**Core Java Interview Questions and Answers**

1. **What are the important features of Java 8 release?**

Java 8 has been released in March 2014, so it’s one of the hot topic in java interview questions. If you answer this question clearly, it will show that you like to keep yourself up-to-date with the latest technologies.

Java 8 has been one of the biggest release after Java 5 annotations and generics. Some of the important features of Java 8 are:

* 1. [Interface changes with default and static methods](http://www.journaldev.com/2752/java-8-interface-changes-static-method-default-method)
  2. [Functional interfaces and Lambda Expressions](http://www.journaldev.com/2763/java-8-functional-interfaces)
  3. [Java Stream API for collection classes](http://www.journaldev.com/2774/java-8-stream)
  4. [Java Date Time API](http://www.journaldev.com/2800/java-8-date-localdate-localdatetime-instant)

I strongly recommend to go through above links to get proper understanding of each one of them, also read [Java 8 Features](http://www.journaldev.com/2389/java-8-features-with-examples).

1. **What do you mean by platform independence of Java?**

Platform independence means that you can run the same Java Program in any Operating System. For example, you can write java program in Windows and run it in Mac OS.

1. **What is JVM and is it platform independent?**

Java Virtual Machine (JVM) is the heart of java programming language. JVM is responsible for converting byte code into machine readable code. JVM is not platform independent, thats why you have different JVM for different operating systems. We can customize JVM with Java Options, such as allocating minimum and maximum memory to JVM. It’s called virtual because it provides an interface that doesn’t depend on the underlying OS.

1. **What is the difference between JDK and JVM?**

Java Development Kit (JDK) is for development purpose and JVM is a part of it to execute the java programs.

JDK provides all the tools, executables and binaries required to compile, debug and execute a Java Program. The execution part is handled by JVM to provide machine independence.

1. **What is the difference between JVM and JRE?**

Java Runtime Environment (JRE) is the implementation of JVM. JRE consists of JVM and java binaries and other classes to execute any program successfully. JRE doesn’t contain any development tools like java compiler, debugger etc. If you want to execute any java program, you should have JRE installed.

1. **Which class is the superclass of all classes?**

java.lang.Object is the root class for all the java classes and we don’t need to extend it.

1. **Why Java doesn’t support multiple inheritance?**

Java doesn’t support multiple inheritance in classes because of “Diamond Problem”. To know more about diamond problem with example, read [Multiple Inheritance in Java](http://www.journaldev.com/1775/multiple-inheritance-in-java).

However multiple inheritance is supported in interfaces. An interface can extend multiple interfaces because they just declare the methods and implementation will be present in the implementing class. So there is no issue of diamond problem with interfaces.

1. **Why Java is not pure Object Oriented language?**

Java is not said to be pure object oriented because it support primitive types such as int, byte, short, long etc. I believe it brings simplicity to the language while writing our code. Obviously java could have wrapper objects for the primitive types but just for the representation, they would not have provided any benefit.

As we know, for all the primitive types we have wrapper classes such as Integer, Long etc that provides some additional methods.

1. **What is difference between path and classpath variables?**

PATH is an environment variable used by operating system to locate the executables. That’s why when we install Java or want any executable to be found by OS, we need to add the directory location in the PATH variable. If you work on Windows OS, read this post to learn [how to setup PATH variable on Windows](http://www.journaldev.com/476/java-tutorial-1-setting-up-java-environment-on-windows).

Classpath is specific to java and used by java executables to locate class files. We can provide the classpath location while running java application and it can be a directory, ZIP files, JAR files etc.

1. **What is the importance of main method in Java?**

main() method is the entry point of any standalone java application. The syntax of main method is public static void main(String args[]).

main method is public and static so that java can access it without initializing the class. The input parameter is an array of String through which we can pass runtime arguments to the java program. Check this post to learn [how to compile and run java program](http://www.journaldev.com/481/java-hello-world-program).

1. **What is overloading and overriding in java?**

When we have more than one method with same name in a single class but the arguments are different, then it is called as method overloading.

Overriding concept comes in picture with inheritance when we have two methods with same signature, one in parent class and another in child class. We can use @Override annotation in the child class overridden method to make sure if parent class method is changed, so as child class.

1. **Can we overload main method?**

Yes, we can have multiple methods with name “main” in a single class. However if we run the class, java runtime environment will look for main method with syntax as public static void main(String args[]).

1. **Can we have multiple public classes in a java source file?**

We can’t have more than one public class in a single java source file. A single source file can have multiple classes that are not public.

1. **What is Java Package and which package is imported by default?**

Java package is the mechanism to organize the java classes by grouping them. The grouping logic can be based on functionality or modules based. A java class fully classified name contains package and class name. For example, java.lang.Object is the fully classified name of Object class that is part of java.lang package.

java.lang package is imported by default and we don’t need to import any class from this package explicitly.

1. **What are access modifiers?**

Java provides access control through public, private and protected access modifier keywords. When none of these are used, it’s called default access modifier.  
A java class can only have public or default access modifier. Read [Java Access Modifiers](http://www.journaldev.com/2345/java-access-modifiers) to learn more about these in detail.



1. **What is final keyword?**

final keyword is used with Class to make sure no other class can extend it, for example String class is final and we can’t extend it.

We can use final keyword with methods to make sure child classes can’t override it.

final keyword can be used with variables to make sure that it can be assigned only once. However the state of the variable can be changed, for example we can assign a final variable to an object only once but the object variables can change later on.

Java interface variables are by default final and static.

1. **What is static keyword?**

static keyword can be used with class level variables to make it global i.e all the objects will share the same variable.

static keyword can be used with methods also. A static method can access only static variables of class and invoke only static methods of the class.

Read more in detail at [java static keyword](http://www.journaldev.com/1365/java-static-keyword-class-method-variable-block-import).

1. **What is finally and finalize in java?**

finally block is used with try-catch to put the code that you want to get executed always, even if any exception is thrown by the try-catch block. finally block is mostly used to release resources created in the try block.

finalize() is a special method in Object class that we can override in our classes. This method get’s called by garbage collector when the object is getting garbage collected. This method is usually overridden to release system resources when object is garbage collected.

1. **Can we declare a class as static?**

We can’t declare a top-level class as static however an inner class can be declared as static. If inner class is declared as static, it’s called static nested class.  
Static nested class is same as any other top-level class and is nested for only packaging convenience.

Read more about inner classes at [java inner class](http://www.journaldev.com/996/java-inner-class).

1. **What is static import?**

If we have to use any static variable or method from other class, usually we import the class and then use the method/variable with class name.

import java.lang.Math;

//inside class

double test = Math.PI \* 5;

We can do the same thing by importing the static method or variable only and then use it in the class as if it belongs to it.

import static java.lang.Math.PI;

//no need to refer class now

double test = PI \* 5;

Use of static import can cause confusion, so it’s better to avoid it. Overuse of static import can make your program unreadable and unmaintainable.

1. **What is try-with-resources in java?**

One of the Java 7 features is try-with-resources statement for automatic resource management. Before Java 7, there was no auto resource management and we should explicitly close the resource. Usually, it was done in the finally block of a try-catch statement. This approach used to cause memory leaks when we forgot to close the resource.

From Java 7, we can create resources inside try block and use it. Java takes care of closing it as soon as try-catch block gets finished. Read more at [Java Automatic Resource Management](http://www.journaldev.com/592/java-try-with-resources).

1. **What is multi-catch block in java?**

Java 7 one of the improvement was multi-catch block where we can catch multiple exceptions in a single catch block. This makes are code shorter and cleaner when every catch block has similar code.

If a catch block handles multiple exception, you can separate them using a pipe (|) and in this case exception parameter (ex) is final, so you can’t change it.

Read more at [Java multi catch block](http://www.journaldev.com/629/java-catch-multiple-exceptions-rethrow-exception).

1. **What is static block?**

Java static block is the group of statements that gets executed when the class is loaded into memory by Java ClassLoader. It is used to initialize static variables of the class. Mostly it’s used to create static resources when class is loaded.

1. **What is an interface?**

Interfaces are core part of java programming language and used a lot not only in JDK but also java design patterns, most of the frameworks and tools. Interfaces provide a way to achieve abstraction in java and used to define the contract for the subclasses to implement.

Interfaces are good for starting point to define Type and create top level hierarchy in our code. Since a java class can implements multiple interfaces, it’s better to use interfaces as super class in most of the cases. Read more at [java interface](http://www.journaldev.com/1601/interface-in-java).

1. **What is an abstract class?**

Abstract classes are used in java to create a class with some default method implementation for subclasses. An abstract class can have abstract method without body and it can have methods with implementation also.

abstract keyword is used to create a abstract class. Abstract classes can’t be instantiated and mostly used to provide base for sub-classes to extend and implement the abstract methods and override or use the implemented methods in abstract class. Read important points about abstract classes at [java abstract class](http://www.journaldev.com/1582/abstract-class-in-java).

1. **What is the difference between abstract class and interface?**

abstract keyword is used to create abstract class whereas interface is the keyword for interfaces.

Abstract classes can have method implementations whereas interfaces can’t.

A class can extend only one abstract class but it can implement multiple interfaces.

We can run abstract class if it has main() method whereas we can’t run an interface.

Some more differences in detail are at [Difference between Abstract Class and Interface](http://www.journaldev.com/1607/difference-between-abstract-class-and-interface-in-java).

1. **Can an interface implement or extend another interface?**

Interfaces don’t implement another interface, they extend it. Since interfaces can’t have method implementations, there is no issue of diamond problem. That’s why we have multiple inheritance in interfaces i.e an interface can extend multiple interfaces.

1. **What is Marker interface?**

A marker interface is an empty interface without any method but used to force some functionality in implementing classes by Java. Some of the well known marker interfaces are Serializable and Cloneable.

1. **What are Wrapper classes?**

Java wrapper classes are the Object representation of eight primitive types in java. All the wrapper classes in java are immutable and final. Java 5 autoboxing and unboxing allows easy conversion between primitive types and their corresponding wrapper classes.

Read more at [Wrapper classes in Java](http://www.journaldev.com/1002/wrapper-class-in-java).

1. **What is Enum in Java?**

Enum was introduced in Java 1.5 as a new type whose fields consists of fixed set of constants. For example, in Java we can create Direction as enum with fixed fields as EAST, WEST, NORTH, SOUTH.

enum is the keyword to create an enum type and similar to class. Enum constants are implicitly static and final. Read more in detail at [java enum](http://www.journaldev.com/716/java-enum).

1. **What is Java Annotations?**

Java Annotations provide information about the code and they have no direct effect on the code they annotate. Annotations are introduced in Java 5. Annotation is metadata about the program embedded in the program itself. It can be parsed by the annotation parsing tool or by compiler. We can also specify annotation availability to either compile time only or till runtime also. Java Built-in annotations are @Override, @Deprecated and @SuppressWarnings. Read more at [java annotations](http://www.journaldev.com/721/java-annotations).

1. **What is Java Reflection API? Why it’s so important to have?**

Java Reflection API provides ability to inspect and modify the runtime behavior of java application. We can inspect a java class, interface, enum and get their methods and field details. Reflection API is an advanced topic and we should avoid it in normal programming. Reflection API usage can break the design pattern such as Singleton pattern by invoking the private constructor i.e violating the rules of access modifiers.

Even though we don’t use Reflection API in normal programming, it’s very important to have. We can’t have any frameworks such as Spring, Hibernate or servers such as Tomcat, JBoss without Reflection API. They invoke the appropriate methods and instantiate classes through reflection API and use it a lot for other processing.

Read [Java Reflection Tutorial](http://www.journaldev.com/1789/java-reflection-example-tutorial) to get in-depth knowledge of reflection api.

1. **What is composition in java?**

Composition is the design technique to implement has-a relationship in classes. We can use Object composition for code reuse.

Java composition is achieved by using instance variables that refers to other objects. Benefit of using composition is that we can control the visibility of other object to client classes and reuse only what we need. Read more with example at [Java Composition](http://www.journaldev.com/1325/composition-in-java-example) example.

1. **What is the benefit of Composition over Inheritance?**

One of the best practices of java programming is to “favor composition over inheritance”. Some of the possible reasons are:

* 1. Any change in the superclass might affect subclass even though we might not be using the superclass methods. For example, if we have a method test() in subclass and suddenly somebody introduces a method test() in superclass, we will get compilation errors in subclass. Composition will never face this issue because we are using only what methods we need.
  2. Inheritance exposes all the super class methods and variables to client and if we have no control in designing superclass, it can lead to security holes. Composition allows us to provide restricted access to the methods and hence more secure.
  3. We can get runtime binding in composition where inheritance binds the classes at compile time. So composition provides flexibility in invocation of methods.

You can read more about above benefits of composition over inheritance at [java composition vs inheritance](http://www.journaldev.com/1775/multiple-inheritance-in-java).

1. **How to sort a collection of custom Objects in Java?**

We need to implement Comparable interface to support sorting of custom objects in a collection. Comparable interface has compareTo(T obj) method which is used by sorting methods and by providing this method implementation, we can provide default way to sort custom objects collection.

However, if you want to sort based on different criteria, such as sorting an Employees collection based on salary or age, then we can create Comparator instances and pass it as sorting methodology. For more details read [Java Comparable and Comparator](http://www.journaldev.com/780/comparable-and-comparator-in-java-example).

1. **What is inner class in java?**

We can define a class inside a class and they are called nested classes. Any non-static nested class is known as inner class. Inner classes are associated with the object of the class and they can access all the variables and methods of the outer class. Since inner classes are associated with instance, we can’t have any static variables in them.

We can have local inner class or anonymous inner class inside a class. For more details read [java inner class](http://www.journaldev.com/996/java-inner-class).

1. **What is anonymous inner class?**

A local inner class without name is known as anonymous inner class. An anonymous class is defined and instantiated in a single statement. Anonymous inner class always extend a class or implement an interface.

Since an anonymous class has no name, it is not possible to define a constructor for an anonymous class. Anonymous inner classes are accessible only at the point where it is defined.

1. **What is Classloader in Java?**

Java Classloader is the program that loads byte code program into memory when we want to access any class. We can create our own classloader by extending ClassLoader class and overriding loadClass(String name) method. Learn more at [java classloader](http://www.journaldev.com/349/java-classloader).

1. **What are different types of classloaders?**

There are three types of built-in Class Loaders in Java:

* 1. Bootstrap Class Loader – It loads JDK internal classes, typically loads rt.jar and other core classes.
  2. Extensions Class Loader – It loads classes from the JDK extensions directory, usually $JAVA\_HOME/lib/ext directory.
  3. System Class Loader – It loads classes from the current classpath that can be set while invoking a program using -cp or -classpath command line options.

1. **What is ternary operator in java?**

Java ternary operator is the only conditional operator that takes three operands. It’s a one liner replacement for if-then-else statement and used a lot in java programming. We can use ternary operator if-else conditions or even switch conditions using nested ternary operators. An example can be found at [java ternary operator](http://www.journaldev.com/963/java-ternary-operator).

1. **What does super keyword do?**

super keyword can be used to access super class method when you have overridden the method in the child class.

We can use super keyword to invoke super class constructor in child class constructor but in this case it should be the first statement in the constructor method.

package com.journaldev.access;

public class SuperClass {

public SuperClass(){

}

public SuperClass(int i){}

public void test(){

System.out.println("super class test method");

}

}

Use of super keyword can be seen in below child class implementation.

package com.journaldev.access;

public class ChildClass extends SuperClass {

public ChildClass(String str){

//access super class constructor with super keyword

super();

//access child class method

test();

//use super to access super class method

super.test();

}

@Override

public void test(){

System.out.println("child class test method");

}

}

1. **What is break and continue statement?**

We can use break statement to terminate for, while, or do-while loop. We can use break statement in switch statement to exit the switch case. You can see the example of break statement at [java break](http://www.journaldev.com/588/java-switch-case-string). We can use break with label to terminate the nested loops.

The continue statement skips the current iteration of a for, while or do-while loop. We can use continue statement with label to skip the current iteration of outermost loop.

1. **What is this keyword?**

this keyword provides reference to the current object and it’s mostly used to make sure that object variables are used, not the local variables having same name.

//constructor

public Point(int x, int y) {

this.x = x;

this.y = y;

}

We can also use this keyword to invoke other constructors from a constructor.

public Rectangle() {

this(0, 0, 0, 0);

}

public Rectangle(int width, int height) {

this(0, 0, width, height);

}

public Rectangle(int x, int y, int width, int height) {

this.x = x;

this.y = y;

this.width = width;

this.height = height;

}

1. **What is default constructor?**

No argument constructor of a class is known as default constructor. When we don’t define any constructor for the class, java compiler automatically creates the default no-args constructor for the class. If there are other constructors defined, then compiler won’t create default constructor for us.

1. **Can we have try without catch block?**

Yes, we can have try-finally statement and hence avoiding catch block.

But checked exception should be caught or throws

1. **What is Garbage Collection?**

Garbage Collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects. In Java, process of deallocating memory is handled automatically by the garbage collector.

We can run the garbage collector with code Runtime.getRuntime().gc() or use utility method System.gc(). For a detailed analysis of Heap Memory and Garbage Collection, please read [Java Garbage Collection](http://www.journaldev.com/2856/java-jvm-memory-model-memory-management-in-java).

1. **What is Serialization and Deserialization?**

We can convert a Java object to an Stream that is called Serialization. Once an object is converted to Stream, it can be saved to file or send over the network or used in socket connections.

The object should implement Serializable interface and we can use java.io.ObjectOutputStream to write object to file or to any OutputStream object. Read more at [Java Serialization](http://www.journaldev.com/927/how-to-write-object-to-file-in-java).

The process of converting stream data created through serialization to Object is called deserialization. Read more at [Java Deserialization](http://www.journaldev.com/933/how-to-read-object-from-file-in-java).

1. **How to run a JAR file through command prompt?**

We can run a jar file using java command but it requires Main-Class entry in jar manifest file. Main-Class is the entry point of the jar and used by java command to execute the class. Learn more at [java jar file](http://www.journaldev.com/1344/how-to-run-jar-file-in-java).

1. **What is the use of System class?**

Java System Class is one of the core classes. One of the easiest way to log information for debugging is System.out.print() method.

System class is final so that we can’t subclass and override it’s behavior through inheritance. System class doesn’t provide any public constructors, so we can’t instantiate this class and that’s why all of it’s methods are static.

Some of the utility methods of System class are for array copy, get current time, reading environment variables. Read more at [Java System Class](http://www.journaldev.com/1847/java-system-java-lang-system-class).

1. **What is instanceof keyword?**

We can use instanceof keyword to check if an object belongs to a class or not. We should avoid it’s usage as much as possible. Sample usage is:

public static void main(String args[]){

Object str = new String("abc");

if(str instanceof String){

System.out.println("String value:"+str);

}

if(str instanceof Integer){

System.out.println("Integer value:"+str);

}

}

Since str is of type String at runtime, first if statement evaluates to true and second one to false.

1. **Can we use String with switch case?**

One of the Java 7 feature was improvement of switch case of allow Strings. So if you are using Java 7 or higher version, you can use String in switch-case statements. Read more at [Java switch-case String example](http://www.journaldev.com/588/java-switch-case-string).

1. **Java is Pass by Value or Pass by Reference?**

This is a very confusing question, we know that object variables contain reference to the Objects in heap space. When we invoke any method, a copy of these variables is passed and gets stored in the stack memory of the method. We can test any language whether it’s pass by reference or pass by value through a simple generic swap method, to learn more read [Java is Pass by Value and Not Pass by Reference](http://www.journaldev.com/3884/java-is-pass-by-value-and-not-pass-by-reference).

http://www.journaldev.com/3884/java-is-pass-by-value-and-not-pass-by-reference

1. **What is difference between Heap and Stack Memory?**

Major difference between Heap and Stack memory are as follows:

* 1. Heap memory is used by all the parts of the application whereas stack memory is used only by one thread of execution.
  2. Whenever an object is created, it’s always stored in the Heap space and stack memory contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space.
  3. Memory management in stack is done in LIFO manner whereas it’s more complex in Heap memory because it’s used globally.

For a detailed explanation with a sample program, read [Java Heap vs Stack Memory](http://www.journaldev.com/4098/java-heap-space-vs-stack-memory).

1. **Java Compiler is stored in JDK, JRE or JVM?**

The task of java compiler is to convert java program into bytecode, we have javac executable for that. So it must be stored in JDK, we don’t need it in JRE and JVM is just the specs.

1. **What will be the output of following programs?**
   1. **static method in class**
   2. package com.journaldev.util;
   3. public class Test {
   4. public static String toString(){
   5. System.out.println("Test toString called");
   6. return "";
   7. }
   9. public static void main(String args[]){
   10. System.out.println(toString());
   11. }

}

**Answer**: The code won’t compile because we can’t have an Object class method with static keyword. Note that Object class has toString() method. You will get compile time error as “This static method cannot hide the instance method from Object”. The reason is that static method belongs to class and since every class base is Object, we can’t have same method in instance as well as in class. You won’t get this error if you change the method name from toString() to something else that is not present in super class Object.

* 1. **static method invocation**
  2. package com.journaldev.util;
  3. public class Test {
  4. public static String foo(){
  5. System.out.println("Test foo called");
  6. return "";
  7. }
  9. public static void main(String args[]){
  10. Test obj = null;
  11. System.out.println(obj.foo());
  12. }

}

**Answer**: Well this is a strange situation. We all have seen NullPointerException when we invoke a method on object that is NULL. But here this program will work and prints “Test foo called”.

The reason for this is the java compiler code optimization. When the java code is compiled to produced byte code, it figures out that foo() is a static method and should be called using class. So it changes the method call obj.foo() to Test.foo() and hence no NullPointerException.

I must admit that it’s a very tricky question and if you are interviewing someone, this will blow his mind off.

# **Top 25 Most Frequently Asked Interview Core Java Interview Questions And Answers**

We are sharing 25 java interview questions , these questions are frequently asked by the recruiters.Java questions can be asked from any core java topic . So we try our best to provide you the java interview questions and answers for experienced which should be in your to do list before facing java questions in  technical interview  .  
 **Top 25  Java Interview Questions :**  
 **1. Which two method you need to implement for key Object in HashMap ?**

In order to use any object as Key in HashMap, it must implements equals and hashcode method in Java. Read [**How HashMap works in Java**](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html)  for detailed explanation on how equals and hashcode method is used to put and get object from HashMap.   
  
**2. What is immutable object? Can you write immutable object?**Immutable classes are Java classes whose objects can not be modified once created. Any modification in Immutable object result in new object. For example is String is immutable in Java. Mostly Immutable are also final in Java, in order to prevent sub class from overriding methods in Java which can compromise Immutability. You can achieve same functionality by making member as non final but private and not modifying them except in constructor.

**3. What is the difference between creating String as new() and literal?**

When we create string with new() Operator, it’s created in heap and not added into string pool while String created using literal are created in String pool itself which exists in PermGen area of heap.

String s = new String("Test");  
   
does not  put the object in String pool , we need to call String.intern() method which is used to put  them into String pool explicitly. its only when you create String object as String literal e.g. String s = "Test" Java automatically put that into String pool.

**4. What is**[**difference between StringBuffer and StringBuilder**](http://javahungry.blogspot.com/2013/06/difference-between-string-stringbuilder.html)**in Java ?**

Classic Java questions which some people thing tricky and some consider very easy. StringBuilder in Java is introduced in Java 5 and only difference between both of them is that Stringbuffer methods are synchronized while StringBuilder is non synchronized. See [StringBuilder vs StringBuffer](http://javahungry.blogspot.com/2013/06/difference-between-string-stringbuilder.html" \t "_blank) for more differences.

**5.  Write code to find the First non repeated character in the String  ?**

Another good Java interview question, This question is mainly asked by Amazon and equivalent companies. See [first non repeated character in the string : Amazon interview question](http://javahungry.blogspot.com/2013/12/first-non-repeated-character-in-string-java-program-code-example.html)

**6. What is the difference between ArrayList and Vector ?**  
This question is mostly used as a start up question in Technical interviews  on the topic of Collection framework . Answer is explained in detail here [Difference between ArrayList and Vector](http://javahungry.blogspot.com/2013/12/difference-between-arraylist-and-vector-in-java-collection-interview-question.html) .

**7. How do you handle error condition  while writing stored procedure or accessing stored procedure from java?**

This is one of the tough Java interview question and its open for all, my friend didn't know the answer so he didn't mind telling me. my take is that stored procedure should return error code if some operation fails but if stored procedure itself fail than catching SQLException is only choice.

**8. What is difference between Executor.submit() and Executer.execute() method ?**

*There is a difference when looking at exception handling. If your tasks throws an exception and if it was submitted with execute this exception will go to the uncaught exception handler (when you don't have provided one explicitly, the default one will just print the stack trace to System.err). If you submitted the task with submit any thrown exception, checked exception or not, is then part of the task's return status. For a task that was submitted with submit and that terminates with an exception, the Future.get will re-throw this exception, wrapped in an ExecutionException.*

**9. What is the difference between factory and abstract factory pattern?**

*Abstract Factory provides one more level of abstraction. Consider different factories each extended from an Abstract Factory and responsible for creation of different hierarchies of objects based on the type of factory. E.g. AbstractFactory extended by AutomobileFactory, UserFactory, RoleFactory etc. Each individual factory would be responsible for creation of objects in that genre.*

You can also refer What is Factory method design pattern in Java to know more details.

**10. What is Singleton? is it better to make whole method synchronized or only critical section synchronized ?**

[Singleton in Java is a class with just one instance in whole Java application](http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html), for example java.lang.Runtime is a Singleton class. Creating Singleton was tricky prior Java 4 but once Java 5 introduced Enum its very easy. see my article [How to create thread-safe Singleton in Java](http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html) for more details on writing Singleton using enum and double checked locking which is purpose of this Java interview question.

http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html

**11. Can you write critical section code for singleton?**

This core Java question is followup of previous question and expecting candidate to write Java singleton using double checked locking. Remember to use volatile variable to make Singleton thread-safe.

**12. Can you write code for**[**iterating**](http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html)**over hashmap in Java 4 and Java 5 ?**

Tricky one but he managed to write using while and for loop.  
  
**13. When do you override hashcode and equals() ?**  
Whenever necessary especially if you want to do equality check or want to use your object as key in HashMap.  
  
**14. What will be the problem if you don't override hashcode() method ?**  
You will not be able to recover your object from hash Map if that is used as key in HashMap.  
See here  [How HashMap works in Java](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html) for detailed explanation.  
  
**15. Is it better to synchronize critical section of getInstance() method or whole getInstance() method ?**  
Answer is critical section because if we lock whole method than every time some one call this method will have to wait even though we are not creating any object)  
  
**16. What is the difference when String is gets created using literal or new() operator ?**  
When we create string with new() its created in heap and not added into string pool while String created using literal are created in String pool itself which exists in Perm area of heap.  
  
**17. Does not overriding hashcode() method has any performance implication ?**  
This is a good question and open to all , as per my knowledge a poor hashcode function will result in frequent collision in HashMap which eventually increase time for adding an object into Hash Map.  
  
**18. What’s wrong using HashMap in multithreaded environment? When get() method go to infinite loop ?**  
Another good question. His answer was during concurrent access and re-sizing.  
  
**19.  What do you understand by thread-safety ? Why is it required ? And finally, how to achieve thread-safety in Java Applications ?**   
  
Java Memory Model defines the legal interaction of threads with the memory in a real computer system. In a way, it describes what behaviors are legal in multi-threaded code. It determines when a Thread can reliably see writes to variables made by other threads. It defines semantics for volatile, final & synchronized, that makes guarantee of visibility of memory operations across the Threads.  
  
Let's first discuss about Memory Barrier which are the base for our further discussions. There are two type of memory barrier instructions in JMM - read barriers and write barrier.  
  
A read barrier invalidates the local memory (cache, registers, etc) and then reads the contents from the main memory, so that changes made by other threads becomes visible to the current Thread.  
A write barrier flushes out the contents of the processor's local memory to the main memory, so that changes made by the current Thread becomes visible to the other threads.  
**JMM semantics for synchronized**  
When a thread acquires monitor of an object, by entering into a synchronized block of code, it performs a read barrier (invalidates the local memory and reads from the heap instead). Similarly exiting from a synchronized block as part of releasing the associated monitor, it performs a write barrier (flushes changes to the main memory)  
Thus modifications to a shared state using synchronized block by one Thread, is guaranteed to be visible to subsequent synchronized reads by other threads. This guarantee is provided by JMM in presence of synchronized code block.  
  
**JMM semantics for Volatile  fields**  
Read & write to volatile variables have same memory semantics as that of acquiring and releasing a monitor using synchronized code block. So the visibility of volatile field is guaranteed by the JMM. Moreover afterwards Java 1.5, volatile reads and writes are not reorderable with any other memory operations (volatile and non-volatile both). Thus when Thread A writes to a volatile variable V, and afterwards Thread B reads from variable V, any variable values that were visible to A at the time V was written are guaranteed now to be visible to B.

Let's try to understand the same using the following code

Data data = null;

volatile boolean flag = false;

Thread A

-------------

data = new Data();

flag = true; <-- writing to volatile will flush data as well as flag to main memory

Thread B

-------------

if(flag==true){ <-- as="" barrier="" data.="" flag="" font="" for="" from="" perform="" read="" reading="" volatile="" well="" will="">

use data; <!--- data is guaranteed to visible even though it is not declared volatile because of the JMM semantics of volatile flag.

}

**20.  What will happen if you call return statement or System.exit on try or catch block ? will finally block execute?**

This is a very *popular tricky Java question* and its tricky because many programmer think that finally block always executed. This question challenge that concept by putting return statement in 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 return statement in try block or catch block but finally block won't run if you call System.exit form try or catch.

**19. Can you override 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 private or static method in Java, if you create similar method with same return type and same method arguments that's called method hiding. 

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

This tricky Java questions is part of [How HashMap works in Java](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html), which is also a popular topic to create confusing and tricky question in Java. well if you put the same key again than it will replace the old mapping because HashMap doesn't allow duplicate keys.

**21. If a method throws NullPointerException in super class, can we override it with a method which throws RuntimeException?**

One more tricky Java questions from overloading and overriding concept. Answer is you can very well throw super class of RuntimeException in overridden method but you can not do same if its checked Exception.

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

public int compareTo(Object o){

   Employee emp = (Employee) emp;

   return this.id - o.id;

}

**23. 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 real tricky question for you. This Java question can be tricky even for experienced and senior programmer, who are not really exposed to deadlock and race conditions. Key point here is order, if you acquire resources in a particular order and release resources in reverse order you can prevent deadlock.

**24. What is difference between CyclicBarrier and CountDownLatch in Java**

Relatively newer Java tricky question, only been introduced form Java 5. 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.

**25. 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.

# Java Inner Class

AUGUST 14, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [6 COMMENTS](http://www.journaldev.com/996/java-inner-class#comments)

Java inner class is defined inside the body of another class. Java inner class can be declared private, public, protected, or with default access whereas an outer class can have only public or default access.

Java Nested classes are divided into two types.

### static nested class

If the nested class is static, then it’s called static nested class. Static nested classes can access only static members of the outer class. Static nested class is same as any other top-level class and is nested for only packaging convenience.

Static class object can be created with following statement.

OuterClass.StaticNestedClass nestedObject =

new OuterClass.StaticNestedClass();

## java inner class

[](http://cdn.journaldev.com/wp-content/uploads/2012/12/java-inner-class.jpg)  
Any non-static nested class is known as inner class in java. Java inner class is associated with the object of the class and they can access all the variables and methods of the outer class.

Since inner classes are associated with instance, we can’t have any static variables in them.

Object of java inner class are part of the outer class object and to create an instance of inner class, we first need to create instance of outer class.

Java inner class can be instantiated like this;

OuterClass outerObject = new OuterClass();

OuterClass.InnerClass innerObject = outerObject.new InnerClass();

There are two special kinds of java inner classes.

### local inner class

If a class is defined in a method body, it’s known as local inner class.

Since local inner class is not associated with Object, we can’t use private, public or protected access modifiers with it. The only allowed modifiers are abstract or final.

A local inner class can access all the members of the enclosing class and local final variables in the scope it’s defined.

Local inner class can be defined as:

public void print() {

//local inner class inside the method

class Logger {

String name;

}

//instantiate local inner class in the method to use

Logger logger = new Logger();

### anonymous inner class

A local inner class without name is known as anonymous inner class. An anonymous class is defined and instantiated in a single statement.

Anonymous inner class always extend a class or implement an interface. Since an anonymous class has no name, it is not possible to define a constructor for an anonymous class.

Anonymous inner classes are accessible only at the point where it is defined.  
It’s a bit hard to define how to create anonymous inner class, we will see it’s real time usage in test program below.

Here is a java class showing how to define java inner class, static nested class, local inner class and anonymous inner class.

OuterClass.java

## 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));

}

}

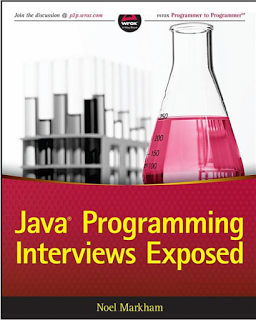
Answer: This question is tricky because unlike the [Integer](http://java67.blogspot.sg/2013/03/how-to-convert-java-string-to-int-or.html), where MIN\_VALUE is negative, both the MAX\_VALUE and MIN\_VALUE of the Double class are positive numbers. The Double.MIN\_VALUE is 2^(-1074), a double constant whose 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 throw ArithmeticExcpetion 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://aax-us-east.amazon-adsystem.com/x/c/QsXQicW0RGm8ybtX9H6mezIAAAFdnx5gLgEAAAFKAVswcrc/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868/ref=as_at?creativeASIN=1118722868&linkCode=w61&imprToken=fRjnsUK8dtUo6j66iNKZyw&slotNum=0&tag=javamysqlanta-20) for more of such popular questions,

[](http://aax-us-east.amazon-adsystem.com/x/c/QsXQicW0RGm8ybtX9H6mezIAAAFdnx5gLgEAAAFKAVswcrc/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868/ref=as_at?creativeASIN=1118722868&linkCode=w61&imprToken=fRjnsUK8dtUo6j66iNKZyw&slotNum=1&tag=javamysqlanta-20)

**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 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](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)

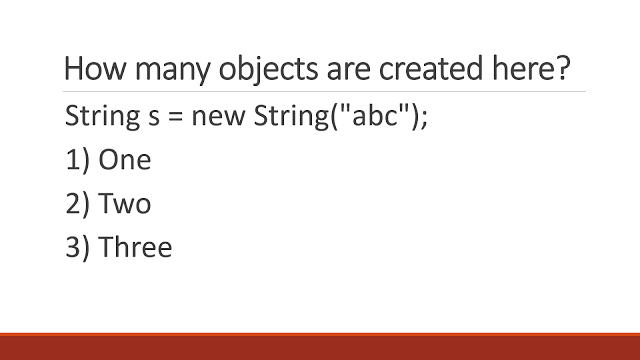
System.out.println("x=" + x);

**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?**

[](https://4.bp.blogspot.com/-kitlSuknjow/V3xrNGYnOLI/AAAAAAAAGi0/8mkbr22IlngwqfdKzZgB_SRWkqNe0Ar_wCLcB/s1600/How%2Bmany%2Bobject%2Bis%2Bcreated%2Bhere%2BString%2Btricky%2Bquestion%2Bjava.png)

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?

public String getDescription(Object obj){

return obj.toString;

}

public String getDescription(String obj){

return obj;

}

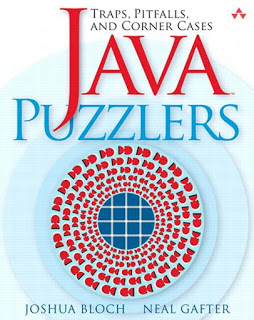
and

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**](http://aax-us-east.amazon-adsystem.com/x/c/QsXQicW0RGm8ybtX9H6mezIAAAFdnx5gLgEAAAFKAVswcrc/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/dp/032133678X/ref=as_at?creativeASIN=032133678X&linkCode=w61&imprToken=fRjnsUK8dtUo6j66iNKZyw&slotNum=2&tag=javamysqlanta-20), I am sure you ill find them really challenging to solve, I certainly did.

[](http://aax-us-east.amazon-adsystem.com/x/c/QsXQicW0RGm8ybtX9H6mezIAAAFdnx5gLgEAAAFKAVswcrc/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/dp/032133678X/ref=as_at?creativeASIN=032133678X&linkCode=w61&imprToken=fRjnsUK8dtUo6j66iNKZyw&slotNum=3&tag=javamysqlanta-20)

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

Read more: <http://www.java67.com/2012/09/top-10-tricky-java-interview-questions-answers.html#ixzz4oXTXrklk>