5 Technical Core Java Questions Answers asked in Telephonic Interviews for 5 to 6 Years Experienced Programmer

Once experience of Java Programmers grow in 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* also change a bit. Ofcourse basics like**data structure**, **algorithms**and **object-oriented programming** remains same, but sort of questions and there answers will definitely started more detailed. I often receive queries about core Java questions asked to a senior developer, or I am going for an interview of senior Java developer, what kind of question I should expect. This some time puzzles me, that once you become senior, you automatically starts taking part on interview, and sort of have an idea on what to expect on Interviews, but same time, I can understand that having an idea of questions before going on Interview, helps preparation. Ofcourse, you are not going to get 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 see much changes on telephonic round, which remains same, some fact based, some tricky question. On face to face, yes it become more detailed and more tricky, especially with nasty follow-ups. In this article, I am going to share **15 technical core Java Interview Questions**, which I have seen to be asked to Senior and experienced developers on different interviews, mostly on telephonic rounds. I am not posting answers as of now, but  you can find answers of most of questions on here or [Javarevisited blog](http://javarevisited.blogspot.co.uk/).

**15 Core Java Interview Questions for 5 yo 6 years Experienced Programmers**

[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 this question in on different rounds, including telephonic and face to face rounds on different companies, mostly on banks.  
  
**1) How does get method of HashMap works in Java?**  
Yes, this is still one of the most popular questions for senior developer,  you can expect this on telephonic round, followed by lot's of related questions, see [here](http://java67.blogspot.sg/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html)for answers  
  
**2) Which two method HashMap key object should implement?**  (equals and hashcode)  
**3) Why should an object used as key should be Immutable?** (so that hashcode always return same value)  
  
**4) How does ConcurrentHashMap achieves it's scalability?** or sometime this interview question is also asked as,difference between ConcurrentHashMap and Hashtable in Java, see [here](http://java67.blogspot.sg/2012/08/difference-between-hashmap-and-concurrentHashMap-java-collection.html)for answers.  
  
**5) 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.  
  
**6) How do find if your program has deadlock?** ( By taking thread dump using kill -3, using JConsole orVisualVM), 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?  
  
**7) 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)  
  
**8) What is busy spinning? Why should you use it?**  
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 CPU for going to [sleep](http://java67.blogspot.sg/2012/08/what-are-difference-between-wait-and.html). This can be used in a particular scenario, where wait time is very minimal, by not releasing CPU or suspending thread, your thread retain all cached data andinstruction, which may be lost if further 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 range of Micro to Milli seconds.  
  
**9) What is ReadWrite Lock? Does ConcurrentHashMap uses ReadWrite Lock?**  
ReadWrite Lock is an implementation of lock stripping, where two separate locks are used for read and write operation. Since read operation doesn't modify state of object, it's safe to allow multiple access of shared object to multiple reader without locking, and by splitting lock into ReadLock and WriteLock, you can easily do that. Java provides an implementation of ReadWriteLock in form of ReentrantReadWritLock, which is worth looking. Also[ConcurrentHashMap](http://javarevisited.blogspot.sg/2013/02/concurrenthashmap-in-java-example-tutorial-working.html) doesn't use ReadWriteLock, instead it divides maps into several segments and lock them separately using different locks, which means any given time, only a portion of map is locked, instead of whole map. This question is also very popular on Senior and experienced level Java interviews, expect Interviewer to go into more detail, e.g. asking you to provided an implementation of ReadWriteLock with different policies.  
  
**10) 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 upon 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).  
  
**11) Which design patterns have you used?**  
Always expect design and patterns related question for Senior developer Core Java Interview. It's best 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 design pattern, which you have really used in your project and knows it's tradeoffs. As once you mention a particular design pattern say Factory, Interviewer's next question would be, have you used in your project? So be ready with proper example and why you choose a particular pattern.  
  
**12) Do you know about Open Closed Design Principle or Liskov Substitution Principle?**  
Design patterns are based upon object oriented design principles. I strongly suggest to take a look at my article [10 SOLID and Object Oriented design principle](http://javarevisited.blogspot.de/2012/03/10-object-oriented-design-principles.html), Java programmer should know, to at least have a basic idea of what are these principles and how they help you to write better object oriented code. If you don't know answer of this question, you can politely say no, as it's not expected from you to know answer of every question, but answering question, which most developer doesn't answer, can make your candidature much stronger.  
 **13) Which design pattern you will use to shield your code from a third party library, which will likely to be replaced in another couple of years?**  
This is just another example of scenario based design pattern question, you can expect in different formats, some with more detail explanation with context, or some with only intent around. One way to shield your code form third party library is to depend upon interface rather than implementation and than use dependency injection to provide a particular implementation. This kind of questions are also asked quite frequently to experienced and senior *Java developers with 5 to 7 years of experience*.  
 **14)  How  do you prevent SQL Injection in Java Code?**  
This question is more asked to Java EE developers than core Java developers but still a good question to know,[PreparedStatement](http://javarevisited.blogspot.sg/2012/03/why-use-preparedstatement-in-java-jdbc.html)is the way to go. PreparedStatement not only provides better performance but also shield from SQL Injection attack. If you are working more on Java EE or J2EE side, than you should also be familiar with other securityissues including Session Fixation attack or Cross Site Scripting attack and how to resolve them.  
 **15) Tell me about different reference type 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 case of low memory issues. If you wrap an object with WeakReference than it will be garbage collected if there is no strong reference and GC is running low on memory. WeakHashMap is a Map implementation, whose keys uses WeakReference, so if only Map contains reference of any object and no other, those object can be garbage collected, if GC needs memory.  
  
That's all on this **list of core Java Interview Questions for senior developers**. I haven't included lot of question from another important topic Exception handling, Garbage Collection tuning and JVM Internals, which is also very popular among *Java programmers with 5 to 6 years of experience*, may be I will include them in next part. By the way, if you don't find answer of any of these core Java Question, let me know. I may update the post with more detailed option, based upon my readers request.

**How to make HashMap synchronized.**

HashMap can be synchronized by *Map m = Collections.synchronizedMap(hashMap);*

**Difference between Comparable and Comparator**

[**http://www.digizol.com/2008/07/java-sorting-comparator-vs-comparable.html**](http://www.digizol.com/2008/07/java-sorting-comparator-vs-comparable.html)

Comparable

A comparable object is capable of comparing itself with another object. The class itself must implements the java.lang.Comparable interface in order to be able to compare its instances.

Comparator

A comparator object is capable of comparing two different objects. The class is not comparing its instances, but some other class’s instances. This comparator class must implement the java.util.Comparator interface. - See more at: <http://www.digizol.com/2008/07/java-sorting-comparator-vs-comparable.html#sthash.KWbbbXvx.dpuf>

**Java Enumeration :**

<http://javarevisited.blogspot.com/2011/08/enum-in-java-example-tutorial.html>

Enum in Javais a keyword, a feature which is used to represent fixed number of well known values in Java, For example Number of days in Week, Number of planets in Solar system etc. **Enumeration (Enum) in Java** was introduced in JDK 1.5

### How to represent enumerable value without Java enum

[java enum example, enum in java tutorial](http://javarevisited.blogspot.com/2011/08/convert-string-to-integer-to-string.html)Since **Enum in Java** is only available from **Java 1.5** its worth to discuss how we used to represent enumerable values in Java prior JDK 1.5 and without it. I use public static [final constant](http://javarevisited.blogspot.sg/2011/12/final-variable-method-class-java.html) to replicate enum like behavior. Let’s see an Enum example in Java to understand the concept better. In this example we will use US Currency Coin as enumerable which has values like PENNY (1) NICKLE (5), DIME (10), and QUARTER (25).

**public class** CurrencyDenom {

**public** **static** **final** **int** *PENNY* = 1;

**public** **static** **final** **int** *NICKLE* = 5;

**public** **static** **final** **int** *DIME* = 10;

**public** **static** **final** **int** *QUARTER* = 25;

}

**public class** Currency {

**private int** currency; //CurrencyDenom.PENNY,CurrencyDenom.NICKLE,

                         // CurrencyDenom.DIME,CurrencyDenom.QUARTER

}  
  
 Though this can server our purpose it has some serious limitations:  
  
**1) No Type-Safety**: First of all it’s not [type-safe](http://javarevisited.blogspot.sg/2011/09/generics-java-example-tutorial.html); you can assign any valid int value to currency e.g. 99 though there is no coin to represent that value.  
  
**2) No Meaningful Printing**: printing value of any of these constant will print its numeric value instead of meaningful name of coin e.g. when you print NICKLE it will print "5" instead of "NICKLE"  
  
**3) No namespace:** to access the currencyDenom constant we need to prefix class name e.g. CurrencyDenom.PENNY instead of just using PENNY though this can also be achieved by using [static import in JDK 1.5](http://javarevisited.blogspot.sg/2011/11/static-keyword-method-variable-java.html)  
  
**Java Enum** is answer of all this limitation. Enum in Java is type-safe, provides meaningful String names and has there own namespace. Now let's see same example using Enum in Java:

**public** **enum** Currency {PENNY, NICKLE, DIME, QUARTER};

Here Currency is our **enum** and PENNY, NICKLE, DIME, QUARTER are **enum constants**. Notice **curly braces around enum constants** because Enum are type like [class](http://javarevisited.blogspot.sg/2011/10/class-in-java-programming-general.html)and[interface in Java](http://javarevisited.blogspot.sg/2012/04/10-points-on-interface-in-java-with.html). Also we have followed similar naming convention for enum like class and interface (first letter in Caps) and since *Enum constants are implicitly static final* we have used all caps to specify them like Constants in Java.

### What is Enum in Java

Now back to primary questions **“What is Enum in java”** simple *answer Enum is a keyword in java* and on more detail term Java Enum is type like class and interface and can be used to define a set of Enum constants. Enum constants are [implicitly static and final](http://javarevisited.blogspot.sg/2011/12/final-variable-method-class-java.html) and you can not change there value once created. Enum in Java provides type-safety and can be used inside switch statment like int variables. Since enum is a keyword you can not use as variable name and since its only introduced in JDK 1.5 all your previous code which has enum as variable name will not work and needs to be re-factored.  
  
Read more: <http://javarevisited.blogspot.com/2011/08/enum-in-java-example-tutorial.html#ixzz2wY34r6kG>

## What is autoboxing and unboxing in Java

[What is auboxing means in Java with Example](http://3.bp.blogspot.com/-K6q0DQ1v-tw/TWu8owBtc2I/AAAAAAAAADA/oBoHDBiJ8ag/s1600/17.jpg)When Java automatically converts a primitive type like int into corresponding wrapper class object e.g. [Integer](http://javarevisited.blogspot.sg/2011/08/convert-string-to-integer-to-string.html) than its called **autoboxing**  because primitive is boxed into wrapper class while in opposite case is called**unboxing**, where an Integer object is converted into primitive int.

**What is Generics in Java**

[Java Generics example tutorial , parametrized type class method](http://4.bp.blogspot.com/-v_PbvhXcJ7I/UBJ7S-JdzAI/AAAAAAAAAZ4/Z7XHJpHQ_f4/s1600/scratch_001.gif)Generic in Java is added to provide compile time type-safety of code and removing risk of ClassCastException at [runtime](http://javarevisited.blogspot.sg/2012/03/what-is-static-and-dynamic-binding-in.html) which was quite frequent error in Java code, for those who doesn’t know what is type-safety at compile time, it’s just a check by compiler that correct Type is used in correct place and there should not be any ClassCastException.  
  
Read more: <http://javarevisited.blogspot.com/2011/09/generics-java-example-tutorial.html#ixzz2wYAfBxrR>

<http://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html>