**Question 13) How do you share an object between threads? or How to pass an object from one thread to another?**  
There are multiple ways to do that e.g. Queues, Exchanger etc, but BlockingQueue using [Producer Consumer pattern](http://java67.blogspot.sg/2012/12/producer-consumer-problem-with-wait-and-notify-example.html) is the easiest way to pass an object from thread to another.  
  
  
  
**Question 14) How do find if your program has a deadlock?**  
By taking thread dump using kill -3, using JConsole or VisualVM), I suggest to prepare this core java interview question in more detail, as Interviewer definitely likes to go with more detail e.g. they will press with questions like, have you really done that in your project or not?  
  
  
**Question 15) How do you avoid deadlock while coding?**  
By ensuring locks are acquire and released in an ordered manner, see [here](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html)for detail answer of this question.  
  
  
That's all on this list of *core Java Interview Questions for senior developers and experienced programmers*. I haven't included a lot of question from another important topics like Exception handling, Garbage Collection tuning and JVM Internals, which is also very popular among Java programmers with 5 to 6 years of experience, maybe I will include them in the [next part](http://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html).    
  
By the way, if you don't find the answer of any of these core Java Question, let me know. I may update the post with a more detailed option, based on my readers request.

Read more: <http://www.java67.com/2013/07/15-advanced-core-java-interview-questions-answers-senior-experienced-5-6-years-programmers-developers.html#ixzz4DQSmIOd1>