Table of Contents

* [Object Oriented Programming (OOP)](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#1)
* [General Questions about Java](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#2)
* [Java Threads](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#3)
* [Java Collections](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#4)
* [Garbage Collectors](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#5)
* [Exception Handling](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#6)
* [Java Applets](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#7)
* [Swing](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#8)
* [JDBC](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#9)
* [Remote Method Invocation (RMI)](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#10)
* [Servlets](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#11)
* [JSP](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#12)

Object Oriented Programming (OOP)

Java is a computer programming language that is concurrent, class-based and object-oriented. The advantages of object oriented software development are shown below:

* Modular development of code, which leads to easy maintenance and modification.
* Reusability of code.
* Improved reliability and flexibility of code.
* Increased understanding of code.

Object-oriented programming contains many significant features, such as **encapsulation**, **inheritance**, **polymorphism**and [**abstraction**](http://www.javacodegeeks.com/2014/07/abstraction-in-java.html). We analyze each feature separately in the following sections.

Encapsulation

Encapsulation provides objects with the ability to hide their internal characteristics and behavior. Each object provides a number of methods, which can be accessed by other objects and change its internal data. In Java, there are three access modifiers: public, private and protected. Each modifier imposes different access rights to other classes, either in the same or in external packages. Some of the advantages of using encapsulation are listed below:

* The internal state of every objected is protected by hiding its attributes.
* It increases usability and maintenance of code, because the behavior of an object can be independently changed or extended.
* It improves modularity by preventing objects to interact with each other, in an undesired way.

You can refer to our tutorial [here](http://examples.javacodegeeks.com/java-basics/encapsulation-in-java/) for more details and examples on encapsulation.

Polymorphism

Polymorphism is the ability of programming languages to present the same interface for differing underlying data types. A polymorphic type is a type whose operations can also be applied to values of some other type.

Inheritance

Inheritance provides an object with the ability to acquire the fields and methods of another class, called base class. Inheritance provides re-usability of code and can be used to add additional features to an existing class, without modifying it.

Abstraction

[Abstraction](http://www.javacodegeeks.com/2014/04/why-abstraction-is-really-important.html) is the process of separating ideas from specific instances and thus, develop classes in terms of their own functionality, instead of their implementation details. Java supports the creation and existence of abstract classes that expose interfaces, without including the actual implementation of all methods. The abstraction technique aims to separate the implementation details of a class from its behavior.

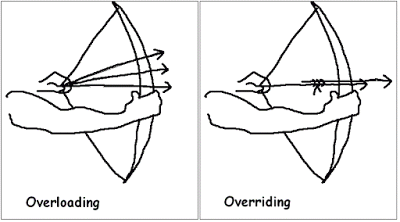
Differences between Abstraction and Encapsulation

Abstraction and encapsulation are complementary concepts. On the one hand, abstraction focuses on the behavior of an object. On the other hand, encapsulation focuses on the implementation of an object’s behavior. Encapsulation is usually achieved by hiding information about the internal state of an object and thus, can be seen as a strategy used in order to provide abstraction.

## Overloading and Overriding Interview Questions

Ok, let's start on questions. Here is my list of 17 method overloading and overriding interview question which covers basics as well as some tricky part of overloading and overriding. You can also write some code to test the concept and see when compiler gives error and which method is get called at run-time.

**1) What is method overloading in Java?**  
If you have two method which does same thing its better they have same name, but two method cannot have same name until you overload them. So overloading is a process of declaring two methods with same name but different method signature e.g. System.out which is object of PrintStream class has several println() method to print different data types e.g. byte, short, int, char, float and double. All of them are called overloaded method. Overloaded method calls are resolved during compile time in Java and they must have different method signatures. See [here](http://java67.blogspot.sg/2012/08/what-is-method-overloading-in-java-example.html) to learn more about method overloading in Java.  
  
  
**2) What is method overriding in Java?**  
Method overriding is another way to define method with same name but different code but it must be in sub class. Overriding is based upon run-time Polymorphism as method calls are resolved at run-time depending upon actual object.  For example if a variable of type Parent holds an object of Child class then method invoked will be from child class and not parent class, provides its overridden. In order to override a method, you must follow rules of method overriding which means declaring method with same signature in sub class. See [here](http://java67.blogspot.sg/2012/08/what-is-method-overriding-in-java-example-tutorial.html) to learn more about method overriding in Java.  
  
  
**3) What is method hiding in Java?**  
static method cannot be overriding in Java because their method calls are resolved at compile time but it didn't prevent you from declaring method with same name in sub class. In this case we say that method in sub class has hided static method from parent class. If you have a case where variable of Parent class is pointing to object of Child class then also static method from Parent class is called because overloading is resolved at compile time. See [here](http://java67.blogspot.sg/2012/08/can-we-override-static-method-in-java.html) to learn more about method hiding in Java.  
  
  
**4) What are rules of overloading a method in Java?**  
One and only rule of method overloading in Java is that method signature of all overloaded method must be different. Method signature is changed by changing either number of method arguments, or type of method arguments e.g. System.out.println() method is overloaded to accept different primitive types like int, short, byte, float etc. They all accept just one argument but their type is different. You can also change method signature by changing order of method argument but that often leads to ambiguous code so better to be avoided. See [here](http://java67.blogspot.sg/2012/09/what-is-rules-of-overloading-and-overriding-in-java.html) for full list of rules.  
  
  
**5) Difference between method overloading and overriding?**  
Fundamental difference between overloading and overriding is that former took place during compile time while later took place during run-time. Due to this reason, its only possible to overload virtual methods in Java. You cannot overload methods which are resolved during compile time e.g. private, static and final method cannot be overridden in Java. Also, rules of method overloading and overriding are different, for example in order to overload a method its method signature must be different but for overriding method it must be same. See this image to learn more difference between overriding and overloading in Java.

[](http://java67.blogspot.sg/2012/09/difference-between-overloading-vs-overriding-in-java.html)

**6) Can we overload static method in Java?**  
Yes, its possible to overload static method in Java. You can declare as many static method with same name as you want until all of them have different method signature. Remember, return type is not part of method signature, so they must have either different number of arguments, or different type of argument. There is a third option also which changes order of argument but I suggest not to do that because it often result in ambiguous method call. It's very important to follow [these best practices](http://javarevisited.blogspot.sg/2013/01/java-best-practices-method-overloading-constructor.html) while overloading a static method in Java.  
  
  
**7) Can we override static method in Java?**  
No, you cannot override static method in Java because they are resolved and bonded during compile time. Since overriding is a run-time activity and if a method call is already resolved at compile time then it will not take place and that's why its not possible to override static method in Java. But, you can define another static method of same signature in sub class, this is known as method hiding. Actual method called will depends upon the type of class and not on type of object as its the case with overriding. See [here](http://javarevisited.blogspot.sg/2013/03/can-we-overload-and-override-static-method-java.html) to learn more about why you cannot override static method in Java.  
  
  
**8) Can you prevent overriding a method without using final modifier?**  
Yes, there are some funky ways to prevent method overriding in Java. Though final modifier is only for that purpose you can use private keyword to prevent method overriding. How? If you remember correctly, in order to override a method, the class must be extensible. If you make the constructor of parent class private then its not possible to extend that class because its constructor will not be accessible in sub class, which is automatically invoked by sub class constructor, hence its not possible to override any method from that class. This technique is used in Singleton design pattern, where constructor is purposefully made private and a static getInstance() method is provided to access singleton instance. See [here](http://javarevisited.blogspot.sg/2015/04/3-ways-to-prevent-method-overriding-in.html) to learn more techniques to prevent method overriding in Java.  
  
  
**9) Can we override a private method in Java?**  
No, you cannot override private method in Java. Since private methods are not visible outside the class, they are not available in sub-class hence they cannot be overridden. By the way, how about overriding a private method inside an Inner class? Is it possible? See [here](http://java67.blogspot.sg/2013/08/can-we-override-private-method-in-java-inner-class.html) to learn more why you cannot override private method in Java.  
  
  
**10) What is co-variant method overriding?**  
One of the rule of method overriding is that return type of overriding method must be same as overridden method but this restriction is relaxed little bit from Java 1.5 and now overridden method can return sub class of return type of original method. This relaxation is known as co-variant method overriding and it allows you to remove casting at client end. One of the best example of this comes [when you override clone() method](http://javarevisited.blogspot.sg/2015/01/java-clone-tutorial-part-2-overriding-with-mutable-field-example.html). Original Object.clone()method returns Object which needs to cast, but with co-variant method overriding you can directly return relevant type e.g. Date class returns object of java.util.Date instead of java.lang.Object. See here to learn more about co-variant method overriding in Java.  
  
  
**11) Can we change argument list of overridden method?**  
No, you cannot change the argument list of overridden method in Java. An overriding method must have same signature as original method. Only return type can be changed that to only to sub type of return type of original method.  
  
  
**12) Can we change return type of method in subclass while overriding?**  
No, you cannot change the return type of method during overriding. It would be violation of rules of overriding. Though from Java 5 onward you can replace the return type with sub type e.g. if original method has return type as java.lang.Object then you can change return type of overridden method as any type e.g. clone() method. This is also known as [co-variant method overriding](http://javarevisited.blogspot.sg/2014/03/covariant-method-overriding-of-java-5.html) in Java.  
  
  
**13) Can we override a method which throws run-time exception without throws clause?**  
Yes, you can. There is no restriction on throwing RuntimeException from overriding method. So if your original method throws NullPointerException than its not necessary to throw NPE from overriding method as well.  
  
  
**14) How do you call super class version of an overriding method in sub class?**  
You can call it using super keyword. For example if you have a method calculate() in both parent and child class then from child class you can invoke parent class method calculate() as super.calculate(). It's very similar to calling super class constructor from sub class as shown [here](http://java67.blogspot.sg/2012/12/how-constructor-chaining-works-in-java.html).  
  
  
**15) What are rules of method overriding in Java?**  
Some rules of method overriding are following :

* Overriding method cannot throw higher exception than overridden one, but that's only true for checked exception.
* Overriding method cannot restrict access of overridden method e.g. if original method is public then overriding method must be public, but it can expand access e.g. if original method is protected than overriding method can be protected or public.

See [here](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html) for full list of rules of method overriding in Java.  
  
  
**16) Can we override a non-static method as static in Java?**  
No, its not possible to define a non-static method of same name as static method in parent class, its compile time error in Java. See here to learn more about [overriding static method in Java](http://java67.blogspot.sg/2012/11/what-is-static-class-variable-method.html).  
  
  
**17) Can we override constructor in Java?**  
No, you cannot override constructor in Java because they are not inherited. Remember, we are talking about overriding here not overloading, you can overload construct but you cannot override them. Overriding always happens at child class and since constructor are not inherited and their name is always same as the class name its not possible to override them in Java, to learn more about constructor see [here](http://java67.blogspot.sg/2014/09/Why-constructor-is-important-in-java-example.html)  
  
  
**18) Can we override final method in Java?**  
No, you cannot override final method in Java. Trying to override final method in subclass will result in compile time error. Actually making a method final is signal to all developer that this method is not for inheritance and it should be use in its present form. You generally make a method final due to security reasons, to learn more see [here](http://javarevisited.blogspot.sg/2013/12/when-to-make-method-final-in-java.html).  
  
  
**19) Can you overload or override main method in Java?**  
Since main() is a static method in Java, it follows rules associated with static method, which means you can overload main method but you cannot override it. By the way, even if you overload a main method, JVM will always call the standard public static void main(String args[]) method to start your program, if you want to call your overloaded method you need to do it explicitly in your code as shown [here](http://java67.blogspot.sg/2015/06/can-you-overload-or-override-main-in-java.html).  
  
  
That's all about some **Java interview questions from method overloading and overriding concept**. I must admit its a tricky concept to master. I have seen even experienced developer struggling to solve quiz and multiple choice questions based upon overloading and overriding. So, if you want to do well on Java Interviews as well as wants to write good, powerful and flexible code using object oriented programming technique, spare some time to understand overloading and overriding in Java. It will help you to leverage Polymorphism better.  
  
  
Read more: <http://www.java67.com/2015/08/top-10-method-overloading-overriding-interview-questions-answers-java.html#ixzz4fFSx31cy>

General Questions about Java

**1. What is JVM ? Why is Java called the “Platform Independent Programming Language” ?** A Java virtual machine (JVM) is a process [virtual machine](http://www.javacodegeeks.com/2013/12/part-1-of-3-synopsis-of-articles-videos-on-performance-tuning-jvm-gc-in-java-mechanical-sympathy-et-al.html) that can execute Java [bytecode](http://www.javacodegeeks.com/2013/12/mastering-java-bytecode.html). Each Java source file is compiled into a bytecode file, which is executed by the JVM. Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

**2. What is the Difference between JDK and JRE ?** The Java Runtime Environment (JRE) is basically the Java Virtual Machine (JVM) where your Java programs are being executed. It also includes browser plugins for applet execution. The Java Development Kit (JDK) is the full featured Software Development Kit for Java, including the JRE, the compilers and tools (like [JavaDoc](http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html), and [Java Debugger](http://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html)), in order for a user to develop, compile and execute Java applications.

**3. What does the “static” keyword mean ? Can you override private or static method in Java ?** The static keyword denotes that a member variable or method can be accessed, without requiring an instantiation of the class to which it belongs. A user cannot override [static methods in Java](http://www.javacodegeeks.com/2012/05/java-static-methods-can-be-code-smell.html), because method overriding is based upon dynamic binding at runtime and static methods are statically binded at compile time. A static method is not associated with any instance of a class so the concept is not applicable.

**4. Can you access non static variable in static context ?** A static variable in Java belongs to its class and its value remains the same for all its instances. A static variable is initialized when the class is loaded by the JVM. If your code tries to access a non-static variable, without any instance, the compiler will complain, because those variables are not created yet and they are not associated with any instance.

**5. What are the Data Types supported by Java ? What is Autoboxing and Unboxing ?** The eight primitive data types supported by the Java programming language are:

* byte
* short
* int
* long
* float
* double
* boolean
* char

Autoboxing is the [automatic conversion made by the Java compiler](http://www.javacodegeeks.com/2013/07/java-generics-tutorial-example-class-interface-methods-wildcards-and-much-more.html) between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an [Integer](http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html?is-external=true), a double to a [Double](http://docs.oracle.com/javase/7/docs/api/java/lang/Double.html), and so on. If the conversion goes the other way, this operation is called unboxing.

**6. What is Function Overriding and Overloading in Java ?** Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

**7. What is a Constructor, Constructor Overloading in Java and Copy-Constructor ?** A constructor gets invoked when a new object is created. Every class [has a constructor](http://www.javacodegeeks.com/2014/01/which-is-better-option-cloning-or-copy-constructors.html). In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class. The constructor overloading is similar to method overloading in Java. Different constructors can be created for a single class. Each constructor must have its own unique parameter list. Finally, Java does support copy constructors like C++, but the difference lies in the fact that Java doesn’t create a default copy constructor if you don’t write your own.

**8. Does Java support multiple inheritance ?** No, Java does not support multiple inheritance. Each class is able to extend only on one class, but is able to implement more than one interfaces.

**9. What is the difference between an Interface and an Abstract class ?** Java provides and supports the creation both of [abstract classes](http://examples.javacodegeeks.com/java-basics/java-abstract-class-example/) and interfaces. Both implementations share some common characteristics, but they differ in the following features:

* All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
* A class may implement a number of Interfaces, but can extend only one abstract class.
* In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
* Abstract classes can implement interfaces without even providing the implementation of interface methods.
* Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
* Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
* An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

Also check out the [Abstract class and Interface differences for JDK 8](http://www.javacodegeeks.com/2014/04/abstract-class-versus-interface-in-the-jdk-8-era.html).

**10. What are pass by reference and pass by value ?** When an object is passed by value, this means that a copy of the object is passed. Thus, even if changes are made to that object, it doesn’t affect the original value. When an object is passed by reference, this means that the actual object is not passed, rather a reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.

Java Threads

**11. What is the difference between processes and threads ?** A process is an execution of a program, while a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is a single execution sequence within a process. A process can contain multiple threads. A [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is sometimes called a lightweight process.

**12. Explain different ways of creating a thread. Which one would you prefer and why ?** There are three ways that can be used in order for a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) to be created:

* A class may extend the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class.
* A class may implement the [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface.
* An application can use the [Executor](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html) framework, in order to create a thread pool.

The [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface is preferred, as it does not require an object to inherit the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class. In case your application design requires multiple inheritance, only interfaces can help you. Also, the thread pool is very efficient and can be implemented and used very easily.

**13. Explain the available thread states in a high-level.** During its execution, a thread can reside in one of the following [states](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html):

* [**NEW**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#NEW): The thread becomes ready to run, but does not necessarily start running immediately.
* [**RUNNABLE**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#RUNNABLE): The Java Virtual Machine (JVM) is actively executing the thread’s code.
* [**BLOCKED**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#BLOCKED): The thread is in a blocked state while waiting for a monitor lock.
* [**WAITING**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#WAITING): The thread waits for another thread to perform a particular action.
* [**TIMED\_WAITING**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#TIMED_WAITING): The thread waits for another thread to perform a particular action up to a specified waiting time.
* [**TERMINATED**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#TERMINATED): The thread has finished its execution.

**14. What is the difference between a synchronized method and a synchronized block ?** In Java programming, each object has a lock. A thread can acquire the lock for an object by using the synchronized keyword. The synchronized keyword can be applied in a method level (coarse grained lock) or block level of code (fine grained lock).

**15. How does thread synchronization occurs inside a monitor ? What levels of synchronization can you apply ?** The JVM uses locks in conjunction with monitors. A monitor is basically a guardian that watches over a sequence of synchronized code and ensuring that only one thread at a time executes a synchronized piece of code. Each monitor is associated with an object reference. The thread is not allowed to execute the code until it obtains the lock.

**16. What’s a deadlock ?** A condition that occurs when [two processes are waiting for each other to complete](http://www.javacodegeeks.com/2013/01/java-deadlock-example-how-to-analyze-deadlock-situation.html), before proceeding. The result is that both processes wait endlessly.

**17. How do you ensure that N threads can access N resources without deadlock ?** A very simple way to avoid deadlock while using N threads is to impose an ordering on the locks and force each thread to follow that ordering. Thus, if all threads lock and unlock the mutexes in the same order, no deadlocks can arise.

Java Collections

**18. What are the basic interfaces of Java Collections Framework ?** [Java Collections Framework](http://docs.oracle.com/javase/7/docs/technotes/guides/collections/overview.html) provides a well designed set of interfaces and classes that support operations on a collections of objects. The most basic interfaces that reside in the Java Collections Framework are:

* [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html), which represents a group of objects known as its elements.
* [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html), which is a collection that cannot contain duplicate elements.
* [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html), which is an ordered collection and can contain duplicate elements.
* [Map](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html), which is an object that maps keys to values and cannot contain duplicate keys.

**19. Why Collection doesn’t extend Cloneable and Serializable interfaces ?** The [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) interface specifies groups of objects known as elements. Each concrete implementation of a [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) can choose its own way of how to maintain and order its elements. Some collections allow duplicate keys, while some other collections don’t. The semantics and the implications of either cloning or serialization come into play when dealing with actual implementations. Thus, the concrete implementations of collections should decide how they can be cloned or serialized.

**20. What is an Iterator ?** The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface provides a number of methods that are able to iterate over any [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html). Each Java [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) contains the [iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html#iterator%28%29) method that returns an [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) instance. Iterators are [capable of removing elements from the underlying collection](http://www.javacodegeeks.com/2011/05/avoid-concurrentmodificationexception.html) during the iteration. **21. What differences exist between Iterator and ListIterator ?** The differences of these elements are listed below:

* An [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can be used to traverse the [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html) and [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) collections, while the [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can be used to iterate only over [Lists](http://docs.oracle.com/javase/7/docs/api/java/util/List.html).
* The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can traverse a collection only in forward direction, while the [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can traverse a [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) in both directions.
* The [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) implements the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface and contains extra functionality, such as adding an element, replacing an element, getting the index position for previous and next elements, etc.

**22. What is difference between fail-fast and fail-safe ?** The [Iterator's](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) fail-safe property works with the clone of the underlying collection and thus, it is not affected by any modification in the collection. All the collection classes in java.util package are fail-fast, while the collection classes in java.util.concurrent are fail-safe. Fail-fast iterators throw a [ConcurrentModificationException](http://examples.javacodegeeks.com/java-basics/exceptions/java-util-concurrentmodificationexception-how-to-handle-concurrent-modification-exception/), while fail-safe iterator never throws such an exception.

**23. How HashMap works in Java ?** A [HashMap in Java stores key-value pairs](http://www.javacodegeeks.com/2014/03/how-hashmap-works-in-java.html). The [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) requires a hash function and uses [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) are its capacity, its load factor and the threshold resizing.

**24. What is the importance of hashCode() and equals() methods ?** In Java, a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) uses the [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and [equals](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals%28java.lang.Object%29) methods to determine the index of the key-value pair and to detect duplicates. More specifically, the [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29)method is used in order to determine where the specified key will be stored. Since different keys may produce the same hash value, the [equals](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals%28java.lang.Object%29) method is used, in order to determine whether the specified key actually exists in the collection or not. Therefore, the implementation of both methods is crucial to the accuracy and efficiency of the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html).

**25. What differences exist between HashMap and Hashtable ?** Both the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) and [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) classes implement the Map interface and thus, have very similar characteristics. However, they differ in the following features:

* A [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) allows the existence of null keys and values, while a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) doesn’t allow neither null keys, nor null values.
* A [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) is synchronized, while a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is not. Thus, [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is preferred in single-threaded environments, while a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) is suitable for multi-threaded environments.
* A [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) provides its set of keys and a Java application can iterate over them. Thus, a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is fail-fast. On the other hand, a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) provides an [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) of its keys.
* The [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) class is considered to be a legacy class.

**26. What is difference between Array and ArrayList ? When will you use Array over ArrayList ?** The [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html)and [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) classes differ on the following features:

* [Arrays](http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html) can contain primitive or objects, while an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) can contain only objects.
* [Arrays](http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html) have fixed size, while an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) is dynamic.
* An [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html)provides more methods and features, such as [addAll](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#addAll(java.util.Collection)), [removeAll](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#removeAll(java.util.Collection)), [iterator](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#iterator()), etc.
* For a list of primitive data types, the collections use autoboxing to reduce the coding effort. However, this approach makes them slower when working on fixed size primitive data types.

**27. What is difference between ArrayList and LinkedList ?** Both the [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) and [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) classes implement the List interface, but they differ on the following features:

* An [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) is an index based data structure backed by an [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html). It provides random access to its elements with a performance equal to O(1). On the other hand, a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) stores its data as list of elements and every element is linked to its previous and next element. In this case, the search operation for an element has execution time equal to O(n).
* The Insertion, addition and removal operations of an element are faster in a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) compared to an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), because there is no need of resizing an array or updating the index when an element is added in some arbitrary position inside the collection.
* A [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) consumes more memory than an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), because every node in a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) stores two references, one for its previous element and one for its next element.

Check also our article [ArrayList vs. LinkedList](http://www.javacodegeeks.com/2013/12/arraylist-vs-linkedlist.html).

**28. What is Comparable and Comparator interface ? List their differences.** Java provides the [Comparable](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html)interface, which contains only one method, called [compareTo](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html#compareTo(T)). This method compares two objects, in order to impose an order between them. Specifically, it returns a negative integer, zero, or a positive integer to indicate that the input object is less than, equal or greater than the existing object. Java provides the [Comparator](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html) interface, which contains two methods, called [compare](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#compare(T,%20T)) and [equals](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#equals(java.lang.Object)). The first method compares its two input arguments and imposes an order between them. It returns a negative integer, zero, or a positive integer to indicate that the first argument is less than, equal to, or greater than the second. The second method requires an object as a parameter and aims to decide whether the input object is equal to the comparator. The method returns true, only if the specified object is also a comparator and it imposes the same ordering as the comparator.

**29. What is Java Priority Queue ?** The [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is an unbounded queue, based on a priority heap and its elements are ordered in their natural order. At the time of its creation, we can provide a Comparator that is responsible for ordering the elements of the [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html). A [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) doesn’t allow [null values](http://examples.javacodegeeks.com/java-basics/exceptions/java-lang-nullpointerexception-how-to-handle-null-pointer-exception/), those objects that doesn’t provide natural ordering, or those objects that don’t have any comparator associated with them. Finally, the Java [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is not thread-safe and it requires O(log(n)) time for its enqueing and dequeing operations.

**30. What do you know about the big-O notation and can you give some examples with respect to different data structures ?** The [Big-O notation](http://www.javacodegeeks.com/2011/04/simple-big-o-notation-post.html) simply describes how well an algorithm scales or performs in the worst case scenario as the number of elements in a data structure increases. The Big-O notation can also be used to describe other behavior such as memory consumption. Since the collection classes are actually data structures, we usually use the Big-O notation to chose the best implementation to use, based on time, memory and performance. Big-O notation can give a good indication about performance for large amounts of data.

**31. What is the tradeoff between using an unordered array versus an ordered array ?** The major advantage of an ordered array is that the search times have time complexity of O(log n), compared to that of an unordered array, which is O (n). The disadvantage of an ordered array is that the insertion operation has a time complexity of O(n), because the elements with higher values must be moved to make room for the new element. Instead, the insertion operation for an unordered array takes constant time of O(1).

**32. What are some of the best practices relating to the Java Collection framework ?**

* Choosing the right type of the collection to use, based on the application’s needs, is very crucial for its performance. For example if the size of the elements is fixed and know a priori, we shall use an [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html), instead of an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html).
* Some collection classes allow us to specify their initial capacity. Thus, if we have an estimation on the number of elements that will be stored, we can use it to avoid rehashing or resizing.
* Always use Generics for type-safety, readability, and robustness. Also, by using Generics you avoid the [ClassCastException](http://docs.oracle.com/javase/7/docs/api/java/lang/ClassCastException.html) during runtime.
* Use immutable classes provided by the Java Development Kit (JDK) as a key in a Map, in order to avoid the implementation of the [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode%28%29) and equals methods for our custom class.
* Program in terms of interface not implementation.
* Return zero-length collections or arrays as opposed to returning a null in case the underlying collection is actually empty.

**33. What’s the difference between Enumeration and Iterator interfaces ?** [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) is twice as fast as compared to an Iterator and uses very less memory. However, the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) is much safer compared to [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html), because other threads are not able to modify the collection object that is currently traversed by the iterator. Also, [Iterators](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html)allow the caller to remove elements from the underlying collection, something which is not possible with [Enumerations](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html).

**34. What is the difference between HashSet and TreeSet ?** The [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) is Implemented using a hash table and thus, its elements are not ordered. The add, remove, and contains methods of a [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) have constant time complexity O(1). On the other hand, a [TreeSet](http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) is implemented using a tree structure. The elements in a [TreeSet](http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) are sorted, and thus, the add, remove, and contains methods have time complexity of O(logn).

## Java Collection interview Questions Answers

Here is list of my favorite*,*[*frequently asked Questions from Java collection framework*](http://javarevisited.blogspot.sg/2011/11/collection-interview-questions-answers.html). Almost all of these questions have appeared in Java interview at various level ranging from Junior to Senior software engineer level at different Companies e.g. Capegemini, Tech Mahindra, TCS , Satyam and CTS.

**What is Difference between Hashtable and HashMap in Java?**

This Java collection interview questions is I guess most popular one. Most of Java programmer who has at least 2 years of experience has seen this question on core Java or J2EE interview. Well there are many difference between them but most important is [thread-safety](http://javarevisited.blogspot.sg/2012/01/how-to-write-thread-safe-code-in-java.html), HashMap is not thread-safe while Hashtable is thread-safe collection. See [Hashtable vs HashMap in Java](http://java67.blogspot.sg/2012/08/5-difference-between-hashtable-hashmap-Java-collection.html) for more differences between HashMap and Hashtable in Java.

**What is difference between Hashtable and ConcurrentHashMap in Java?**

Another frequently asked Java collection interview question post Java 5 world which introduced Concurrent Collection classes like ConcurrentHashMap and CopyOnWriteArrayList along with Concurrency utilities e.g. [CyclicBarrier](http://javarevisited.blogspot.sg/2012/07/cyclicbarrier-example-java-5-concurrency-tutorial.html) and [CountDownLatch](http://javarevisited.blogspot.sg/2012/07/countdownlatch-example-in-java.html). Well both Hashtable and ConcurrentHashMap are thread-safe here but later provides more scalability than former. See [Difference between ConcurrentHashMap and Hashtable in Java](http://javarevisited.blogspot.sg/2011/04/difference-between-concurrenthashmap.html) for answer of this Java collection interview question.

**What is Difference between Iterator and Enumeration in Java?**

One of the classic interview Questions asked on Java collection framework, This is pretty old and programmer who has been working in Java for 4 to 6 years must have seen this question before. Well [Iterator and ListIterator in Java](http://javarevisited.blogspot.sg/2011/10/java-iterator-tutorial-example-list.html) is a new way to iterator collection in Java and provides ability to remove object while traversing while Enumeration doesn't allow you to remove object. See [Iterator vs Enumeration in Java](http://javarevisited.blogspot.sg/2010/10/what-is-difference-between-enumeration.html) for more differences between both of them.

**What is Difference between fail-safe and fail-fast Iterator in Java?**

This is relatively new Java collection interview question because concept of fail-safe iterator is come along with ConcurrentHashMap and CopyOnWriteArrayList. See [Difference between fail-safe and fail-fast Iterator in Java](http://javarevisited.blogspot.sg/2012/02/fail-safe-vs-fail-fast-iterator-in-java.html) for answer of this Java collection question.

**How HashMap works internally in Java?**

One of the most frequently asked Java interview question to experience Java programmer of 4 to 5 years of experience. I have seen this question on big companies like Morgan Stanley, JP Morgan, Nomura and other banks e.g. Barclays capital. See [How HashMap works internally in Java](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html) for detailed answer of this Java collection interview question.

**Can you write code to traverse Map in Java on 4 ways?**

Another Java collection question which appear as part of [Java Coding interview question](http://java67.blogspot.sg/2012/08/10-java-coding-interview-questions-and.html) and appeared in many interviews. As you know there are multiple ways to traverse or iterate Map in Java e.g. for loop, while loop using Iterator etc. [4 ways to iterator Map in Java](http://javarevisited.blogspot.sg/2011/12/how-to-traverse-or-loop-hashmap-in-java.html) has detailed explanation and sample code which is sufficient to answer this Java collection framework interview question.

**What is difference between Vector and ArrayList in Java?**

Along with [Difference between HashMap and hashtable](http://javarevisited.blogspot.sg/2010/10/difference-between-hashmap-and.html), this Java collection interview question is probably second in the list of frequently asked question on Java collection framework. Both ArrayList and Vector implements List interface from Java 4 but they have differences including synchronization, See [difference between Vector and ArrayList in Java](http://javarevisited.blogspot.sg/2011/09/difference-vector-vs-arraylist-in-java.html) for complete answer of this collection interview question in Java.

**What is difference between ArrayList and LinkedList in Java?**

A follow-up question which is asked in response to previous Java collection interview question. Here also both LinkedList and ArrayList are List implementation but there internal data-structure is different, one is derived from Array while other is derived from LinkedList. See [LinkedList vs ArrayList in Java](http://javarevisited.blogspot.sg/2012/02/difference-between-linkedlist-vs.html) to answer this Java Collection interview question.

**What is difference between List and Set in Java ?**

List vs Set is one of the most important concept to understand in Java Collection framework and this *Java collection interview question* focus on that. Most important difference between them is that List allows duplicates and maintain insertion order while Set doesn't allow duplicates and doesn't maintain any order. See [Difference between Set and List in Java](http://javarevisited.blogspot.sg/2012/04/difference-between-list-and-set-in-java.html) to see more differences between them

**How do you find if ArrayList contains duplicates or not ?**

Since List allows duplicates this becomes a followup question of earlier Java collection framework interview question. See [How to check if ArrayList contains duplicates or not](http://javarevisited.blogspot.sg/2012/02/how-to-check-or-detect-duplicate.html) for answer of this Java collection question.

Read more: <http://www.java67.com/2012/09/java-collection-interview-questions.html#ixzz4fFTQU2jM>

## Java Array Concept Interview Questions

Here are some interview questions based upon array data structure in Java. You need to have good knowledge of how array is implemented and work in Java to answer these questions. Since array is one of the most used data structure, its expected from programmers of all levels (including beginners and experienced) to have good grasp of array concepts. These questions are mostly asked in [telephonic round of Java interviews](http://javarevisited.blogspot.sg/2014/02/top-30-java-phone-interview-questions.html). Your answer to these question must be focused and to the point, without any added syntactic sugar.  
  
  
**Question 1 : Can you change size of array once created?** [[answer](http://java67.blogspot.sg/2014/08/10-points-about-array-in-java.html)]  
No, you cannot change the size of array once created. If you need dynamic array, consider using ArrayList class, which can resize itself.  
  
  
**Question 2 : Can you store String in an array of Integer in Java? compile time error or runtime exception?**[[answer](http://javarevisited.blogspot.co.uk/2013/11/java-array-101-for-programmers-and.html)]  
This is a tricky question. Answer is both yes and no. You cannot store an String in an array of primitive int, it will result in compile time error as shown below, but if you create an array of Object and assign String[] to it and then try to store Integer object on it. Compiler won't be able to detect that and it will throw ArrayStoreExcpetion at runtime.

int[] primes = new int[10];

primes[0] = "a"; //compile time error

Object[] names = new String[3];

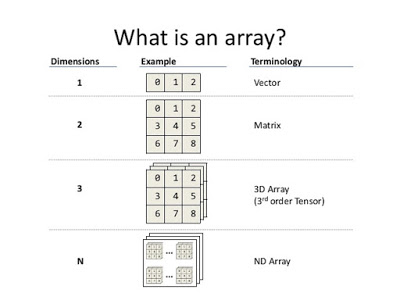
names[0] = new Integer(0); // ArrayStoreException at runtime

**Question 3 : What is difference between ArrayIndexOutfOBounds and ArrayStoreException?**[[answer](http://javarevisited.blogspot.sg/2014/05/exception-in-thread-main-arrayindexoutofboundsexception-java.html)]  
ArrayIndexOutOfBoundsException comes when your code tries to access an invalid index for a given array e.g. negative index or higher index than length - 1. While, ArrayStoreException comes when you have stored an element of type other than type of array, as shown in above example.  
  
  
**Question 4 : Can you use Generics with array?** [[answer](http://javarevisited.blogspot.sg/2011/09/generics-java-example-tutorial.html)]  
No, you cannot use Generic with array, that's why sometime List is better choice over array in Java.  
  
  
**Question 5 : Is it legal to initialize an array int i[] = {1, 2, 3, 4, 5};** [answer]  
Yes, its perfectly legal. You can create and initialize array in same line in Java.  
  
  
**Question 6 : Difference between a[] and []a in Java?**[answer]  
You can declare an array in Java by either prefixing or suffixing[] with variable. There is not much difference between them if you are not creating more than one variable in one line, but if you do then it creates different types of variables, as shown in following example :

int a[], b; // first is int array, second is just int variable

int[] c, d; // both c and d are integer array

**Question 7 : What is two dimensional array?** [[answer](http://java67.blogspot.sg/2014/10/how-to-create-and-initialize-two-dimensional-array-java-example.html)]  
An array of array in Java.  You can declare them like int[][] primes = new int[3][3] which is a matrix of 3x3.  
  
  
**Question 8 : Do you have three dimensional array in Java?** [answer]  
Yes, Java supports N dimensional array. Actually multi-dimensional array in Java is nothing but an array of array, for example, two dimensional array is just an array of one dimensional array.  
  
  
**Question 9 : How to iterate over array in Java?** [[answer](http://java67.blogspot.sg/2013/08/how-to-iterate-over-array-in-java-15.html)]  
You can either use classical for loop with index or advanced for loop introduced in Java 5 to iterate over an array in Java. If you need index to select some element or do something else, use for loop otherwise advanced for loop is better. It's less error prone as you don't need to deal with index.  
  
  
**Question 10 : How to search an array to check if an element exists there?** [[answer](http://java67.blogspot.sg/2012/10/how-to-find-index-of-item-in-array-in.html)]  
You can search an element inside array by using either linear search or binary search. Later is faster but you need to sort the array before performing binary search on it. Arrays class from java.util package provides binarySearch() method to search an element in array. Alternatively, you can also convert array to ArrayList and use its contains() method to find out if an element exists or not.  
  
  
**Question 11 : How to sort an array in Java?** [[answer](http://java67.blogspot.sg/2014/08/4-examples-to-sort-array-in-java.html)]  
You can sort an array in Java by using Arrays.sort() method. Arrays is a utility class which contains lots of static utility method to operate on arrays. This method is overloaded and you can optionally provide a Comparator implementation to sort array in custom order.  
  
  
**Question 12 : How to copy array in Java?** [answer]  
You can either manually copy elements of array by iterating over them, or you can use System.arrayCopy() method to copy elements form one array to other. This is a powerful method which provides fast copy and also allows you to copy entire or part of the array.  
  
  
**Question 13 : How to access elements of array in Java?** [[answer](http://java67.blogspot.sg/2013/08/ata-structures-in-java-programming-array-linked-list-map-set-stack-queue.html)]  
You can access elements of array using index in Java. It starts from 0, so first element is stored in location zero and last element has index length - 1. Trying to access an invalid index in Java e.g. negative index or index higher than size will result in ArrayIndexOutOfBoundsException in Java.  
  
  
**Question 14 :  What is difference between an array and a linked list?** [[answer](http://javarevisited.blogspot.sg/2013/07/difference-between-array-and-linked-list-java.html)]  
Some key difference between array and linked list data structure is, Array requires contiguous memory for its element but linked list elements can be scattered in memory.  Array is good for searching elements if you know the index, but adding and removing elements in array is expensive as compared to linked list.  
  
  
**Question 15 : Can you make array volatile in Java?** [[answer](http://java67.blogspot.sg/2012/08/what-is-volatile-variable-in-java-when.html)]  
This is another tricky question in Java. Yes, you can make an array volatile in Java, but you only make the variable which is pointing to array volatile. If array is changed by replacing individual elements than happens before guarantee provided by volatile variables will not held.  
  
  
**Question 16 : Where does array stored in memory?** [answer]  
Array is created in heap space of JVM memory. Since array is object in Java, even if you create array locally inside a method or block, object is always allocated memory from heap.

[](http://3.bp.blogspot.com/-iBZa-IVSj6Y/VbJXgPwIH-I/AAAAAAAADdY/NiyjIBK2y24/s1600/What%2Bis%2Bin%2Barray.jpg)

## Array based Coding Interview Problems

Some array based coding interview questions, which require to build some logic to solve the problem. These questions are mainly asked to check your problem solving skill apart from your command on programming language. Actually, once you know the logic, you can solve these problem in any programming language e.g. C, C++, Python, Ruby or JavaScript.  
  
  
**Problem 1 : How to find missing number in array of 1 to 100 in Java?** [[solution](http://java67.blogspot.sg/2014/11/how-to-test-if-array-contains-certain-value-in-java.html)]  
You have given an array of integer which contains numbers from 1 to 100, but exactly one number is missing, how do you find that number? You can use additional data structure.  
  
  
**Problem 2 : How do you find all pair whose sum is equal to given number from integer array in Java?** [[solution](http://javarevisited.blogspot.sg/2014/08/how-to-find-all-pairs-in-array-of-integers-whose-sum-equal-given-number-java.html)]  
You have given an array of int primitives and a number, you need to find all pairs in array whose sum is equal to given number e.g. if array is {1, 2, 3,  4, 5} and given sum is 6 then your program should return {2, 4} and {1, 5}  
  
  
**Problem 3 : How do you remove duplicates from array in Java?** [[solution](http://javarevisited.blogspot.sg/2014/01/how-to-remove-duplicates-from-array-java-without-collection-API.html)]  
You have given an array, which could be array of numbers or Strings or any objects. Some objects are added multiple times in array, you need to remove those elements from array. You don't know how many duplicates are there. You can use additional data structure and memory to solve this problem.  
  
  
**Problem 4 : How do you reverse an array in Java?**[[solution](http://javarevisited.blogspot.com/2013/03/how-to-reverse-array-in-java-int-String-array-example.html)]  
Given an array of String or integer, how do you reverse array, so that first element becomes last and last element becomes first e.g. if given array is {1, 2, 3, 4} then your program should reverse it as {4, 3, 2, 1}. You can use additional memory but reversing array in place will get bonus point.  
  
  
**Problem 5 : How to find duplicates numbers in array of integers in Java?** [[solution](http://javarevisited.blogspot.com/2012/02/how-to-check-or-detect-duplicate.html)]  
You have given an array {1, 1, 2, 3, 3, 4}, write a program to return duplicate elements e.g. 1 and 3 because they have appeared more than once in array.  
  
  
**Problem 6 : How to find top two numbers from an integer array in Java?**[[solution](http://java67.blogspot.sg/2014/03/how-to-find-top-two-maximum-number-from-integer-array-java.html)]  
You have given an integer array e.g. { 3, 4,  5,  6 , 7}, you need to find top 2 numbers from this array e.g. your program should print 6 and 7.  
  
Read more: <http://www.java67.com/2015/07/array-concepts-interview-questions-answers-java.html#ixzz4fFTuJW9g>

Garbage Collectors

**35. What is the purpose of garbage collection in Java, and when is it used ?** The purpose of garbage collection is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

**36. What does System.gc() and Runtime.gc() methods do ?** These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

**37. When is the finalize() called ? What is the purpose of finalization ?** The finalize method is called by the garbage collector, just before releasing the object’s memory. It is normally advised to release resources held by the object inside the finalize method.

**38. If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object ?** No, the object will be available for garbage collection in the next cycle of the garbage collector.

**39. What is structure of Java Heap ? What is Perm Gen space in Heap ?** The [JVM has a heap](http://www.javacodegeeks.com/2012/07/5-tips-for-proper-java-heap-size.html) that is the runtime data area from which memory for all class instances and arrays is allocated. It is created at the JVM start-up. Heap memory for objects is reclaimed by an automatic memory management system which is known as a garbage collector. Heap memory consists of live and dead objects. Live objects are accessible by the application and will not be a subject of garbage collection. Dead objects are those which will never be accessible by the application, but have not been collected by the garbage collector yet. Such objects occupy the heap memory space until they are eventually collected by the garbage collector.

**40. What is the difference between Serial and Throughput Garbage collector ?** The throughput garbage collector uses a parallel version of the young generation collector and is meant to be used with applications that have medium to large data sets. On the other hand, the serial collector is usually adequate for most small applications (those requiring heaps of up to approximately 100MB on modern processors).

**41. When does an Object becomes eligible for Garbage collection in Java ?** A Java object is subject to garbage collection when it becomes unreachable to the program in which it is currently used.

**42. Does Garbage collection occur in permanent generation space in JVM ?** Garbage Collection does occur in PermGen space and if PermGen space is full or cross a threshold, it can trigger a full garbage collection. If you look carefully at the output of the garbage collector, you will find that PermGen space is also garbage collected. This is the reason why correct sizing of PermGen space is important to avoid frequent full garbage collections. Also check our article [Java 8: PermGen to Metaspace](http://www.javacodegeeks.com/2013/02/java-8-from-permgen-to-metaspace.html).

Exception Handling

**43. What are the two types of Exceptions in Java ? Which are the differences between them ?** Java has two types of exceptions: checked exceptions and unchecked exceptions. Unchecked exceptions do not need to be declared in a method or a constructor’s throws clause, if they can be thrown by the execution of the method or the constructor, and propagate outside the method or constructor boundary. On the other hand, checked exceptions must be declared in a method or a constructor’s throws clause. See here for tips on [Java exception handling](http://www.javacodegeeks.com/2013/07/java-exception-handling-tutorial-with-examples-and-best-practices.html).

**44. What is the difference between Exception and Error in java ?** [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) and [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) classes are both subclasses of the [Throwable](http://docs.oracle.com/javase/7/docs/api/java/lang/Throwable.html) class. The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) class is used for exceptional conditions that a user’s program should catch. The [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) class defines exceptions that are not excepted to be caught by the user program.

**45. What is the difference between throw and throws ?** The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

**45. What is the importance of finally block in exception handling ?** A finally block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

**46. What will happen to the Exception object after exception handling ?** The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) object will be garbage collected in the next garbage collection.

**47. How does finally block differ from finalize() method ?** A finally block will be executed whether or not an exception is thrown and is used to release those resources held by the application. Finalize is a protected method of the Object class, which is called by the Java Virtual Machine (JVM) just before an object is garbage collected.

Java Applets

**48. What is an Applet ?** A java applet is program that can be included in a HTML page and be executed in a java enabled client browser. Applets are used for creating dynamic and interactive web applications.

**49. Explain the life cycle of an Applet.** An applet may undergo the following states:

* **Init**: An applet is initialized each time is loaded.
* **Start**: Begin the execution of an applet.
* **Stop**: Stop the execution of an applet.
* **Destroy**: Perform a final cleanup, before unloading the applet.

**50. What happens when an applet is loaded ?** First of all, an instance of the applet’s controlling class is created. Then, the applet initializes itself and finally, it starts running.

**51. What is the difference between an Applet and a Java Application ?** Applets are executed within a java enabled browser, but a Java application is a standalone Java program that can be executed outside of a browser. However, they both require the existence of a Java Virtual Machine (JVM). Furthermore, a Java application requires a main method with a specific signature, in order to start its execution. Java applets don’t need such a method to start their execution. Finally, Java applets typically use a restrictive security policy, while Java applications usually use more relaxed security policies.

**52. What are the restrictions imposed on Java applets ?** Mostly due to security reasons, the following restrictions are imposed on Java applets:

* An applet cannot load libraries or define native methods.
* An applet cannot ordinarily read or write files on the execution host.
* An applet cannot read certain system properties.
* An applet cannot make network connections except to the host that it came from.
* An applet cannot start any program on the host that’s executing it.

**53. What are untrusted applets ?** Untrusted applets are those Java applets that cannot access or execute local system files. By default, all downloaded applets are considered as untrusted.

**54. What is the difference between applets loaded over the internet and applets loaded via the file system ?** Regarding the case where an applet is loaded over the internet, the applet is loaded by the applet classloader and is subject to the restrictions enforced by the applet security manager. Regarding the case where an applet is loaded from the client’s local disk, the applet is loaded by the file system loader. Applets loaded via the file system are allowed to read files, write files and to load libraries on the client. Also, applets loaded via the file system are allowed to execute processes and finally, applets loaded via the file system are not passed through the byte code verifier.

**55. What is the applet class loader, and what does it provide ?** When an applet is loaded over the internet, the applet is loaded by the applet classloader. The class loader enforces the Java name space hierarchy. Also, the class loader guarantees that a unique namespace exists for classes that come from the local file system, and that a unique namespace exists for each network source. When a browser loads an applet over the net, that applet’s classes are placed in a private namespace associated with the applet’s origin. Then, those classes loaded by the class loader are passed through the verifier.The verifier checks that the class file conforms to the Java language specification . Among other things, the verifier ensures that there are no stack overflows or underflows and that the parameters to all bytecode instructions are correct.

**56. What is the applet security manager, and what does it provide ?** The applet security manager is a mechanism to impose restrictions on Java applets. A browser may only have one security manager. The security manager is established at startup, and it cannot thereafter be replaced, overloaded, overridden, or extended.

[Swing](http://examples.javacodegeeks.com/desktop-java/swing/)

**57. What is the difference between a Choice and a List ?** A Choice is displayed in a compact form that must be pulled down, in order for a user to be able to see the list of all available choices. Only one item may be selected from a Choice. A [List](http://examples.javacodegeeks.com/desktop-java/swing/jlist/create-jlist-example/) may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

**58. What is a layout manager ?** A layout manager is the used to organize the components in a container.

**59. What is the difference between a Scrollbar and a JScrollPane ?** A [Scrollbar](http://docs.oracle.com/javase/7/docs/api/java/awt/Scrollbar.html) is a [Component](http://docs.oracle.com/javase/7/docs/api/java/awt/Component.html), but not a [Container](http://docs.oracle.com/javase/7/docs/api/java/awt/Container.html). A [ScrollPane](http://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html) is a [Container](http://docs.oracle.com/javase/7/docs/api/java/awt/Container.html). A [ScrollPane](http://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html) handles its own events and performs its own scrolling.

**60. Which Swing methods are thread-safe ?** There are only three thread-safe methods: repaint, revalidate, and invalidate.

**61. Name three Component subclasses that support painting.** The [Canvas](http://docs.oracle.com/javase/7/docs/api/java/awt/Canvas.html), [Frame](http://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html), [Panel](http://docs.oracle.com/javase/7/docs/api/java/awt/Panel.html), and Applet classes support painting.

**62. What is clipping ?** Clipping is defined as the process of confining paint operations to a limited area or shape.

**63. What is the difference between a MenuItem and a CheckboxMenuItem ?** The [CheckboxMenuItem](http://docs.oracle.com/javase/7/docs/api/java/awt/CheckboxMenuItem.html) class extends the [MenuItem](http://docs.oracle.com/javase/7/docs/api/java/awt/MenuItem.html) class and supports a menu item that may be either checked or unchecked.

**64. How are the elements of a BorderLayout organized ?** The elements of a [BorderLayout](http://docs.oracle.com/javase/7/docs/api/java/awt/BorderLayout.html) are organized at the borders (North, South, East, and West) and the center of a container.

**65. How are the elements of a GridBagLayout organized ?** The elements of a [GridBagLayout](http://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html) are organized according to a grid. The elements are of different sizes and may occupy more than one row or column of the grid. Thus, the rows and columns may have different sizes.

**66. What is the difference between a Window and a Frame ?** The [Frame](http://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html) class extends the [Window](http://docs.oracle.com/javase/7/docs/api/java/awt/Window.html) class and defines a main application window that can have a menu bar.

**67. What is the relationship between clipping and repainting ?** When a window is repainted by the AWT painting thread, it sets the clipping regions to the area of the window that requires repainting.

**68. What is the relationship between an event-listener interface and an event-adapter class ?** An event-listener interface defines the methods that must be implemented by an event handler for a particular event. An event adapter provides a default implementation of an event-listener interface.

**69. How can a GUI component handle its own events ?** A GUI component can handle its own events, by implementing the corresponding event-listener interface and adding itself as its own event listener.

**70. What advantage do Java’s layout managers provide over traditional windowing systems ?** Java uses layout managers to lay out components in a consistent manner, across all windowing platforms. Since layout managers aren’t tied to absolute sizing and positioning, they are able to accomodate platform-specific differences among windowing systems.

**71. What is the design pattern that Java uses for all Swing components ?** The design pattern used by Java for all Swing components is the Model View Controller (MVC) pattern.

JDBC

**72. What is JDBC ?** JDBC is an abstraction layer that allows users to choose between databases. [JDBC enables developers to write database applications in Java](http://www.javacodegeeks.com/2014/03/java-8-friday-java-8-will-revolutionize-database-access.html), without having to concern themselves with the underlying details of a particular database.

**73. Explain the role of Driver in JDBC.** The JDBC Driver provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each driver must provide implementations for the following classes of the java.sql package:[Connection](http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html), [Statement](http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html), [PreparedStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html), [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html), [ResultSet](http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html) and [Driver](http://docs.oracle.com/javase/7/docs/api/java/sql/Driver.html).

**74. What is the purpose Class.forName method ?** This method is used to method is used to load the driver that will establish a connection to the database.

**75. What is the advantage of PreparedStatement over Statement ?** PreparedStatements are precompiled and thus, [their performance is much better](http://examples.javacodegeeks.com/core-java/sql/batch-statement-execution-example/). Also, PreparedStatement objects can be reused with different input values to their queries.

**76. What is the use of CallableStatement ? Name the method, which is used to prepare a CallableStatement.** A [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html) is used to execute stored procedures. Stored procedures are stored and offered by a database. Stored procedures may take input values from the user and may return a result. The usage of stored procedures is highly encouraged, because it offers security and modularity.The method that prepares a [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html) is the following:

|  |  |
| --- | --- |
| 1 | CallableStament.prepareCall(); |

**77. What does Connection pooling mean ?** The interaction with a database can be costly, regarding the opening and closing of database connections. Especially, when the number of database clients increases, this cost is very high and a large number of resources is consumed.A pool of database connections is obtained at start up by the application server and is maintained in a pool. A request for a connection is served by a [connection residing in the pool](http://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-connection-pool-configuration-with-c3p0-example/). In the end of the connection, the request is returned to the pool and can be used to satisfy future requests.

Remote Method Invocation (RMI)

**78. What is RMI ?** The Java Remote Method Invocation (Java RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection. Remote Method Invocation (RMI) can also be seen as the process of activating a method on a remotely running object. RMI offers location transparency because a user feels that a method is executed on a locally running object. Check some [RMI Tips here](http://www.javacodegeeks.com/2013/11/two-things-to-remember-when-using-java-rmi.html).

**79. What is the basic principle of RMI architecture ?** The RMI architecture is based on a very important principle which states that the definition of the behavior and the implementation of that behavior, are separate concepts. RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.

**80. What are the layers of RMI Architecture ?** The RMI architecture consists of the following layers:

* **Stub and Skeleton layer**: This layer lies just beneath the view of the developer. This layer is responsible for intercepting method calls made by the client to the interface and redirect these calls to a remote RMI Service.
* **Remote Reference Layer**: The second layer of the RMI architecture deals with the interpretation of references made from the client to the server’s remote objects. This layer interprets and manages references made from clients to the remote service objects. The connection is a one-to-one (unicast) link.
* **Transport layer**: This layer is responsible for connecting the two JVM participating in the service. This layer is based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies.

**81. What is the role of Remote Interface in RMI ?** The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. A class that implements a remote interface should declare the remote interfaces being implemented, define the constructor for each remote object and provide an implementation for each remote method in all remote interfaces.

**82. What is the role of the java.rmi.Naming Class ?** The java.rmi.Naming class provides methods for storing and obtaining references to remote objects in the remote object registry. Each method of the Naming class takes as one of its arguments a name that is a String in URL format.

**83. What is meant by binding in RMI ?** Binding is the process of associating or registering a name for a remote object, which can be used at a later time, in order to look up that remote object. A remote object can be associated with a name using the bind or rebind methods of the Naming class.

**84. What is the difference between using bind() and rebind() methods of Naming Class ?** The bind method bind is responsible for binding the specified name to a remote object, while the rebind method is responsible for rebinding the specified name to a new remote object. In case a binding exists for that name, the binding is replaced.

**85. What are the steps involved to make work a RMI program ?** The following steps must be involved in order for a RMI program to work properly:

* Compilation of all source files.
* Generatation of the stubs using rmic.
* Start the rmiregistry.
* Start the RMIServer.
* Run the client program.

**86. What is the role of stub in RMI ?** A stub for a remote object acts as a client’s local representative or proxy for the remote object. The caller invokes a method on the local stub, which is responsible for executing the method on the remote object. When a stub’s method is invoked, it undergoes the following steps:

* It initiates a connection to the remote JVM containing the remote object.
* It marshals the parameters to the remote JVM.
* It waits for the result of the method invocation and execution.
* It unmarshals the return value or an exception if the method has not been successfully executed.
* It returns the value to the caller.

**87. What is DGC ? And how does it work ?** DGC stands for Distributed Garbage Collection. Remote Method Invocation (RMI) uses DGC for automatic garbage collection. Since RMI involves remote object references across JVM’s, garbage collection can be quite difficult. DGC uses a reference counting algorithm to provide automatic memory management for remote objects.

**88. What is the purpose of using RMISecurityManager in RMI ?** RMISecurityManager provides a security manager that can be used by RMI applications, which use downloaded code. The class loader of RMI will not download any classes from remote locations, if the security manager has not been set.

**89. Explain Marshalling and demarshalling.** When an application wants to pass its memory objects across a network to another host or persist it to storage, the in-memory representation must be converted to a suitable format. This process is called marshalling and the revert operation is called demarshalling.

**90. Explain Serialization and Deserialization.** Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes and includes the object’s data, as well as information about the object’s type, and the types of data stored in the object. Thus, serialization can be seen as a way of flattening objects, in order to be stored on disk, and later, read back and reconstituted. Deserialisation is the reverse process of converting an object from its flattened state to a live object.

Servlets

**91. What is a Servlet ?** [The servlet](http://examples.javacodegeeks.com/enterprise-java/servlet/sample-java-servlet/) is a Java programming language class used to process client requests and generate dynamic web content. Servlets are mostly used to process or store data submitted by an HTML form, provide dynamic content and manage state information that does not exist in the stateless HTTP protocol.

**92. Explain the architechure of a Servlet.** The core abstraction that must be implemented by all servlets is the javax.servlet.Servlet interface. Each servlet must implement it either directly or indirectly, either by extending javax.servlet.GenericServlet or javax.servlet.http.HTTPServlet. Finally, each servlet is able to serve multiple requests in parallel using multithreading.

**93. What is the difference between an Applet and a Servlet ?** An Applet is a client side java program that runs within a Web browser on the client machine. On the other hand, a servlet is a server side component that runs on the web server.An applet can use the user interface classes, while a servlet does not have a user interface. Instead, a servlet waits for client’s HTTP requests and generates a response in every request.

**94. What is the difference between GenericServlet and HttpServlet ?** GenericServlet is a generalized and protocol-independent servlet that implements the Servlet and ServletConfig interfaces. Those servlets extending the GenericServlet class shall override the service method. Finally, in order to develop an HTTP servlet for use on the Web that serves requests using the HTTP protocol, your servlet must extend the HttpServlet instead. Check [Servlet examples here](http://examples.javacodegeeks.com/tag/servlet/).

**95. Explain the life cycle of a Servlet.** On every client’s request, the Servlet Engine loads the servlets and invokes its init methods, in order for the servlet to be initialized. Then, the Servlet object handles all subsequent requests coming from that client, by invoking the service method for each request separately. Finally, the servlet is removed by calling the server’s destroy method.

**96. What is the difference between doGet() and doPost() ?** doGET: The GET method appends the name-value pairs on the request’s URL. Thus, there is a limit on the number of characters and subsequently on the number of values that can be used in a client’s request. Furthermore, the values of the request are made visible and thus, sensitive information must not be passed in that way. doPOST: The POST method overcomes the limit imposed by the GET request, by sending the values of the request inside its body. Also, there is no limitations on the number of values to be sent across. Finally, the sensitive information passed through a POST request is not visible to an external client.

**97. What is meant by a Web Application ?** A Web application is a dynamic extension of a Web or application server. There are two types of web applications: presentation-oriented and service-oriented. A presentation-oriented Web application generates interactive web pages, which contain various types of markup language and dynamic content in response to requests. On the other hand, a service-oriented web application implements the endpoint of a web service. In general, a Web application can be seen as a collection of servlets installed under a specific subset of the server’s URL namespace.

**98. What is a Server Side Include (SSI) ?** Server Side Includes (SSI) is a simple interpreted server-side scripting language, used almost exclusively for the Web, and is embedded with a servlet tag. The most frequent use of SSI is to include the contents of one or more files into a Web page on a Web server. When a Web page is accessed by a browser, the Web server replaces the servlet tag in that Web page with the hyper text generated by the corresponding servlet.

**99. What is Servlet Chaining ?** Servlet Chaining is the method where the output of one servlet is sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. The last servlet in the chain is responsible for sending the response to the client.

**100. How do you find out what client machine is making a request to your servlet ?** The ServletRequest class has functions for finding out the IP address or host name of the client machine. getRemoteAddr() gets the IP address of the client machine and getRemoteHost() gets the host name of the client machine. See example [here](http://examples.javacodegeeks.com/enterprise-java/servlet/get-client-s-address-and-hostname-in-servlet/).

**101. What is the structure of the HTTP response ?** The HTTP response consists of three parts:

* **Status Code**: describes the status of the response. It can be used to check if the request has been successfully completed. In case the request failed, the status code can be used to find out the reason behind the failure. If your servlet does not return a status code, the success status code, HttpServletResponse.SC\_OK, is returned by default.
* **HTTP Headers**: they contain more information about the response. For example, the headers may specify the date/time after which the response is considered stale, or the form of encoding used to safely transfer the entity to the user. See [how to retrieve headers in Servlet here](http://examples.javacodegeeks.com/enterprise-java/servlet/get-all-request-headers-in-servlet/).
* **Body**: it contains the content of the response. The body may contain HTML code, an image, etc. The body consists of the data bytes transmitted in an HTTP transaction message immediately following the headers.

**102. What is a cookie ? What is the difference between session and cookie ?** [A cookie](http://examples.javacodegeeks.com/core-java/net/urlconnection/get-cookies-from-http-connection/) is a bit of information that the Web server sends to the browser. The browser stores the cookies for each Web server in a local file. In a future request, the browser, along with the request, sends all stored cookies for that specific Web server.The differences between session and a cookie are the following:

* The session should work, regardless of the settings on the client browser. The client may have chosen to disable cookies. However, the sessions still work, as the client has no ability to disable them in the server side.
* The session and cookies also differ in the amount of information the can store. The HTTP session is capable of storing any Java object, while a cookie can only store String objects.

**103. Which protocol will be used by browser and servlet to communicate ?** The browser communicates with a servlet by using the HTTP protocol.

**104. What is HTTP Tunneling ?** HTTP Tunneling is a technique by which, communications performed using various network protocols are encapsulated using the HTTP or HTTPS protocols. The HTTP protocol therefore acts as a wrapper for a channel that the network protocol being tunneled uses to communicate. The masking of other protocol requests as HTTP requests is HTTP Tunneling.

**105. What’s the difference between sendRedirect and forward methods ?** The sendRedirect method creates a new request, while the forward method just forwards a request to a new target. The previous request scope objects are not available after a redirect, because it results in a new request. On the other hand, the previous request scope objects are available after forwarding. FInally, in general, the sendRedirect method is considered to be slower compare to the forward method.

**106. What is URL Encoding and URL Decoding ?** The URL encoding procedure is responsible for replacing all the spaces and every other extra special character of a URL, into their corresponding Hex representation. In correspondence, URL decoding is the exact opposite procedure.

JSP

**107. What is a JSP Page ?** A Java Server Page (JSP) is a text document that contains two types of text: static data and JSP elements. Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically-generated content. See [JSP example here](http://examples.javacodegeeks.com/enterprise-java/jsp/sample-jsp-java-server-page/).

**108. How are the JSP requests handled ?** On the arrival of a JSP request, the browser first requests a page with a .jsp extension. Then, the Web server reads the request and using the JSP compiler, the Web server converts the JSP page into a servlet class. Notice that the JSP file is compiled only on the first request of the page, or if the JSP file has changed.The generated servlet class is invoked, in order to handle the browser’s request. Once the execution of the request is over, the servlet sends a response back to the client. See [how to get Request parameters in a JSP](http://examples.javacodegeeks.com/enterprise-java/jsp/get-request-parameter-in-jsp-page/).

**109. What are the advantages of JSP ?** The advantages of using the JSP technology are shown below:

* JSP pages are dynamically compiled into servlets and thus, the developers can easily make updates to presentation code.
* JSP pages can be pre-compiled.
* JSP pages can be easily combined to static templates, including HTML or XML fragments, with code that generates dynamic content.
* Developers can offer customized JSP tag libraries that page authors access using an XML-like syntax.
* Developers can make logic changes at the component level, without editing the individual pages that use the application’s logic.

**110. What are Directives ? What are the different types of Directives available in JSP ?** Directives are instructions that are processed by the JSP engine, when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries. Directives are defined between < %@ and % >.The different types of directives are shown below:

* Include directive: it is used to include a file and merges the content of the file with the current page.
* Page directive: it is used to define specific attributes in the JSP page, like error page and buffer.
* Taglib: it is used to declare a custom tag library which is used in the page.

**111. What are JSP actions ?** JSP actions use constructs in XML syntax to control the behavior of the servlet engine. JSP actions are executed when a JSP page is requested. They can be dynamically inserted into a file, re-use JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.Some of the available actions are listed below:

* jsp:include – includes a file, when the JSP page is requested.
* jsp:useBean – finds or instantiates a JavaBean.
* jsp:setProperty – sets the property of a JavaBean.
* jsp:getProperty – gets the property of a JavaBean.
* jsp:forward – forwards the requester to a new page.
* jsp:plugin – generates browser-specific code.

**112. What are Scriptlets ?** In Java Server Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in a JSP page. The scriptlet is everything inside the tags. Between these tags, a user can add any valid scriplet.

**113. What are Decalarations ?** Declarations are similar to variable declarations in Java. Declarations are used to declare variables for subsequent use in expressions or scriptlets. To add a declaration, you must use the sequences to enclose your declarations.

**114. What are Expressions ?** A JSP expression is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client, by the web server. Expressions are defined between <% = and %>tags.

**115. What is meant by implicit objects and what are they ?** JSP implicit objects are those Java objects that the JSP Container makes available to developers in each page. A developer can call them directly, without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.The following objects are considered implicit in a JSP page:

* application
* page
* request
* response
* session
* exception
* out
* config
* pageContext

## Important Topics for Java Interviews

Apart from quantity, as you can see with a huge number of questions, I have worked hard to maintain quality as well. I have not only shared questions from all important topics but also ensured to include so-called advanced topics which many programmers do not prefer to prepare or just left out because they have not worked on that. Java NIO and JVM internals questions are best examples of that. You can keep design patterns also on the same list but growing number of an experienced programmer are now well aware of GOF design patterns and when to use them. I have also worked hard to keep this list up-to-date to include what interviewers are asking in 2015 and what will be their core focus on coming years. To give you an idea, this list of Java interview questions includes following topics:

1. Multithreading, concurrency and thread basics
2. Date type conversion and fundamentals
3. Garbage Collection
4. Java Collections Framework
5. Array
6. String
7. GOF Design Patterns
8. SOLID design principles
9. Abstract class and interface
10. Java basics e.g. equals() and hashcode
11. Generics and Enum
12. Java IO and NIO
13. Common Networking protocols
14. Data structure and algorithm in Java
15. Regular expressions
16. JVM internals
17. Java Best Practices
18. JDBC
19. Date, Time, and Calendar
20. XML Processing in Java
21. JUnit
22. Programming

You guys are also lucky that nowadays there are some good books available to prepare for Java interviews, one of them   which I particularly find useful and interesting to read is [Java Programming Interview Exposed by Markham](http://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20). It will take you to some of the most important topics for Java and JEE interviews, worth reading even if you are not preparing for Java interview.

## Top 120 Java Interview Questions Answers

So now the time has come to introduce you to this *MEGA list of 120 Java questions* collected from various interviews of last 5 years. I am sure you have seen many of these questions personally on your interviews and many of you would have answered them correctly as well.

### Multithreading, Concurrency and Thread basics Questions

**1) Can we make array volatile in Java?**  
This is one of the tricky Java multi-threading questions you will see in senior Java developer Interview. Yes, you can make an array volatile in Java but only the reference which is pointing to an array, not the whole array. What I mean, if one thread changes the reference variable to points to another array, that will provide a volatile guarantee, but if multiple threads are changing individual array elements they won't be having happens before guarantee provided by the volatile modifier.  
  
  
**2) Can volatile make a non-atomic operation to atomic?**  
This another good question I love to ask on volatile, mostly as a follow-up of the previous question. This question is also not easy to answer because volatile is not about atomicity, but there are cases where you can use a volatile variable to make the operation atomic.  
  
One example I have seen is having a long field in your class. If you know that a long field is accessed by more than one thread e.g. a counter, a price field or anything, you better make it volatile. Why? because reading to a long variable is not atomic in Java and done in two steps, If one thread is writing or updating long value, it's possible for another thread to see half value (fist 32-bit). While reading/writing a volatile long or double (64 bit) is atomic.  
  
  
  
**3) What are practical uses of volatile modifier?**  
One of the practical use of the volatile variable is to make reading double and long atomic. Both double and long are 64-bit wide and they are read in two parts, first 32-bit first time and next 32-bit second time, which is non-atomic but volatile double and long read is atomic in Java. Another use of the volatile variable is to provide a memory barrier, just like it is used in Disrupter framework. Basically, Java Memory model inserts a write barrier after you write to a volatile variable and a read barrier before you read it. Which means, if you write to volatile field then it's guaranteed that any thread accessing that variable will see the value you wrote and anything you did before doing that right into the thread is guaranteed to have happened and any updated data values will also be visible to all threads, because the memory barrier flushed all other writes to the cache.  
  
  
**4) What guarantee volatile variable provides?** ([answer](http://java67.blogspot.sg/2012/08/what-is-volatile-variable-in-java-when.html))  
volatile variables provide the guarantee about ordering and visibility e.g. volatile assignment cannot be re-ordered with other statements but in the absence of any synchronization instruction compiler, JVM or JIT are free to reorder statements for better performance. volatile also provides the happens-before guarantee which ensures changes made in one thread is visible to others. In some cases volatile also provide atomicity e.g. reading 64-bit data types like long and double are not atomic but read of volatile double or long is atomic.  
  
  
**5) Which one would be easy to write? synchronization code for 10 threads or 2 threads?**  
In terms of writing code, both will be of same complexity because synchronization code is independent of a number of threads. Choice of synchronization though depends upon a number of threads because the number of thread present more contention, so you go for advanced synchronization technique e.g. lock stripping, which requires more complex code and expertise.  
  
  
**6) How do you call wait() method? using if block or loop? Why?** ([answer](http://javarevisited.blogspot.sg/2015/07/how-to-use-wait-notify-and-notifyall-in.html))  
wait() method should always be called in loop because it's possible that until thread gets CPU to start running again the condition might not hold, so it's always better to check condition in loop before proceeding. Here is the standard idiom of using wait and notify method in Java:

// The standard idiom for using the wait method

synchronized (obj) {

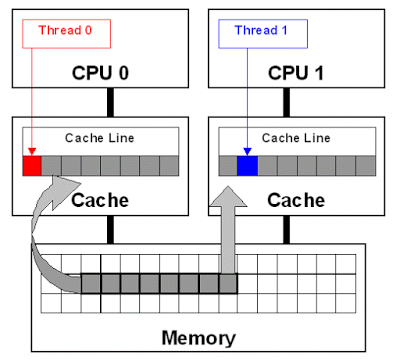
while (condition does not hold)

obj.wait(); // (Releases lock, and reacquires on wakeup)

... // Perform action appropriate to condition

}

See [Effective Java Item 69](http://www.amazon.com/dp/0321356683/?tag=javamysqlanta-20) to learn more about why wait method should call in the loop.  
  
  
**7)  What is false sharing in the context of multi-threading?**  
false sharing is one of the well-known performance issues on multi-core systems, where each process has its local cache. false sharing occurs when threads on different processor modify variables that reside on same cache line as shown in the following image:

[](http://2.bp.blogspot.com/-Tze9foqpb74/VepwCzXHGCI/AAAAAAAADtM/i4KQDaefqk4/s1600/False+Sharing+in+Multi-threaded+application.gif)

False sharing is very hard to detect because the thread may be accessing completely different global variables that happen to be relatively close together in memory. Like many concurrency issues, the primary way to avoid false sharing is careful code review and aligning your data structure with the size of a cache line.  
  
  
**8) What is busy spin? Why should you use it?**  
Busy spin is one of the technique to wait for events without releasing CPU. It's often done to avoid losing data in CPU cached which is lost if the thread is paused and resumed in some other core. So, if you are working on low latency system where your order processing thread currently doesn't have any order, instead of sleeping or calling wait(), you can just loop and then again check the queue for new messages. It's only beneficial if you need to wait for a very small amount of time e.g. in micro seconds or nano seconds. [LMAX Disrupter](http://lmax-exchange.github.io/disruptor/) framework, a high-performance inter-thread messaging library has a BusySpinWaitStrategy which is based on this concept and uses a busy spin loop for EventProcessors waiting on the barrier.  
  
  
**9) How do you take thread dump in Java?**  
You can take a thread dump of Java application in Linux by using **kill -3 PID**, where PID is the process id of Java process. In Windows, you can press **Ctrl + Break**. This will instruct JVM to print thread dump in standard out or err and it could go to console or log file depending upon your application configuration. If you have used Tomcat then when  
  
  
  
**10) is Swing thread-safe?** ([answer](http://javarevisited.blogspot.sg/2013/08/why-swing-is-not-thread-safe-in-java-Swingworker-Event-thread.html))  
No, Swing is not thread-safe. You cannot update Swing components e.g. JTable, JList or JPanel from any thread, in fact, they must be updated from GUI or AWT thread. That's why swings provide invokeAndWait() and invokeLater() method to request GUI update from any other threads. This methods put update request in AWT threads queue and can wait till update or return immediately for an asynchronous update. You can also check the detailed answer to learn more.  
  
  
  
  
**11) What is a thread local variable in Java?** ([answer](http://javarevisited.blogspot.sg/2012/05/how-to-use-threadlocal-in-java-benefits.html))  
Thread-local variables are variables confined to a thread, its like thread's own copy which is not shared between multiple threads. Java provides a ThreadLocal class to support thread-local variables. It's one of the many ways to achieve thread-safety. Though be careful while using thread local variable in manged environment e.g. with web servers where worker thread out lives any application variable. Any thread local variable which is not removed once its work is done can potentially cause a memory leak in Java application.  
  
  
**12) Write wait-notify code for producer-consumer problem?** ([answer](http://java67.blogspot.sg/2012/12/producer-consumer-problem-with-wait-and-notify-example.html))  
Please see the answer for a code example. Just remember to call wait() and notify() method from synchronized block and test waiting for condition on the loop instead of if block.  
  
  
**13) Write code for thread-safe Singleton in Java?** ([answer](http://javarevisited.blogspot.in/2012/12/how-to-create-thread-safe-singleton-in-java-example.html))  
Please see the answer for a code example and step by step guide to creating thread-safe singleton class in Java. When we say thread-safe, which means Singleton should remain singleton even if initialization occurs in the case of multiple threads. Using Java enum as Singleton class is one of the easiest ways to create a thread-safe singleton in Java.  
  
  
**14) The difference between sleep and wait in Java?** ([answer](http://java67.blogspot.sg/2012/08/what-are-difference-between-wait-and.html))  
Though both are used to pause currently running thread, sleep() is actually meant for short pause because it doesn't release lock, while wait() is meant for conditional wait and that's why it release lock which can then be acquired by another thread to change the condition on which it is waiting.  
  
  
**15) What is an immutable object? How do you create an Immutable object in Java?** ([answer](http://javarevisited.blogspot.sg/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html))  
Immutable objects are those whose state cannot be changed once created. Any modification will result in a new object e.g. String, Integer, and other wrapper class. Please see the answer for step by step guide to creating Immutable class in Java.  
  
  
**16) Can we create an Immutable object, which contains a mutable object?**  
Yes, its possible to create an Immutable object which may contain a mutable object, you just need to be a little bit careful not to share the reference of the mutable component, instead, you should return a copy of it if you have to. Most common example is an Object which contain the reference of java.util.Date object.

## Date types and Basic Java Interview Questions

**17) What is the right data type to represent a price in Java?** ([answer](http://javarevisited.blogspot.sg/2012/02/java-mistake-1-using-float-and-double.html))  
BigDecimal if memory is not a concern and Performance is not critical, otherwise double with predefined precision.  
  
  
**18) How do you convert bytes to String?** ([answer](http://javarevisited.blogspot.sg/2014/08/2-examples-to-convert-byte-array-to-String-in-Java.html))  
you can convert bytes to the string using string constructor which accepts byte[], just make sure that right character encoding otherwise platform's default character encoding will be used which may or may not be same.  
  
  
**19) How do you convert bytes to long in Java?** (answer)  
This questions if for you to answer :-)  
  
  
**20) Can we cast an int value into byte variable? what will happen if the value of int is larger than byte?**  
Yes, we can cast but int is 32 bit long in java while byte is 8 bit long in java so when you cast an int to byte higher 24 bits are lost and a byte can only hold a value from -128 to 128.  
  
  
**21) There are two classes B extends A and C extends B, Can we cast B into C e.g. C = (C) B;** ([answer](http://javarevisited.blogspot.sg/2012/12/what-is-type-casting-in-java-class-interface-example.html))  
  
  
**22) Which class contains clone method? Cloneable or Object?** ([answer](http://javarevisited.blogspot.sg/2015/01/java-clone-tutorial-part-2-overriding-with-mutable-field-example.html))  
java.lang.Cloneable is marker interface and doesn't contain any method clone method is defined in the object class. It is also knowing that clone() is a native method means it's implemented in C or C++ or any other native language.  
  
  
**23) Is ++ operator is thread-safe in Java?** (answer)  
 No it's not a thread safe operator because its involve multiple instructions like reading a value, incriminating it and storing it back into memory which can be overlapped between multiple threads.  
  
  
**24) Difference between a = a + b and a += b ?** (answer)  
The += operator implicitly cast the result of addition into the type of variable used to hold the result. When you add two integral variable e.g. variable of type byte, short, or int then they are first promoted to int and them addition happens. If result of addition is more than maximum value of a then a + b will give compile time error but a += b will be ok as shown below

byte a = 127;

byte b = 127;

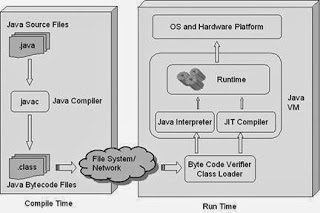
b = a + b; *// error : cannot convert from int to byte*

b += a; *// ok*

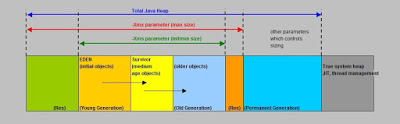
**25) Can I store a double value in a long variable without casting?** ([answer](http://java67.blogspot.com/2014/11/how-to-convert-double-to-long-in-java-example.html))  
No, you cannot store a double value into a long variable without casting because the range of double is more  that long and you we need to type cast. It's not dificult to answer this question but many develoepr get it wrong due to confusion on which one is bigger between double and long in Java.  
  
  
**26) What will this return 3\*0.1 == 0.3? true or false?** (answer)  
This is one of the really tricky questions. Out of 100, only 5 developers answered this question and only of them have explained the concept correctly. The short answer is false because some floating point numbers can not be represented exactly.  
  
  
**27) Which one will take more memory, an int or Integer?** (answer)  
An Integer object will take more memory an Integer is the an object and it  store meta data overhead about the object and int is primitive type so its takes less space.  
  
  
**28) Why is String Immutable in Java?** ([answer](http://java67.blogspot.sg/2014/01/why-string-class-has-made-immutable-or-final-java.html))  
One of my favorite Java interview question. The String is Immutable in java because java designer thought that string will be heavily used and making it immutable allow some optimization easy sharing same String object between multiple clients. See the link for the more detailed answer. This is a great question for Java programmers with less experience as it gives them food for thought, to think about how things works in Java, what Jave designers might have thought when they created String class etc.  
  
**29) Can we use String in the switch case?** ([answer](http://javarevisited.blogspot.sg/2011/08/string-switch-case-jdk7-example.html))  
Yes from Java 7 onward we can use String in switch case but it is just syntactic sugar. Internally string hash code is used for the switch. See the detaiedl answer for more explanation and discussion.  
  
**30) What is constructor chaining in Java?** ([answer](http://java67.blogspot.sg/2012/12/how-constructor-chaining-works-in-java.html))  
When you call one constructor from other than it's known as constructor chaining in Java. This happens when you have multiple, overloaded constructor in the class.

### JVM Internals and Garbage Collection Interview Questions

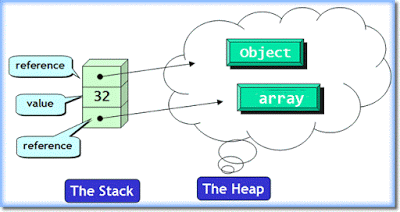
In the year 2015 I have seen increased focus on JVM internal and Garbage collection tuning, monitoring Java appliation, dealing with Java performance issues on various Java interviews. This is actually become the prime topic for interviewing any experienced Java developer for senior position e.g. technical lead, VP or team lead. If you feel you are short of experience and knowledge in this area then you should read atleast one book mentioned in my list of [Java Performance books](http://javarevisited.blogspot.com/2014/07/top-5-java-performance-tuning-books.html). I vote goes to Java Performance, The Definitive guide by Scott.  
  
**31) What is the size of int in 64-bit JVM?**  
The size of an int variable is constant in Java, it's always 32-bit irrespective of platform. Which means the size of primitive int is same in both 32-bit and 64-bit Java virtual machine.  
  
**32) The difference between Serial and Parallel Garbage Collector?** ([answer](http://javarevisited.blogspot.sg/2011/04/garbage-collection-in-java.html))  
Even though both the serial and parallel collectors cause a stop-the-world pause during Garbage collection. The main difference between them is that a serial collector is a default copying collector which uses only one GC thread for garbage collection while a parallel collector uses multiple GC threads for garbage collection.  
  
**33) What is the size of an int variable in 32-bit and 64-bit JVM?** (answer)  
The size of int is same in both 32-bit and 64-bit JVM, it's always 32 bits or 4 bytes.  
  
**34) A difference between WeakReference and SoftReference in Java?** ([answer](http://javarevisited.blogspot.sg/2014/03/difference-between-weakreference-vs-softreference-phantom-strong-reference-java.html))  
Though both WeakReference and SoftReference helps garbage collector and memory efficient, WeakReference becomes eligible for garbage collection as soon as last strong reference is lost but SoftReference even thought it can not prevent GC, it can delay it until JVM absolutely need memory.  
  
**35) How do WeakHashMap works?** (answer)  
WeakHashMap works like a normal HashMap but uses WeakReference for keys, which means if the key object doesn't have any reference then both key/value mapping will become eligible for garbage collection.  
  
**36) What is -XX:+UseCompressedOops JVM option? Why use it?** ([answer](http://javarevisited.blogspot.com/2012/06/what-is-xxusecompressedoops-in-64-bit.html))  
When you go migrate your Java application from 32-bit to 64-bit JVM, the heap requirement suddenly increases, almost double, due to increasing size of ordinary object pointer from 32 bit to 64 bit. This also adversely affect how much data you can keep in CPU cache, which is much smaller than memory. Since main motivation for moving to 64-bit JVM is to specify large heap size, you can save some memory by using compressed OOP. By using -XX:+UseCompressedOops, JVM uses 32-bit OOP instead of 64-bit OOP.  
  
  
**37) How do you find if JVM is 32-bit or 64-bit from Java Program?** ([answer](http://javarevisited.blogspot.sg/2012/01/find-jvm-is-32-or-64-bit-java-program.html))  
You can find that by checking some system properties like sun.arch.data.model or os.arch  
  
  
**38) What is the maximum heap size of 32-bit and 64-bit JVM?** ([answer](http://javarevisited.blogspot.sg/2013/04/what-is-maximum-heap-size-for-32-bit-64-JVM-Java-memory.html))  
Theoretically, the maximum heap memory you can assign to a 32-bit JVM is 2^32 which is 4GB but practically the limit is much smaller. It also varies between operating systems e.g. form 1.5GB in Windows to almost 3GB in Solaris. 64-bit JVM allows you to specify larger heap size, theoretically 2^64 which is quite large but practically you can specify heap space up to 100GBs. There are even JVM e.g. Azul where heap space of 1000 gigs is also possible.  
  
  
**39) What is the difference between JRE, JDK, JVM and JIT?** ([answer](http://javarevisited.blogspot.sg/2011/12/jre-jvm-jdk-jit-in-java-programming.html))  
JRE stands for Java run-time and it's required to run Java application. JDK stands for Java development kit and provides tools to develop Java program e.g. Java compiler. It also contains JRE. The JVM stands for Java virtual machine and it's the process responsible for running Java application. The JIT stands for Just In Time compilation and helps to boost the performance of Java application by converting Java byte code into native code when the crossed certain threshold i.e. mainly hot code is converted into native code.

[](http://2.bp.blogspot.com/-ls3yC0U7ouo/VhDqX-3OUbI/AAAAAAAAD40/Zcsc5uCaGq0/s1600/JVM+JRE+JDK.jpg)

**40) Explain Java Heap space and Garbage collection?** ([answer](http://javarevisited.blogspot.sg/2011/05/java-heap-space-memory-size-jvm.html))  
When a Java process is started using java command, memory is allocated to it. Part of this memory is used to create heap space, which is used to allocate memory to objects whenever they are created in the program. Garbage collection is the process inside JVM which reclaims memory from dead objects for future allocation.

[](http://3.bp.blogspot.com/-DqV12_uIeZ4/VhDqtPCVIVI/AAAAAAAAD48/uqWZB0BgZUI/s1600/java_heaps_memory.jpg)

**41) Can you guarantee the garbage collection process?**(answer)  
No, you cannot guarantee the garbage collection, though you can make a request using System.gc() or Runtime.gc() method.  
  
  
**42) How do you find memory usage from Java program? How much percent of the heap is used?**  
You can use memory related methods from java.lang.Runtime class to get the free memory, total memory and maximum heap memory in Java.  By using these methods, you can find out how many percents of the heap is used and how much heap space is remaining. Runtime.freeMemory() return amount of free memory in bytes, Runtime.totalMemory() returns total memory in bytes and Runtime.maxMemory() returns maximum memory in bytes.  
  
  
**43) What is the difference between stack and heap in Java?**([answer](http://javarevisited.blogspot.com/2013/01/difference-between-stack-and-heap-java.html))  
Stack and heap are different memory areas in the JVM and they are used for different purposes. The stack is used to hold method frames and local variables while objects are always allocated memory from the heap. The stack is usually much smaller than heap memory and also didn't shared between multiple threads, but heap is shared among all threads in JVM.

[](http://1.bp.blogspot.com/-NZeVo83YJAA/VhDrDO0oWtI/AAAAAAAAD5E/mEek8Ll7NfU/s1600/Difference+between+stack+and+heap+memory+in+Java.gif)

## Basic Java concepts Interview Questions

**44) What's the difference between "a == b" and "a.equals(b)"?** ([answer](http://javarevisited.blogspot.sg/2012/12/difference-between-equals-method-and-equality-operator-java.html))  
The a = b does object reference matching if both a and b are an object and only return true if both are pointing to the same object in the heap space, on the other hand, a.equals(b) is used for logical mapping and its expected from an object to override this method to provide logical equality. For example, String class overrides this equals() method so that you can compare two Strings, which are the different object but contains same letters.  
  
  
**45) What is a.hashCode() used for? How is it related to a.equals(b)?** ([answer](http://javarevisited.blogspot.sg/2011/10/override-hashcode-in-java-example.html))  
hashCode() method returns an int hash value corresponding to an object. It's used in hash based collection classes e.g Hashtable, HashMap, LinkedHashMap and so on. It's very tightly related to equals() method. According to Java specification, two objects which are equal to each other using equals() method must have same hash code.  
  
  
**46) Difference between final, finalize and finally?**([answer](http://javarevisited.blogspot.sg/2012/11/difference-between-final-finally-and-finalize-java.html))  
The final is a modifier which you can apply to variable, methods and classes. If you make a variable final it means its value cannot be changed once initialized. finalize is a method, which is called just before an object is a garbage collected, giving it last chance to resurrect itself, but the call to finalize is not guaranteed. finally is a keyword which is used in exception handling along with try and catch. the finally block is always executed irrespective of whether an exception is thrown from try block or not.  
  
  
**47) What is a compile time constant in Java? What is the risk of using it?**  
public static final variables are also known as a compile time constant, the public is optional there. They are replaced with actual values at compile time because compiler know their value up-front and also knows that it cannot be changed during run-time. One of the problem with this is that if you happened to use a public static final variable from some in-house or third party library and their value changed later than your client will still be using old value even after you deploy a new version of JARs. To avoid that, make sure you compile your program when you upgrade dependency JAR files.

## Java Collections Framework Interview Questions

It also contains Data structure and algorithm Interview question in Java, questions on array, linked list, HashMap, ArrayList, Hashtable, Stack, Queue, PriorityQueue, LinkedHashMap and ConcurrentHashMap.  
  
**48) The difference between List, Set, Map, and Queue in Java?** ([answer](http://java67.blogspot.sg/2013/01/difference-between-set-list-and-map-in-java.html))  
The list is an ordered collection which allows duplicate. It also has an implementation which provides constant time index based access, but that is not guaranteed by List interface. Set is unordered collection which  
  
  
**49) Difference between poll() and remove() method?**  
Both poll() and remove() take out the object from the Queue but if poll() fails then it returns null but if remove fails it throws Exception.  
  
  
**50) The difference between LinkedHashMap and PriorityQueue in Java?** ([answer](http://javarevisited.blogspot.sg/2013/10/what-is-priorityqueue-data-structure-java-example-tutorial.html))  
PriorityQueue guarantees that lowest or highest priority element always remain at the head of the queue, but LinkedHashMap maintains the order on which elements are inserted. When you iterate over a PriorityQueue, iterator doesn't guarantee any order but iterator of LinkedHashMap does guarantee the order on which elements are inserted.  
  
  
**51) Difference between ArrayList and LinkedList in Java?** ([answer](http://java67.blogspot.sg/2012/12/difference-between-arraylist-vs-LinkedList-java.html))  
The obvious difference between them is that ArrrayList is backed by array data structure, supprots random access and LinkedList is backed by linked list data structure and doesn't supprot random access. Accessing an element with the index is O(1) in ArrayList but its O(n) in LinkedList. See the answer for more detailed discussion.  
  
  
**52) What is a couple of ways that you could sort a collection?** ([answer](http://java67.blogspot.sg/2012/07/sort-list-ascending-descending-order-set-arraylist.html))  
You can either use the Sorted collection like TreeSet or TreeMap or you can sort using the ordered collection like a list and using Collections.sort() method.  
  
  
**53) How do you print Array in Java?** ([answer](http://java67.blogspot.sg/2014/03/how-to-print-array-in-java-example-tutorial.html))  
You can print an array by using the Arrays.toString() and Arrays.deepToString() method. Since array doesn't implement toString() by itself, just passing an array to System.out.println() will not print its contents but Arrays.toString() will print each element.

**54) LinkedList in Java is doubly or singly linked list?** (answer)  
It's a doubly linked list, you can check the code in JDK. In Eclipse, you can use the [shortcut](http://javarevisited.blogspot.com/2010/10/eclipse-tutorial-most-useful-eclipse.html), Ctrl + T to directly open this class in Editor.  
  
**55) Which kind of tree is used to implement TreeMap in Java?** (answer)  
A Red Black tree is used to implement TreeMap in Java.

**56) What is the difference between Hashtable and HashMap?** ([answer](http://java67.blogspot.sg/2012/08/5-difference-between-hashtable-hashmap-Java-collection.html))  
There are many differences between these two classes, some of them are following:  
a) Hashtable is a legacy class and present from JDK 1, HashMap was added later.  
b) Hashtable is synchronized and slower but HashMap is not synchronized and faster.  
c) Hashtable doesn't allow null keys but HashMap allows one null key.  
See the answer for more differences between HashMap and Hashtable in Java.  
  
  
**57) How HashSet works internally in Java?** ([answer](http://java67.blogspot.sg/2014/01/how-hashset-is-implemented-or-works-internally-java.html))  
HashSet is internally implemented using an HashMap. Since a Map needs key and value, a default value is used for all keys. Similar to HashMap, HashSet doesn't allow duplicate keys and only one null key, I mean you can only store one null object in HashSet.  
  
  
**58) Write code to remove elements from ArrayList while iterating?** ([answer](http://javarevisited.blogspot.sg/2014/01/ow-to-remove-objects-from-collection-arraylist-java-iterator-traversing.html))  
 Key here is to check whether candidate uses ArrayList's remove() or Iterator's remove(). Here is the [sample code](http://java67.blogspot.com/2015/10/how-to-solve-concurrentmodificationexception-in-java-arraylist.html) which uses right way o remove elements from ArrayList while looping over and avoids ConcurrentModificationException.  
  
  
**59) Can I write my own container class and use it in the for-each loop?**  
Yes, you can write your own container class. You need to implement the Iterable interface if you want to loop over advanced for loop in Java, though. If you implement Collection then you by default get that property.  
  
  
**60) What is default size of ArrayList and HashMap in Java?** ([answer](http://javarevisited.blogspot.sg/2014/07/java-optimization-empty-arraylist-and-Hashmap-cost-less-memory-jdk-17040-update.html))  
As of Java 7 now, default size of ArrayList is 10 and default capacity of HashMap is 16, it must be power of 2. Here is code snippet from ArrayList  and HashMap class :

// from ArrayList.java JDK 1.7

private static final int DEFAULT\_CAPACITY = 10;

//from HashMap.java JDK 7

static final int DEFAULT\_INITIAL\_CAPACITY = 1 **<<** 4; // aka 16

**61) Is it possible for two unequal objects to have the same hashcode?**  
Yes, two unequal objects can have same hashcode that's why collision happen in a hashmap.  
the equal hashcode contract only says that two equal objects must have the same hashcode it doesn't say anything about the unequal object.  
  
**62) Can two equal object have the different hash code?**  
No, thats not possible according to hash code contract.  
  
  
**63) Can we use random numbers in the hashcode() method?** ([answer](http://javarevisited.blogspot.sg/2011/10/override-hashcode-in-java-example.html))  
No, because hashcode of an object should be always same. See the answer to learning more about things to remember while overriding hashCode() method in Java.  
  
  
**64) What is the difference between Comparator and Comparable in Java?** ([answer](http://java67.blogspot.sg/2013/08/difference-between-comparator-and-comparable-in-java-interface-sorting.html))  
The Comparable interface is used to define the  natural order of object while Comparator is used to define custom order. Comparable can be always one, but we can have multiple comparators to define customized order for objects.  
  
**65) Why you need to override hashcode, when you override equals in Java?** ([answer](http://javarevisited.blogspot.sg/2015/01/why-override-equals-hashcode-or-tostring-java.html))  
 Because equals have code contract mandates to override equals and hashcode together .since many container class like HashMap or HashSet depends on hashcode and equals contract.

### Java IO and NIO Interview questions

IO is very important from Java interview point of view. You should have a good knowledge of old Java IO, NIO, and NIO2 alsong with some operating system and disk IO fundamentals. Here are some frequently asked questions form Java IO.  
  
66) In my Java program, I have three sockets? How many threads I will need to handle that?  
  
67) How do you create ByteBuffer in Java?  
  
68) How do you write and read from ByteBuffer in Java?  
  
69) Is Java BIG endian or LITTLE endian?  
  
70) What is the byte order of ByteBuffer?  
  
71) The difference between direct buffer and non-direct buffer in Java? ([answer](http://javarevisited.blogspot.sg/2015/08/difference-between-direct-non-direct-mapped-bytebuffer-nio-java.html))  
  
72) What is the memory mapped buffer in Java? ([answer](http://javarevisited.blogspot.sg/2012/01/memorymapped-file-and-io-in-java.html))  
  
73) What is TCP NO DELAY socket option?  
  
74) What is the difference between TCP and UDP protocol? ([answer](http://javarevisited.blogspot.com/2014/07/9-difference-between-tcp-and-udp-protocol.html))  
  
75) The difference between ByteBuffer and StringBuffer in Java? (answer)

### Java Best Practices Interview question

Contains best practices from different parts of Java programming e.g. Collections, String, IO, Multi-threading, Error and Exception handling, design patterns etc. This section is mostly for experience Java developer, technical lead,  AVP, team lead and coders who are responsible for products. If you want to create quality products you must know and follow the best practices.  
  
**76) What best practices you follow while writing multi-threaded code in Java?** ([answer](http://javarevisited.blogspot.com/2015/05/top-10-java-multithreading-and.html))  
Here are couple of best practices which I follow while writing concurrent code in Java:  
a) Always name your thread, this will help in debugging.  
b) minimize the scope of synchronization, instead of making whole method synchronized, only critical section should be synchronized.  
c) prefer volatile over synchronized if you can can.  
e) use higher level concurrency utilities instead of waitn() and notify for inter thread communication e.g. BlockingQueue, CountDownLatch and Semeaphore.  
e) Prefer concurrent collection over synchronized collection in Java. They provide better scalability.  
  
  
**77) Tell me few best practices you apply while using Collections in Java?** (answer)  
Here are couple of best practices I follow while using Collectionc classes from Java:  
a) Always use the right collection e.g. if you need non-synchronized list then use ArrayList and not Vector.  
b) Prefer concurrent collection over synchronized collection because they are more scalable.  
c) Always use interface to a represent and access a collection e.g. use List to store ArrayList, Map to store HashMap and so on.  
d) Use iterator to loop over collection.  
e) Always use generics with collection.  
  
  
**78) Can you tell us at least 5 best practice you use while using threads in Java?** ([answer](http://java67.blogspot.com/2014/01/10-points-about-thread-and-javalangthread-in-java.html))  
This is similar to the previous question and you can use the answer given there. Particularly with thread, you should:  
a) name your thread  
b) keep your task and thread separate, use Runnable or Callable with thread pool executor.  
c) use thread pool  
d) use volatile to indicate compiler about ordering, visibility, and atomicity.  
e) avoid thread local variable because incorrect use of ThreadLocal class in Java can create a memory leak.  
Look there are many best practices and I give extra points to the developer which bring something new, something even I don't know. I make sure to ask this question to Java developers of 8 to 10 years of experience just to gauge his hands on experience and knowledge.  
  
  
**79) Name 5 IO best practices?** (answer)  
IO is very important for performance of your Java application. Ideally you should avoid IO in critical path of your application. Here are couple of Java IO best practices you can follow:

a) Use buffered IO classes instead of reading individual bytes and char.

b) Use classes from NIO and NIO2

c) Always close streams in finally block or use try-with-resource statements.

d) use memory mapped file for faster IO.

If a Java candidate doesn't know about IO and NIO, especially if he has at least 2 to 4 years of experience, he needs some reading.  
  
  
**80) Name 5 JDBC best practices your follow?** ([answer](http://javarevisited.blogspot.sg/2012/08/top-10-jdbc-best-practices-for-java.html))  
Another good Java best practices for experienced Java developer of 7 to 8 years experience. Why it's important? because they are the ones which set the trend in the code and educate junior developers. There are many best practices and you can name as per your confort and conviniece. Here are some of the more common ones:  
a) use batch statement for inserting and updating data.  
b) use PreparedStatement to avoid SQL exception and better performance.  
c) use database connection pool  
d) access resultset using column name instead of column indexes.  
e) Don't generate dynamic SQL by concatenating String with user input.  
  
  
**81) Name couple of method overloading best practices in Java?** ([answer](http://javarevisited.blogspot.sg/2013/01/java-best-practices-method-overloading-constructor.html))  
Here are some best practices you can follow while overloading a method in Java to avoid confusion with auto-boxing:  
a) Don't overload method where one accepts int and other accepts Integer.  
b) Don't overload method where number of argument is same and only order of argument is different.  
c) Use varargs after overloaded methods has more than 5 arguments.

### Date, Time and Calendar Interview questions in Java

**82) Does SimpleDateFormat is safe to use in the multi-threaded program?** ([answer](http://javarevisited.blogspot.sg/2012/03/simpledateformat-in-java-is-not-thread.html))  
No, unfortunately, DateFormat and all its implementations including SimpleDateFormat is not thread-safe, hence should not be used in the multi-threaded program until external thread-safety measures are applied e.g. confining SimpleDateFormat object into a ThreadLocal variable. If you don't do that, you will get an incorrect result while parsing or formatting dates in Java. Though, for all practical date time purpose, I highly recommend **joda-time** library.  
  
  
**83) How do you format a date in Java? e.g. in the ddMMyyyy format?** ([answer](http://javarevisited.blogspot.com/2011/09/convert-date-to-string-simpledateformat.html))  
You can either use SimpleDateFormat class or joda-time library to format date in Java. DateFormat class allows you to format date on many popular formats. Please see the answer for code samples to format date into different formats e.g. dd-MM-yyyy or ddMMyyyy.  
  
  
84) How do you show timezone in formatted date in Java? ([answer](http://java67.blogspot.sg/2013/01/how-to-format-date-in-java-simpledateformat-example.html))  
  
85) The difference between java.util.Date and java.sql.Date in Java? ([answer](http://java67.blogspot.sg/2014/02/how-to-convert-javautildate-to-javasqldate-example.html))  
  
86) How to you calculate the difference between two dates in Java? ([program](http://javarevisited.blogspot.sg/2015/07/how-to-find-number-of-days-between-two-dates-in-java.html))  
  
87) How do you convert a String(YYYYMMDD) to date in Java? ([answer](http://java67.blogspot.sg/2014/12/string-to-date-example-in-java-multithreading.html))

### Unit testing JUnit Interview questions

89) How do you test static method? (answer)  
You can use PowerMock library to test static methods in Java.  
  
90) How to do you test a method for an exception using JUnit? ([answer](http://javarevisited.blogspot.sg/2013/04/JUnit-tutorial-example-test-exception-thrown-by-java-method.html))  
  
91) Which unit testing libraries you have used for testing Java programs? (answer)  
  
92) What is the difference between @Before and @BeforeClass annotation? ([answer](http://javarevisited.blogspot.sg/2013/04/JUnit-tutorial-example-test-exception-thrown-by-java-method.html))

### Programming and Coding Questions

93) How to check if a String contains only numeric digits? ([solution](http://java67.blogspot.com/2014/01/java-regular-expression-to-check-numbers-in-String.html))  
  
94) How to write LRU cache in Java using Generics? (answer)  
  
95) Write a Java program to convert bytes to long? (answer)  
  
96) How to reverse a String in Java without using StringBuffer? ([solution](http://java67.blogspot.com/2012/12/how-to-reverse-string-in-java-stringbuffer-stringbuilder.htm))  
  
97) How to find the word with the highest frequency from a file in Java? ([solution](http://java67.blogspot.com/2015/10/java-program-to-find-repeated-words-and-count.html))  
  
98) How do you check if two given String are anagrams? ([solution](http://javarevisited.blogspot.sg/2013/03/Anagram-how-to-check-if-two-string-are-anagrams-example-tutorial.html))  
  
99) How to print all permutation of a String in Java? ([solution](http://javarevisited.blogspot.com/2015/08/how-to-find-all-permutations-of-string-java-example.html))  
  
100) How do you print duplicate elements from an array in Java? ([solution](http://javarevisited.blogspot.com/2015/06/3-ways-to-find-duplicate-elements-in-array-java.html))  
  
101) How to convert String to int in Java? ([solution](http://java67.blogspot.com/2015/08/2-ways-to-parse-string-to-int-in-java.html))  
  
102) How to swap two integers without using temp variable? ([solution](http://java67.blogspot.com/2015/08/how-to-swap-two-integers-without-using.html))

### Java Interview questions from OOP and Design Patterns

It contains Java Interview questions from SOLID design principles, OOP fundamentals e.g. class, object, interface, Inheritance, Polymorphism, Encapsulation, and Abstraction as well as more advanced concepts like Composition, Aggregation, and Association. It also contains questions from GOF design patterns.  
  
**103) What is the interface? Why you use it if you cannot write anything concrete on it?**  
The interface is used to define API. It tells about the contract your classes will follow. It also supports abstraction because a client can use interface method to leverage multiple implementations e.g. by using List interface you can take advantage of [random access of ArrayList](http://javarevisited.blogspot.com/2015/07/java-arraylist-tutorial.html) as well as flexible insertion and deletion of LinkedList. The interface doesn't allow you to write code to keep things abstract but from Java 8 you can declare static and default methods inside interface which are concrete.  
  
  
**104) The difference between abstract class and interface in Java?** ([answer](http://javarevisited.blogspot.sg/2013/05/difference-between-abstract-class-vs-interface-java-when-prefer-over-design-oops.html))  
There are multiple differences between abstract class and interface in Java, but the most important one is Java's restriction on allowing a class to extend just one class but allows it to implement multiple interfaces. An abstract class is good to define default behavior for a family of class, but the interface is good to define Type which is later used to leverage Polymorphism. Please check the answer for a more thorough discussion of this question.  
  
  
**105) Which design pattern have you used in your production code? apart from Singleton?**  
This is something you can answer from your experience. You can generally say about dependency injection, factory pattern, decorator pattern or observer pattern, whichever you have used. Though be prepared to answer follow-up question based upon the pattern you choose.  
  
  
**106) Can you explain Liskov Substitution principle?** ([answer](http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html))  
This is one of the toughest questions I have asked in Java interviews. Out of 50 candidates, I have almost asked only 5 have managed to answer it. I am not posting an answer to this question as I like you to do some research, practice and spend some time to understand this confusing principle well.  
  
  
**107) What is Law of Demeter violation? Why it matters?** ([answer](http://javarevisited.blogspot.com/2014/05/law-of-demeter-example-in-java.html))  
Believe it or not, Java is all about application programming and structuring code. If  you have good knowledge of common coding best practices, patterns and what not to do than only you can write quality code.  Law of Demeter suggests you "talk to friends and not stranger", hence used to reduce coupling between classes.  
  
  
**108) What is Adapter pattern? When to use it?**  
Another frequently asked Java design pattern questions. It provides interface conversion. If your client is using some interface but you have something else, you can write an Adapter to bridge them together. This is good for Java software engineer having 2 to 3 years experience because the question is neither difficult nor tricky but requires knowledge of OOP design patterns.  
  
  
**109) What is "dependency injection" and "inversion of control"? Why would someone use it?** ([answer](http://javarevisited.blogspot.sg/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html))  
  
**110) What is an abstract class? How is it different from an interface? Why would you use it?** ([answer](http://java67.blogspot.sg/2014/06/why-abstract-class-is-important-in-java.html))  
One more classic question from Programming Job interviews, it is as old as chuck Norris. An abstract class is a class which can have state, code and implementation, but an interface is a contract which is totally abstract. Since I have answered it many times, I am only giving you the gist here but you should read the article linked to answer to learn this useful concept in much more detail.  
  
  
**111) Which one is better constructor injection or setter dependency injection?** ([answer](http://javarevisited.blogspot.sg/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html))  
Each has their own advantage and disadvantage. Constructor injection guaranteed that class will be initialized with all its dependency, but setter injection provides flexibility to set an optional dependency. Setter injection is also more readable if you are using an XML file to describe dependency. Rule of thumb is to use constructor injection for mandatory dependency and use setter injection for optional dependency.  
  
  
**112) What is difference between dependency injection and factory design pattern?** ([answer](http://javarevisited.blogspot.sg/2015/06/difference-between-dependency-injection.html))  
Though both patterns help to take out object creation part from application logic, use of dependency injection results in cleaner code than factory pattern. By using dependency injection, your classes are nothing but POJO which only knows about dependency but doesn't care how they are acquired. In the case of factory pattern, the class also needs to know about factory to acquire dependency. hence, DI results in more testable classes than factory pattern. Please see the answer for a more detailed discussion on this topic.  
  
  
**113) Difference between Adapter and Decorator pattern?** ([answer](http://javarevisited.blogspot.sg/2015/01/adapter-vs-decorator-vs-facade-vs-proxy-pattern-java.html))  
Though the structure of Adapter and Decorator pattern is similar, the difference comes on the intent of each pattern. The adapter pattern is used to bridge the gap between two interfaces, but Decorator pattern is used to add new functionality into the class without the modifying existing code.  
  
  
**114) Difference between Adapter and Proxy Pattern?** ([answer](http://javarevisited.blogspot.sg/2015/01/adapter-vs-decorator-vs-facade-vs-proxy-pattern-java.html))  
Similar to the previous question, the difference between Adapter and Proxy patterns is in their intent. Since both Adapter and Proxy pattern encapsulate the class which actually does the job, hence result in the same structure, but Adapter pattern is used for interface conversion while the Proxy pattern is used to add an extra level of indirection to support distribute, controlled or intelligent access.  
  
  
**115) What is Template method pattern?** (answer)  
Template pattern provides an outline of an algorithm and lets you configure or customize its steps. For examples, you can view a sorting algorithm as a template to sort object. It defines steps for sorting but let you configure how to compare them using Comparable or something similar in another language. The method which outlines the algorithms is also known as template method.  
  
  
**116) When do you use Visitor design pattern?** (answer)  
The visitor pattern is a solution of problem where you need to add operation on a class hierarchy but without touching them. This pattern uses double dispatch to add another level of indirection.  
  
  
**117) When do you use Composite design pattern?** (answer)  
Composite design pattern arranges objects into tree structures to represent part-whole hierarchies. It allows clients treat individual objects and container of objects uniformly. Use Composite pattern when you want to represent part-whole hierarchies of objects.

**118) The difference between Inheritance and Composition?** ([answer](http://javarevisited.blogspot.sg/2015/06/difference-between-inheritance-and-Composition-in-Java-OOP.html))  
Though both allows code reuse, Composition is more flexible than Inheritance because it allows you to switch to another implementation at run-time. Code written using Composition is also easier to test than code involving inheritance hierarchies.  
  
  
**119) Describe overloading and overriding in Java?** ([answer](http://java67.blogspot.sg/2012/09/difference-between-overloading-vs-overriding-in-java.html))  
Both overloading and overriding allow you to write two methods of different functionality but with the same name, but overloading is compile time activity while overriding is run-time activity. Though you can overload a method in the same class, but you can only override a method in child classes. Inheritance is necessary for overriding.  
  
  
**120) The difference between nested public static class and a top level class in Java?** ([answer](http://javarevisited.blogspot.sg/2012/12/inner-class-and-nested-static-class-in-java-difference.html))  
You can have more than one nested public static class inside one class, but you can only have one top-level public class in a Java source file and its name must be same as the name of Java source file.  
  
  
**121) Difference between Composition, Aggregation and Association in OOP?** ([answer](http://javarevisited.blogspot.sg/2014/02/ifference-between-association-vs-composition-vs-aggregation.html))  
If two objects are related to each other, they are said to be associated with each other. Composition and Aggregation are two forms of association in object-oriented programming. The composition is stronger association than Aggregation. In Composition, one object is OWNER of another object while in Aggregation one object is just USER of another object. If an object A is composed of object B then B doesn't exist if A ceased to exists, but if object A is just an aggregation of object B then B can exists even if A ceased to exist.  
  
  
**122) Give me an example of design pattern which is based upon open closed principle?** ([answer](http://javarevisited.blogspot.sg/2011/11/great-example-of-open-closed-design.html))  
This is one of the practical questions I ask experienced Java programmer. I expect them to know about OOP design principles as well as patterns. Open closed design principle asserts that your code should be open for extension but closed for modification. Which means if you want to add new functionality, you can add it easily using the new code but without touching already tried and tested code.  There are several design patterns which are based upon open closed design principle e.g. [Strategy pattern](http://java67.blogspot.com/2014/12/strategy-pattern-in-java-with-sample.html) if you need a new strategy, just implement the interface and configure, no need to modify core logic. One working example is Collections.sort() method which is based on Strategy pattern and follows the open-closed principle, you don't modify sort() method to sort a new object, what you do is just implement Comparator in your own way.  
  
  
**123) Difference between Abstract factory and Prototype design pattern?** (answer)  
This is the practice question for you, If you are feeling bored just reading and itching to write something, why not write the answer to this question. I would love to see an example the, which should answer where you should use the Abstract factory pattern and where is the Prototype pattern is more suitable.  
  
  
**124) When do you use Flyweight pattern?** (answer)  
This is another popular question from the design pattern. Many Java developers with 4 to 6 years of experience know the definition but failed to give any concrete example. Since many of you might not have used this pattern, it's better to look examples from JDK. You are more likely have used them before and they are easy to remember as well. Now let's see the answer.  
Flyweight pattern allows you to share object to support large numbers without actually creating too many objects. In order to use Flyweight pattern, you need to make your object Immutable so that they can be safely shared. String pool and pool of Integer and Long object in JDK are good examples of Flyweight pattern.

### Miscellaneous Java Interview Questions

It contains XML Processing in Java Interview question, JDBC Interview question, Regular expressions Interview questions, Java Error and Exception Interview Questions, Serialization,  
  
**125) The difference between nested static class and top level class?** ([answer](http://java67.blogspot.sg/2012/10/nested-class-java-static-vs-non-static-inner.html))  
One of the fundamental questions from Java basics. I ask this question only to junior Java developers of 1 to 2 years of experience as it's too easy for an experience Java programmers. The answer is simple, a public top level class must have the same name as the name of the source file, there is no such requirement for nested static class. A nested class is always inside a top level class and you need to use the name of the top-level class to refer nested static class e.g. HashMap.Entry is a nested static class, where HashMap is a top level class and Entry is nested static class.  
  
  
**126) Can you write a regular expression to check if String is a number?** ([solution](http://javarevisited.blogspot.sg/2012/10/regular-expression-example-in-java-to-check-String-number.html))  
If you are taking Java interviews then you should ask at least one question on the regular expression. This clearly [differentiates an average programmer with a good programmer](http://javarevisited.blogspot.com/2015/05/how-to-differentiate-between-average.html). Since one of the traits of a good developer is to know tools, regex is the best tool for searching something in the log file, preparing reports etc. Anyway, answer to this question is, a numeric String can only contain digits i.e. 0 to 9 and + and - sign that too at start of the String, by using this information you can write following regular expression to check if given String is number or not  
  
  
**127) The difference between checked and unchecked Exception in Java?** ([answer](http://java67.blogspot.sg/2012/12/difference-between-runtimeexception-and-checked-exception.html))  
checked exception is checked by the compiler at compile time. It's mandatory for a method to either handle the checked exception or declare them in their throws clause. These are the ones which are a sub class of Exception but doesn't descend from RuntimeException. The unchecked exception is the descendant of RuntimeException and not checked by the compiler at compile time. This question is now becoming less popular and you would only find this with interviews with small companies, both investment banks and startups are moved on from this question.  
  
  
**128) The difference between throw and throws in Java?** ([answer](http://javarevisited.blogspot.sg/2012/02/difference-between-throw-and-throws-in.html))  
the throw is used to actually throw an instance of java.lang.Throwable class, which means you can throw both Error and Exception using throw keyword e.g.

throw new IllegalArgumentException("size must be multiple of 2")

On the other hand, throws is used as part of method declaration and signals which kind of exceptions are thrown by this method so that its caller can handle them. It's mandatory to declare any unhandled checked exception in **throws** clause in Java. Like the previous question, this is another frequently asked Java interview question from errors and exception topic but too easy to answer.  
  
  
**129) The difference between Serializable and Externalizable in Java?** ([answer](http://javarevisited.blogspot.sg/2012/01/serializable-externalizable-in-java.html))  
This is one of the frequently asked questions from Java Serialization. The interviewer has been asking this question since the day Serialization was introduced in Java, but yet only a few good candidate can answer this question with some confidence and practical knowledge. Serializable interface is used to make Java classes serializable so that they can be transferred over network or their state can be saved on disk, but it leverages default serialization built-in JVM, which is expensive, fragile and not secure. Externalizable allows you to fully control the Serialization process, specify a custom binary format and add more security measure.  
  
  
**130) The difference between DOM and SAX parser in Java?** ([answer](http://javarevisited.blogspot.sg/2011/12/difference-between-dom-and-sax-parsers.html))  
Another common Java question but from XML parsing topic. It's rather simple to answer and that's why many interviewers prefers to ask this question on the telephonic round. DOM parser loads the whole XML into memory to create a tree based DOM model which helps it quickly locate nodes and make a change in the structure of XML while SAX parser is an event based parser and doesn't load the whole XML into memory. Due to this reason DOM is faster than SAX but require more memory and not suitable to parse large XML files.  
  
  
**131) Tell me 3 features introduced on JDK 1.7?** ([answer](http://javarevisited.blogspot.sg/2014/04/10-jdk-7-features-to-revisit-before-you.html))  
This is one of the good questions I ask to check whether the candidate is aware of recent development in Java technology space or not. Even though JDK 7 was not a big bang release like JDK 5 or JDK 8, it still has a lot of good feature to count on e.g. try-with-resource statements, which free you from closing streams and resources when you are done with that, Java automatically closes that. Fork-Join pool to implement something like the Map-reduce pattern in Java. Allowing String variable and literal into switch statements. Diamond operator for improved type inference, no need to declare generic type on the right-hand side of variable declaration anymore, results in more readable and succinct code. Another worth noting feature introduced was improved exception handling e.g. allowing you to catch multiple exceptions in the same catch block.  
  
  
**132) Tell me 5 features introduced in JDK 1.8?** ([answer](http://javarevisited.blogspot.sg/2014/02/10-example-of-lambda-expressions-in-java8.html))  
This is the follow-up question of the previous one. Java 8 is path breaking release in Java's history, here are the top 5 features from JDK 8 release

* **Lambda expression**, which allows you pass an anonymous function as object.
* **Stream API**, take advantage of multiple cores of modern CPU and allows you to write succinct code.
* **Date and Time API**, finally you have a solid and easy to use date and time library right into JDK
* **Extension methods**, now you can have static and default method into your interface
* **Repeated annotation**, allows you apply the same annotation multiple times on a type

**133) What is the difference between Maven and ANT in Java?** ([answer](http://javarevisited.blogspot.sg/2015/01/difference-between-maven-ant-jenkins-and-hudson.html))  
Another great question to check the all round knowledge of Java developers. It's easy to answer questions from core Java but when you ask about setting things up, building Java artifacts, many Java software engineer struggles. Coming back to the answer of this question, Though both are build tool and used to create Java application build, Maven is much more than that. It provides standard structure for Java project based upon "convention over configuration" concept and automatically manage dependencies (JAR files on which your application is dependent) for Java application. Please see the answer for more differences between Maven and ANT tool.  
  
  
That's all guys, **lots of Java Interview questions?** isn't it? I am sure if you can answer this list of Java questions you can easily crack any core Java or advanced Java interview. Though I have not included questions from Java EE or J2EE topics e.g. Servlet, JSP, JSF, JPA, JMS, EJB or any other Java EE technology or from major web frameworks like Spring MVC, Struts 2.0, Hibernate or both SOAP and RESTful web services, it's still useful for Java developers preparing for Java web developer position, because every Java interview starts with questions from fundamentals and JDK API. If you think, I have missed any popular Java question here and you think it should be in this list then feel free to suggest me. My goal is to create the best list of Java Interview Questions with latest and greatest question from recent interviews.

Read more: <http://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html#ixzz4esBrtcVa>

### **Top Java design pattern questions and answers**

Here is my list of t*op 10 design pattern interview question in Java*. I have also provided answer of those Java design pattern question as link. no matter which level of Java interview are you going e.g. programmer, software engineer, senior software engineer in Java, you can expect few question from Java design pattern.

**1. When to use Strategy Design Pattern in Java?**  
[Java design pattern interview question and answers for senior and experience programmer](http://3.bp.blogspot.com/-1lzFJzIgaHk/UF2Ci6kY5pI/AAAAAAAAAes/OYiM7r-DHzc/s1600/17.jpg)Strategy pattern in quite useful for implementing set of related algorithms e.g. compression algorithms, filtering strategies etc. Strategy design pattern allows you to create Context classes, which uses Strategy implementation classes for applying business rules. This pattern follow open closed design principle and quite useful in Java. One example of Strategy pattern from JDK itself is a Collections.sort() method and [Comparator interface](http://java67.blogspot.sg/2012/10/how-to-sort-object-in-java-comparator-comparable-example.html), which is a strategy interface and defines strategy for comparing objects. Because of this pattern, we don't need to modify sort() method (closed for modification) to compare any object, at same time we can implement Comparator interface to define new comparing strategy (open for extension).  
  
**2. What is Observer design pattern in Java? When do you use Observer pattern in Java?**  
This is one of the most common Java design pattern interview question. Observer pattern is based upon notification, there are two kinds of object Subject and Observer. Whenever there is change on subject's state observer will receive notification. See [What is Observer design pattern in Java with real life example](http://javarevisited.blogspot.sg/2011/12/observer-design-pattern-java-example.html) for more details.  
  
**3. Difference between Strategy and State design Pattern in Java?**  
This is an interesting Java design pattern interview questions as both Strategy and State pattern has same structure. If you look at UML class diagram for both pattern they look exactly same, but there intent is totally different. State design pattern is used to define and mange state of object, while Strategy pattern is used to define a set of interchangeable algorithm and let's client to choose one of them. So Strategy pattern is a client driven pattern while Object can manage there state itself.  
  
**4. What is decorator pattern in Java? Can you give an example of Decorator pattern?**  
Decorator pattern is another popular java design pattern question which is common because of its heavy usage in java.io package. BufferedReader and BufferedWriter are good example of decorator pattern in Java. See [How to use Decorator pattern in Java](http://javarevisited.blogspot.com/2011/11/decorator-design-pattern-java-example.html) fore more details.  
  
**5. When to use Composite design Pattern in Java? Have you used previously in your project?**  
This design pattern question is asked on Java interview not just to check familiarity with Composite pattern but also, whether candidate has real life experience or not. *Composite pattern* is also a core Java design pattern, which allows you to treat both whole and part object to treat in similar way. Client code, which deals with Composite or individual object doesn't differentiate on them, it is possible because Composite class also implement same interface as there individual part. One of the good example of Composite pattern from JDK is JPanel class, which is both Component and Container.  When paint() method is called on JPanel, it internally called paint() method of individual components and let them draw themselves. On second part of this design pattern interview question, be truthful, if you have used then say yes, otherwise say that you are familiar with concept and used it by your own. By the way always remember, giving an example from your project creates better impression.  
  
**6. What is Singleton pattern in Java?**  
Singleton pattern in Java is a pattern which allows only one instance of Singleton class available in whole application. java.lang.Runtime is good example of Singleton pattern in Java. There are lot's of follow up questions on Singleton pattern see [10 Java singleton interview question answers](http://javarevisited.blogspot.com/2011/03/10-interview-questions-on-singleton.html) for those followups   
  
**7. Can you write thread-safe Singleton in Java?**  
There are multiple ways to write thread-safe singleton in Java e.g by writing singleton using double checked locking, by using static Singleton instance initialized during [class loading.](http://javarevisited.blogspot.sg/2012/07/when-class-loading-initialization-java-example.html) By the way using Java enum to create thread-safe singleton is most simple way. See [Why Enum singleton is better in Java](http://javarevisited.blogspot.gr/2012/07/why-enum-singleton-are-better-in-java.html) for more details.  
  
**8. When to use Template method design Pattern in Java?**Template pattern is another popular core Java design pattern interview question. I have seen it appear many times in real life project itself. Template pattern outlines an algorithm in form of template method and let subclass implement individual steps. Key point to mention, while answering this question is that template method should be final, so that subclass can not override and change steps of algorithm, but same time individual step should be abstract, so that child classes can implement them.  
  
**9. What is Factory pattern in Java? What is advantage of using static factory method to create object?**  
Factory pattern in Java is a creation Java design pattern and favorite on many Java interviews.Factory pattern used to create object by providing static factory methods. There are many advantage of providing factory methods e.g. caching immutable objects, easy to introduce new objects etc. See [What is Factory pattern in Java and benefits](http://javarevisited.blogspot.sg/2011/12/factory-design-pattern-java-example.html) for more details.  
  
**10. Difference between Decorator and Proxy pattern in Java?**Another tricky Java design pattern question and trick here is that both Decorator and Proxy implements interface of the object they decorate or encapsulate. As I said, many Java design pattern can have similar or exactly same structure but they differ in there intent. Decorator pattern is used to implement functionality on already created object, while Proxy pattern is used for controlling access to object. One more difference between Decorator and Proxy design pattern is that, Decorator doesn't create object, instead it get object in it's constructor, while Proxy actually creates objects.  
  
**11. When to use Setter and Constructor Injection in Dependency Injection pattern?**  
Use Setter injection to provide optional dependencies of an object, while use Constructor injection to provide mandatory dependency of an object, without which it can not work. This question is related to [Dependency Injection design pattern](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html) and mostly asked in context of Spring framework, which is now become an standard for developing Java application. Since Spring provides IOC container, it also gives you way to specify dependencies either by using setter methods or constructors. You can also take a look my [previous post](http://javarevisited.blogspot.com/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html) on same topic.  
  
  
**12. What is difference between Factory and Abstract factory in Java**  
see  [here](http://javarevisited.blogspot.sg/2011/04/top-20-core-java-interview-questions.html) to answer this Java design pattern interview question.   
  
  
**13. When to use Adapter pattern in Java? Have you used it before in your project?**  
Use Adapter pattern when you need to make two class work with incompatible interfaces. Adapter pattern can also be used to encapsulate third party code, so that your application only depends upon Adapter, which can adapt itself when third party code changes or you moved to a different third party library. By the way this Java design pattern question can also be asked by providing actual scenario.  
 **14. Can you write code to implement producer consumer design pattern in Java?**  
Producer consumer design pattern is a concurrency design pattern in Java which can be implemented using multiple way. if you are working in Java 5 then its better to use Concurrency util to implement producer consumer pattern instead of plain old [wait and notify in Java](http://javarevisited.blogspot.sg/2011/05/wait-notify-and-notifyall-in-java.html).  Here is a good example of implementing [producer consumer problem using BlockingQueue in Java](http://javarevisited.blogspot.sg/2012/02/producer-consumer-design-pattern-with.html).  
  
**15. What is Open closed design principle in Java?**  
Open closed design principle is one of the SOLID principle defined by Robert C. Martin, popularly known as Uncle Bob. This principle advices that a code should be open for extension but close for modification. At first this may look conflicting but once you explore power of polymorphism, you will start finding patterns which can provide stability and flexibility of this principle. One of the key example of this is State and Strategy design pattern, where Context class is closed for modification and new functionality is provided by writing new code by implementing new state of strategy. See [this](http://javarevisited.blogspot.com/2011/11/great-example-of-open-closed-design.html) article to know more about Open closed principle.  
  
**16. What is Builder design pattern in Java? When do you use Builder pattern ?**  
Builder pattern in Java is another creational design pattern in Java and often asked in Java interviews because of its specific use when you need to build an object which requires multiple properties some optional and some mandatory. See [When to use Builder pattern in Java](http://javarevisited.blogspot.sg/2012/06/builder-design-pattern-in-java-example.html) for more details    
  
**17. Can you give an example of  SOLID design principles in Java?**  
There are lots of SOLID design pattern which forms acronym SOLID, read this [list of SOLID design principles for Java programmer](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html)  to answer this Java interview question.  
 **18. What is difference between Abstraction and Encapsulation in Java?**  
I have already covered answer of this Java interview question in my previous post as [Difference between encapsulation and abstraction in Java](http://java67.blogspot.sg/2012/08/difference-between-abstraction-and-encapsulation-java-oops.html). See there to answer this question.   
  
  
This was my list of **10 popular design pattern interview question in Jav**a. I have not included MVC (Model View Controller) design pattern because that is more specific to J2EE and [Servlet JSP interview](http://javarevisited.blogspot.sg/2011/09/servlet-interview-questions-answers.html), but if you are going for any Java interview which demands experience in J2EE than you must prepare MVC design pattern. That's all on Java design pattern interview question and answers. Please let us know if you have any other interesting question on Java design pattern.

Read more: <http://www.java67.com/2012/09/top-10-java-design-pattern-interview-question-answer.html#ixzz4esCOZ0zi>

### **Servlet Questions Asked in Interview**

**Question 1: In** web**.**xml **file   <load-on-startup>1</load-on-startup> is defined between <servlet></servlet> tag what does it means.** ([detailed answer](http://javarevisited.blogspot.com/2011/12/load-on-startup-servlet-webxml-example.html))

Ans**:** whenever we request for any servlet the servlet container will initialize the servlet and load it which is defined in our config file called web.xml by default it will not initialize when our context is loaded .defining like this <load-on-startup>1</load-on-startup> is also known as pre-initialization of servlet means now the servlet for which we have defined this tag has been initialized in starting when context is loaded before getting any request.When this servlet question was asked to me in an interview few years back , I was not even aware of this element but this questions pointed me to look DTD of web.xml and understand other elements as well.

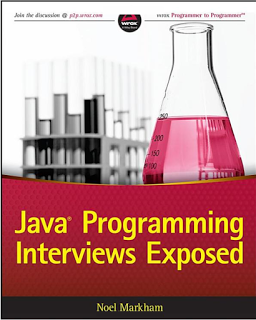
**Question 2: How can we create deadlock condition on our servlet?**([detailed answer](http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html))

Ans: one simple way to call doPost() method inside doGet() and doGet()method inside doPost() it will create deadlock situation for a servlet. This is rather a simple servlet interview questions but yet tricky if you don’t think of it 

**Question 3: For initializing a servlet can we use a constructor in place of init()?**([detailed answer](http://java67.blogspot.com/2016/02/can-servlet-have-constructor-in-java.html))

Ans: No, we can not use constructor for initializing a servlet because for initialization we need an object of servletConfig using this object we get all the parameter which are defined in deployment descriptor for initializing a servlet and in servlet class we have only default constructor according to older version of java so if we want to pass a Config object we don’t have parametrized constructor and apart from this servlet is loaded and initialized by container so it's a job of container to call the method according to servlet specification they have lifecycle method so init() method is called firstly.

More important Java doesn't allow interfaces to declare constructors. These kinds of *servlet interview questions* are quite popular on service based companies who just want to dig one level more. You can also refer [Programming Interviews Exposed](http://www.amazon.com/dp/1118261364/?tag=javamysqlanta-20) for more of such questions.

[](http://www.amazon.com/dp/1118261364/?tag=javamysqlanta-20)

**Question 4: Why super.init (config) is the first statement inside init(config) method**. ([detailed answer](http://javarevisited.blogspot.com/2015/02/constructor-vs-init-method-in-servlet.html))

Ans: This will be the first statement if we are overriding the init(config) method by this way we will store the config object for future reference and we can use by getServletConfig()  to get information about config object if will not do this config object will be lost and we have only one way to get config object because servlet pass config object only in init method . Without doing this if we call the ServletConfig method will get **NullPointerException.**

**Question5: Can we call destroy() method inside the init() method is yes what will happen?**(detailed answer)

Ans: Yes we can call like this but  if we have not overridden this method container will call the default method and nothing will happen.after calling this if any we have overridden the method then the code written inside is executed.

**Question 6: How can we refresh servlet on client and server side automatically?**([detailed answer](http://java67.blogspot.com/2012/10/servlet-jsp-interview-questions-answer-faq-experience.html))

Ans: On the client side we can use Meta HTTP refresh and server side we can use server push.

[](http://3.bp.blogspot.com/-UCqimFIJ5yA/TeJPeOkzKUI/AAAAAAAAAK0/Mv7GFHeydf0/s1600/golkunda+fort+.jpg)

**Question 7: How can you get the information about one servlet context in another servlet**? (detailed answer)

Ans: In context object we can set the attribute which we want on another servlet and we can get that attribute using their name on another servlet.

Context.setAttribute (“name”,” value”)

Context.getAttribute (“name”)

**Question 8: Why we need to implement Single Thread model in the case of Servlet.**(detailed answer)

Ans: In J2EE we can implement our servlet in two different ways either by using:

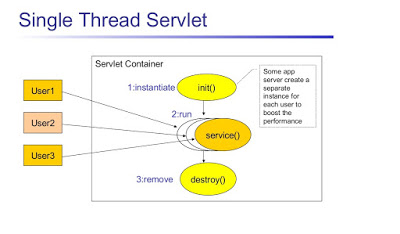
1. Single Thread Model

2. Multithread Model

Depending upon our scenario, if we have implemented single thread means only one instance is going handle one request at a time no two thread will concurrently execute service method of the servlet.

**The example** in banking accounts where sensitive data is handled mostly this scenario was used this interface is deprecated in Servlet API version 2.4.

As the name signifies multi-thread means a servlet is capable of handling multiple requests at the same time. This servlet interview question was quite popular few years back on entry level but now it's losing its shine.

[](https://3.bp.blogspot.com/-mU5exqNhY10/VvSxsvfhYXI/AAAAAAAAFTs/JuIE2-EdbqcId_MUxFlXXtyesXWL1QS0g/s1600/Single+Thread+Servlet.jpg)

**Question 9: What is servlet collaboration?**([detailed answer](http://java67.blogspot.com/2016/01/6-difference-between-include-directive-and-include-action-in-jsp.html))

Ans communication between two servlets is called servlet collaboration which is achieved by 3 ways.

**1. RequestDispatchers include () and forward() method .**

**2. Using** [**sendRedirect()**](http://javarevisited.blogspot.com/2011/09/sendredirect-forward-jsp-servlet.html)**method of Response object.**

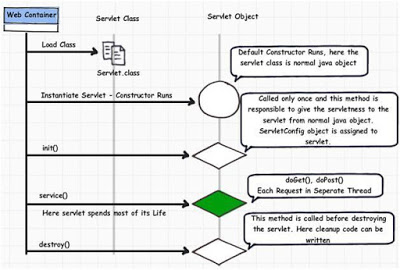
**3. Using servlet Context methods**

**Question 10: What is the difference between ServletConfig and ServletContext?**([detailed answer](http://java67.blogspot.com/2012/09/difference-between-servletconfig-and-servletcontext-j2ee-jsp.html))

Ans: **ServletConfig** as the name implies provide the information about the configuration of a servlet which is defined inside the web.xml file or we can say deployment descriptor.its a specific object for each servlet.

**ServletContext** is an application specific object which is shared by all the servlet belongs to one application in one JVM .this is a single object which represents our application and all the servlet access application specific data using this object.servlet also use their method to communicate with the container.

**Question 11: Explain Servlet Life Cycle in Java EE environment?**  
A picture is worth thousand words, here is a diagram which explains the Servlet life cycle:

[](https://2.bp.blogspot.com/-C7jH3iUVBKc/VvSxXrmNNCI/AAAAAAAAFTw/GcQna15jEqMRmcvoSRbnwlM-rZJcYZI3g/s1600/servlet-life-cycle-in-java.jpg)

**Question 12: What is the difference between HttpServlet and GenericServlet in Servlet API?** ([answer](http://java67.blogspot.com/2012/12/difference-between-genericservlet-vs-httpservlet-jsp.html))  
GenericServlet provides framework to create a Servlet for any protocol e.g. you can write Servlet to receive content from FTP, SMTP etc, while HttpServlet is built-in Servlet provided by Java for handling HTTP requests. See detailed answer for deep discussion.

These **Servlet interview questions** are good for the quick recap of important concept before appearing on any J2EE interview. Please share if you have come across any other interesting interview question on Servlets.

Read more: <http://javarevisited.blogspot.com/2011/09/servlet-interview-questions-answers.html#ixzz4esCiQx00>

## JSP Interview Questions and Answers

Here is collection of 10 most frequently asked *interview questions on JSP* , there were the lot on my kitty but I only included 10 to avoid getting post lengthy, also, my purpose was to prepare a list of questions which can be easily referenced before going for any JSP or Java interview. Not wasting any more time here are my top 10 JSP interview questions answers:

### Interview Questions Answers on JSP

**Question 1: Explain include Directive and include Action of JSP**

Ans:  This is a very *popular interview question on JSP*, which has been asked from a long time and still asked in the various interview. This question is good to test some fundamental concept like translation of JSP and difference between translation time and run time kind of concept.

Syntax for include Directive is **<%@ include file="fileName" %>** which means we are including some file to our JSP Page when we use include directive contents of included file will be added to calling JSP page at translation time means when the calling JSP is converted to servlet ,all the contents are added to that page .one important thing is that any JSP page is complied if we make any changes to that particular page but if we have changed the included file or JSP page the main calling JSP page will not execute again so the output will not be according to our expectation, this one is the main disadvantage of using the include directive that why it is mostly use to add static  resources, like Header and footer .

Syntax for include action is **<jsp:include page=”relativeURL” />** it’s a runtime procedure means the result of the JSP page which is mentioned in relative URL is appended  to calling JSP at runtime on their response object at the location where we have used this tag

So any changes made to included page is being effected every time, this is the main advantage of this action but only relative URL we can use here ,because request and response object is passed between calling JSP and included JSP.

**Question** **2: Difference Between** **include Directive and include Action of JSP**

This JSP interview question is a continuation of the earlier question I just made it a separate one to write an answer in clear tabular format.

|  |  |
| --- | --- |
| **Include Directive** | **Include Action** |
| include directive is processed at the translation time | Include action is processed at the run time. |
| include directive can use relative or absolute path | Include action always use relative path |
| Include directive can only include contents of resource it will not process the dynamic resource | Include action process the dynamic resource and result will be added to calling JSP |
| We can not pass any other parameter | Here we can pass other parameter also using JSP:param |
| We cannot  pass any request or response object to calling jsp to included file or JSP or vice versa | In this case it’s possible. |

**Question 3: Is it possible for one JSP to extend another java class if yes how?**

**Ans:** Yes it is possible we can extends another JSP using this <%@ include page extends="classname" %> it’s a perfectly correct because when JSP is converted to servlet its implements **javax.servlet.jsp.HttpJspPage** interface, so for jsp page its possible to extend another java class . This question can be tricky if you don’t know some basic fact , though it's not advisable to write java code in jsp instead it's better to use expression language and tag library.

**Question 4: What is < jsp:usebean >tag why it is used.**

[](http://javarevisited.blogspot.com/2011/08/convert-string-to-integer-to-string.html)

Ans: This was very popular JSP interview question during early 2000, it has lost some of its shine but still asked now and then on interviews.

**JSP Syntax**

<jsp:useBean   
        id="beanInstName"   
        scope="page | request | session | application"   
          
            class="package.class"    type="package.class"   
  
           </jsp:useBean>

This tag is used to create an instance of java bean, first of all, it tries to find out the bean if bean instance already exists assign stores a reference to it in the variable. If we specified the type, gives the Bean that type.otherwise instantiates it from the class we specify, storing a reference to it in the new variable.so jsp:usebean is a simple way to create a java bean.

**Example:**

<jsp:useBean id="stock" scope="request" class="market.Stock" />

<jsp:setProperty name="bid" property="price" value="0.0" />

a <jsp:useBean> element contains a <jsp:setProperty> element that sets property values in the Bean,we have <jsp:getProperty>element also to get the value from the bean.

**Explanation of Attribute**

 id="beanInstanceName"

A variable that identifies the Bean in the scope we specify. If the Bean has already been created by another <jsp:useBean> element, the value of id must match the value of id used in the original <jsp:useBean> element.

scope="**page** | request | session | application"

The scope in which the Bean exists and the variable named in id is available. The default value is page. The meanings of the different scopes are shown below:

* page – we can use the Bean within the JSP page with the <jsp:useBean> element
* request – we can use the Bean from any JSP page processing the same request, until a JSP page sends a response to the client or forwards the request to another file.
* session – we can use the Bean from any JSP page in the same session as the JSP page that created the Bean. The Bean exists across the entire session, and any page that participates in the session can use it..
* application – we can use the Bean from any JSP page in the same application as the JSP page that created the Bean. The Bean exists across an entire JSP application, and any page in the application can use the Bean.

class="*package.class*"

Instantiates a Bean from a class, using the new keyword and the class constructor. The class must not be abstract and must have a public, no-argument constructor.

type="*package.class*"

If the Bean already exists in the scope, gives the Bean a data type other than the class from which it was instantiated. If you use type without class or beanName, no Bean is instantiated.

**Question 5: How can one Jsp Communicate with Java file.**

Ans:we have import tag <%@ page import="market.stock.\*” %> like this we can import all the java file to our jsp and use them as a regular class another way is  servlet can send  the instance of the java class to our  jsp and we can retrieve that object from the request obj and use it in our page.

**Question 6: what are the implicit Object**

Ans: This is a fact based *interview question* what it checks is how much coding you do in JSP if you are doing it frequently you definitely know them. Implicit object is the object that is created by web container provides to a developer to access them in their program using JavaBeans and Servlets. These objects are called implicit objects because they are automatically instantiated.they are bydefault available in JSP page.

They are request, response, pageContext, session, and application, out, config, page, and exception.

**Question 7: In JSP page how can we handle runtime exception?**

Ans: This is another popular JSP interview question which has asked to check how candidate used to handle Error and Exception in JSP. We can use the errorPage attribute of the page directive to have uncaught run-time exceptions automatically forwarded to an error processing page.

Example: <%@ page errorPage="error.jsp" %>

It will redirect the browser to the JSP page error.jsp if an uncaught exception is encountered during request processing. Within error.jsp, will have to indicate that it is an error-processing page, using the directive: <%@ page isErrorPage="true" %>

**Question 8: Why is \_jspService() method starting with an '\_' while other life cycle methods do not?**

Ans: main JSP life cycle method are jspInit() jspDestroy() and \_jspService() ,bydefault whatever content we write in our jsp page will go inside the \_jspService() method by the container if again will try to override this method JSP compiler will give error but we can override other two life cycle method as we have implementing this two in jsp so making this difference container use \_ in jspService() method and shows that we cant override this method.

**Question 9: How can you pass information form one jsp to included jsp:**

Ans: This *JSP interview question* is little tricky and fact based. Using < Jsp: param> tag we can pass parameter from main jsp to included jsp page

Example:

<jsp:include page="newbid.jsp" flush="true">  
<jsp:param name="price" value="123.7"/>  
<jsp:param name="quantity" value="4"/>

**Question 10: what is the need of tag library?**

Ans tag library is a collection of custom tags. Custom actions helps recurring tasks will be handled more easily they can be reused across more than one application and increase productivity. JSP tag libraries are used by Web application designers who can focus on presentation issues rather than being concerned with how to access databases and other enterprise services. Some of the popular tag libraries are Apache display tag library and String tag library. You can also check my post on display [tag library example on Spring](http://javarevisited.blogspot.com/2011/09/displaytag-examples-tutorial-jsp-struts.html).

Read more: <http://javarevisited.blogspot.com/2011/10/jsp-interview-questions-answers-for.html#ixzz4esCwXkWO>

## 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://www.amazon.com/Java-Programming-Interviews-Exposed-Markham/dp/1118722868?tag=javamysqlanta-20) for more of such popular questions,

**Does Java support multiple inheritances?**  
This is the trickiest question in Java if C++ can support direct multiple inheritances than why not Java is the argument Interviewer often give. Answer of this question is much more subtle then it looks like, because 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)

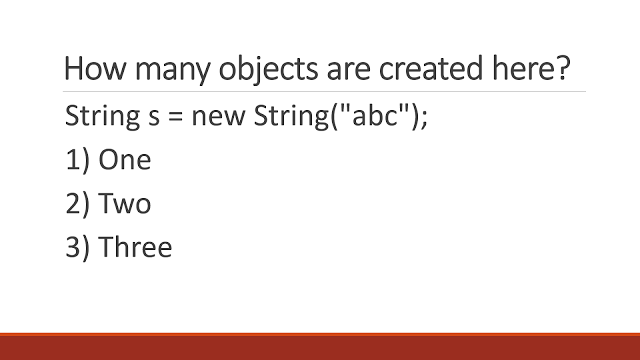
System.out.println("x=" + x);

**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+many+object+is+created+here+String+tricky+question+java.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.

Read more: <http://www.java67.com/2012/09/top-10-tricky-java-interview-questions-answers.html#ixzz4esDVwfEM>