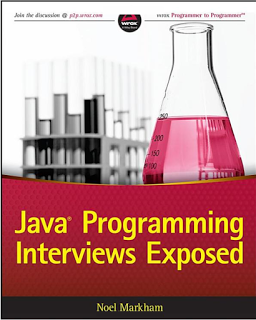
Top 12 Java Thread, Concurrency and Multithreading Interview Questions For experienced

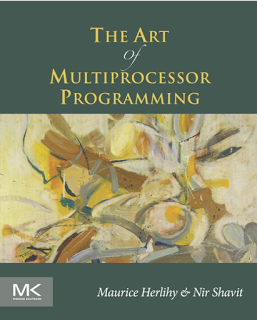
**Java concurrency and thread interview questions answers**  
Multithreading is an important feature of Java programming language, which means threads are also an important part of any Java interview. It's true and in fact at beginners and freshers level [Thread interview questions in Java](http://javarevisited.blogspot.sg/2011/07/java-multi-threading-interview.html) are one of most difficult to answer. One reason for interview question related to multithreading and concurrency being difficult is confusion around how multiple threads works together and second is threads are genuinely a complicated topic to understand and use correctly. Mostly *thread interview questions* checks Java programmers knowledge on Java Thread API, Java concurrency API, issues related to multi-threading like a [race condition](http://javarevisited.blogspot.com/2012/02/what-is-race-condition-in.html), [thread-safety](http://javarevisited.blogspot.sg/2012/01/how-to-write-thread-safe-code-in-java.html), and [deadlock](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html). Some time multithreading and concurrency interview question also focus on parallel design patterns like solving the producer-consumer problem, implementing work steal pattern or solving dining philosopher problem in Java. This is especially true while interviewing experienced Java developers with 4 to 6 years of experience.  
  
In this article, we will take a look at different kinds of **multithreading and concurrency questions asked in various interviews**e.g. on telephonic or face to face interview, on written test, to both experienced and senior Java developers, and some tips to answer them correctly.  
  
Questions asked on [telephonic](http://java67.blogspot.com/2015/03/top-40-core-java-interview-questions-answers-telephonic-round.html) or the [first round of interviews](http://java67.blogspot.com/2014/07/21-frequently-asked-java-interview-questions-answers.html) are tend to be easier and you must answer them to the point with a keyword, which interviewer is expecting. On the face to face interview, be prepare for different kinds of follow-up questions. Books like [Java Programming Interview exposed](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20) also helps you to prepare better and answer to the point.

[](https://2.bp.blogspot.com/-YHIEhiMimYw/V109tQb0kDI/AAAAAAAAGNM/m9HCXnUq3m4ELhLp2v-Z5pn35z31rCsygCLcB/s1600/Java%2BProgramming%2BInterview%2BExposed.png)

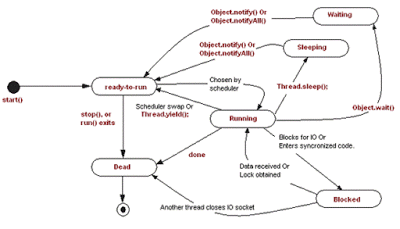
**Java Interview questions on Concurrency and multithreading**

As I said, in this Java article, not only,  I will share some of the most commonly asked thread interview questions at freshers and beginners level, like up to 2 to 4 years of experience, and some tips and tricks to answer them correctly.  
  
By the way, these thread interview questions are equally useful for senior Java developers or guys with some Java experience in hand. I have tried to share answers to these interview questions on the thread as well but I suggest you do some research and learn the topic well to answer any follow-up questions, which comes due to your response to these *thread questions in Java*.  
  
Anyway here is my collection of **Java thread interview questions** and how to answers them in Java :

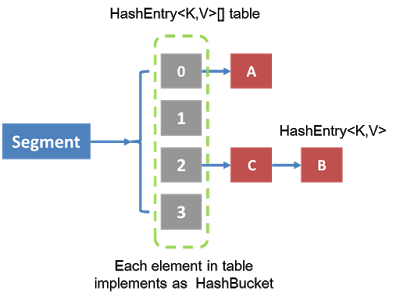
**1) What is the difference between start and run method in Java Thread?**([answer](http://javarevisited.blogspot.com/2014/09/common-java-multi-threading-mistakes-1-run-vs-start.html))  
This thread interview question is also ask as if start() method eventually call run() method then why do you need to call start() method, why not call run() method directly. well, reason is that because start method creates a new thread and call the code written inside the run method on a new thread while calling run method executes that code on the same thread. You can also see the article [start vs run method in Java](http://javarevisited.blogspot.sg/2012/03/difference-between-start-and-run-method.html) for more details.  
  
  
**2) Write code to avoid deadlock in Java where N threads are accessing N shared resources?**([answer](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html))  
This is a classic Java multithreading interview questions, which appears on almost every list of Java thread questions. This question is based on risk and issues faced by parallel programs without proper synchronization or incorrect synchronization. This question explores the concept of looking and best practices on acquiring and releasing the lock on shared resource. By the way, it's been covered in many places as well and I suggest reading  [How to prevent deadlock in Java](http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html), not only for detail answer of this Java multithreading question but also to learn how to prevent deadlock in Java.  
  
  
  
**3) Which one is better to implement thread in Java ? extending Thread class or implementing Runnable?**([answer](http://javarevisited.blogspot.com/2012/01/difference-thread-vs-runnable-interface.html))  
Well, this is another frequently asked questions on any Java thread interview. Essentially these are two way to implement Thread in Java, by extending java.lang.Thread class or by implementing java.lang.Runnable interface.  
  
By extending the class you are using your chance to extend one any only one class as Java does not support multiple inheritances, by implementing a Runnable interface you can still extend another class. So extending Runnable or even Callable is a better choice. You can also see [Runnable vs Thread class in Java](http://javarevisited.blogspot.sg/2012/01/difference-thread-vs-runnable-interface.html) for more answers on this questions.  
  
Given its simplicity and fact-based nature, this question mostly appears on either telephonic round or initial screening rounds. Key points to mention, while answering this question includes multiple inheritance at the class level and separation of defining a task and execution of a task. Runnable only represent a task, while Thread represent both tasks and it's execution.  
  
  
  
**4) What is Busy Spinning? Why will you use Busy Spinning as wait strategy?**([answer](http://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html))  
This is one of the advanced concurrency interview questions in Java and only asked to experienced and senior Java developers, with lots of concurrent coding experience under the belt. By the way, the concept of *busy spinning* is not new, but its usage with multi-core processor has risen recently (see [The Art of Multiprocessor Programming](https://www.amazon.com/Art-Multiprocessor-Programming-Revised-Reprint/dp/0123973376?tag=javamysqlanta-20)).

[](https://www.amazon.com/Art-Multiprocessor-Programming-Revised-Reprint/dp/0123973376?tag=javamysqlanta-20)

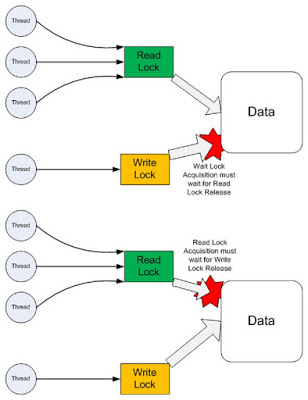
The busy waiting is a wait strategy, where one thread wait for a condition to become true, but instead of calling wait or sleep method and releasing CPU, it just spins. This is particularly useful if the condition is going to be true quite quickly i.e. in a millisecond or microsecond.  
  
The advantage of not releasing CPU is that all cached data and instruction remain unaffected, which may be lost, had this thread is suspended on one core and brought back to another thread. If you can answer this question, that rest assure of a good impression.  
  
  
**5) What is the difference between CountDownLatch and CyclicBarrier in Java?**([answer](http://java67.blogspot.com/2012/08/difference-between-countdownlatch-and-cyclicbarrier-java.html))  
The [CountDownLatch](http://javarevisited.blogspot.sg/2012/07/countdownlatch-example-in-java.html)and [CyclicBarrier in Java](http://javarevisited.blogspot.sg/2012/07/cyclicbarrier-example-java-5-concurrency-tutorial.html)are two important concurrency utility which is added on Java 5 Concurrency API. Both are used to implement scenario, where one thread has to wait for other thread before starting processing but there is a difference between them.  
  
The key point to mention, while answering this question is that CountDownLatch is not reusable once the count reaches to zero, while CyclicBarrier can be reused even after the barrier is broken.  
  
You can also see my previous article [difference between CyclicBarrier and CountDownLatch in Java](http://java67.blogspot.com/2012/08/difference-between-countdownlatch-and-cyclicbarrier-java.html) for a more detailed answer of this concurrency interview question and a real life example of where to use these concurrency utilities.  
  
  
  
**6) What is the difference between wait and sleep in Java?**([method](http://javarevisited.blogspot.com/2011/12/difference-between-wait-sleep-yield.html))  
One more *classic Java multithreading question* from the telephonic round of interviews. The key point to mention while answering this question is to mention that wait will release the lock and must be called from the synchronized context, while sleep will only pause the thread for some time and keep the lock.  
  
By the way, both methods throw IntrupptedException and can be interrupted, which can lead to some follow-up questions like, can we awake a sleeping or waiting for a thread in Java? You can also read a detailed answer on my post of same title [here](http://java67.blogspot.com/2012/08/what-are-difference-between-wait-and.html).

[](https://2.bp.blogspot.com/-kJOX8eUPO80/V10_YpHWQYI/AAAAAAAAGNg/6x_PZfACG4wwiacTyn25tkE3B7TO85P0QCLcB/s1600/Sleep%2Bvs%2Bwait%2Bvs%2Byield%2Bmethod%2Bjava.gif)

**7) How do you solve producer consumer problem in Java?**([solution](http://java67.blogspot.com/2015/12/producer-consumer-solution-using-blocking-queue-java.html))  
One of my favorite questions during any Java multithreading interview, Almost half of the concurrency problems can be categorized in the producer-consumer pattern. There are basically two ways to solve this problem in Java, One by [using wait and notify method](http://java67.blogspot.com/2012/12/producer-consumer-problem-with-wait-and-notify-example.html) and other by using [BlockingQueue](http://javarevisited.blogspot.com/2014/06/synchronousqueue-example-in-java.html) in Java.  later is easy to implement and a good choice if you are coding in Java 5.  
  
The key points to mention, while answering this question is thread-safety and blocking nature of BlockingQueue and how that helps, while writing concurrent code. You can also expect lots of follow-up questions including, what happen if you have multiple producer or multiple consumers, what will happen if a producer is faster than consumer thread or vice-versa. You can also see this link for example of [how to code producer-consumer design in Java using blocking queue](http://javarevisited.blogspot.sg/2012/02/producer-consumer-design-pattern-with.html)  
  
  
  
**8) Why is ConcurrentHashMap faster than Hashtable in Java?**([answer](http://javarevisited.blogspot.com/2016/05/what-is-difference-between-synchronized.html))  
ConcurrentHashMap is introduced as an alternative of Hashtable in Java 5, it is faster because of its design. ConcurrentHashMap divides the whole map into different segments and only lock a particular segment during the update operation, instead of [Hashtable](http://javarevisited.blogspot.com/2012/01/java-hashtable-example-tutorial-code.html), which locks whole Map.  
  
The ConcurrentHashMap also provides lock-free read, which is not possible in Hashtable, because of this and lock striping, ConcurrentHashMap is faster than Hashtable, especially when a number of the reader is more than the number of writers.  
  
In order to better answer this popular Java concurrency interview questions, I suggest reading my post about the [internal working of ConcurrentHashMap in Java](http://javarevisited.blogspot.com/2013/02/concurrenthashmap-in-java-example-tutorial-working.html).

[](https://1.bp.blogspot.com/-s28yqJthVWo/V10_G31sSCI/AAAAAAAAGNY/8oJE5cDD5BUa5VBiCI8Bx-UdpD8_MoeSwCLcB/s1600/Internal%2Bimplementation%2Bof%2BConcurrentHashMap%2Bin%2BJava.png)

**9) What is the difference between submit() and execute() method of Executor and ExecutorService in Java? (**[answer](http://javarevisited.blogspot.com/2016/04/difference-between-ExecutorServie-submit-vs-Executor-execute-method-in-Java.html)**)**  
The main difference between submit and execute method from ExecutorService interface is that former return a result in the form of a Future object, while later doesn't return a result. By the way, both are used to submit a task to thread pool in Java but one is defined in Executor interface,while other is added into ExecutorService interface. This multithreading interview question is also asked in the first round of Java interviews.  
  
  
**10) How do you share data between two threads in Java? (**[answer](http://javarevisited.blogspot.com/2013/12/inter-thread-communication-in-java-wait-notify-example.html)**)**  
One more Java multithreading question from the telephonic round of interview. You can share data between thread by using shared object or shared data structures like Queue. Depending upon, what you are using, you need to provide the thread-safety guarantee, and one way of providing thread-safety is using synchronized keyword.  
  
If you use concurrent collection classes from Java 5 e.g. [BlockingQueue](http://javarevisited.blogspot.com/2012/12/blocking-queue-in-java-example-ArrayBlockingQueue-LinkedBlockingQueue.html), you can easily share data without being bothered about thread safety and inter-thread communication. I like this thread question, because of it's simplicity and effectiveness. This also leads further follow-up questions on issues which arise due to sharing data between threads e.g. race conditions.  
  
  
**11) What is ReentrantLock in Java? Have you used it before? (**[answer](http://javarevisited.blogspot.com/2014/07/top-50-java-multithreading-interview-questions-answers.html)**)**  
ReentrantLock is an alternative of synchronized keyword in Java, it is introduced to handle some of the limitations of synchronized keywords. Many concurrency utility classes and concurrent collection classes from Java 5, including ConcurrentHashMap uses ReentrantLock, to leverage optimization.  
  
The ReentrantLock mostly uses an atomic variable and faster CAS operation to provides better performance. Key points to mention are the [difference between ReentrantLock and synchronized keyword in Java](http://javarevisited.blogspot.com/2013/03/reentrantlock-example-in-java-synchronized-difference-vs-lock.html), which includes the ability to acquire lock interruptibly, timeout feature while waiting for lock etc. ReentrantLock also gives the option to create fair lock in Java.Once again a very good Java concurrency interview question for experienced Java programmers.  
  
  
**12) What is ReadWriteLock in Java? What is the benefit of using ReadWriteLock in Java?(**[answer](http://javarevisited.blogspot.com/2014/10/how-to-use-locks-in-multi-threaded-java-program-example.html)**)**  
This is usually a follow-up question of previous Java concurrency questions. The  ReadWriteLock is again based upon the concept of lock striping, one of the advance thread-safety mechanism which advocates separating locks for reading and writing operations (see [Concurrent Programming in Java: Patterns and principles](https://www.amazon.com/Concurrent-Programming-Java%C2%99-Principles-Pattern/dp/0201310090?tag=javamysqlanta-20) by Doug Lea for more details).  
  
If you have noticed before, reading operation can be done without locking if there is no writer and that can hugely improve the performance of any application. The ReadWriteLock leverage this idea and provide policies to allow maximum concurrency level. Java Concurrency API also provides an implementation of this concept as ReentrantReadWriteLock.  
  
Depending upon Interviewer and experience of the candidate, you can even expect to provide your own implementation of ReadWriteLock, so be prepare for that as well.  
  
Here is a nice diagram which clearly explains working of read-write lock in Java:

[](https://4.bp.blogspot.com/-yYqp8m0BGdk/V11A-s0qwVI/AAAAAAAAGNw/LL_qGX5_qD4dHTFMe3kklI4jYWyDHzIyQCLcB/s1600/ReadWriteLock%2Bin%2BJava.jpg)

These were some of my favorite*interview questions based on multithreading and concurrent in Java*. Threading and Concurrency is a big topic in Java and has lots of interesting, [tricky and tough question](http://java67.blogspot.com/2012/09/top-10-tricky-java-interview-questions-answers.html) but for starters and freshers, these questions certainly help to clear any thread interview in Java.   
  
As I said, mentioning the key points are very important while answering questions on multithreading and concurrency.  I also suggest further reading [Java Concurrency in Practice](http://www.amazon.com/dp/0321349601/?tag=javamysqlanta-20) to learn more about locking, synchronization, concurrent collections and concurrency utility classes to do well in core Java and multithreading interviews.

Read more: <http://www.java67.com/2012/08/5-thread-interview-questions-answers-in.html#ixzz4DQSTuIUA>