**Common JSP interview questions**

By [*admin*](http://www.techinterviews.com/author/admin/) | September 23, 2004

1. **What are the implicit objects?** - Implicit objects are objects that are created by the web container and contain information related to a particular request, page, or application. They are: request, response, pageContext, session, application, out, config, page, exception.
2. **Is JSP technology extensible?** - Yes. JSP technology is extensible through the development of custom actions, or tags, which are encapsulated in tag libraries.
3. **How can I implement a thread-safe JSP page? What are the advantages and Disadvantages of using it?** - You can make your JSPs thread-safe by having them implement the SingleThreadModel interface. This is done by adding the directive <%@ page isThreadSafe="false" %> within your JSP page. With this, instead of a single instance of the servlet generated for your JSP page loaded in memory, you will have N instances of the servlet loaded and initialized, with the service method of each instance effectively synchronized. You can typically control the number of instances (N) that are instantiated for all servlets implementing SingleThreadModel through the admin screen for your JSP engine. More importantly, avoid using the tag for variables. If you do use this tag, then you should set isThreadSafe to true, as mentioned above. Otherwise, all requests to that page will access those variables, causing a nasty race condition. SingleThreadModel is not recommended for normal use. There are many pitfalls, including the example above of not being able to use <%! %>. You should try really hard to make them thread-safe the old fashioned way: by making them thread-safe
4. **How does JSP handle run-time exceptions?** - You can use the errorPage attribute of the page directive to have uncaught run-time exceptions automatically forwarded to an error processing page. For example: <%@ page errorPage="error.jsp" %>  
   redirects the browser to the JSP page error.jsp if an uncaught exception is encountered during request processing. Within error.jsp, if you indicate that it is an error-processing page, via the directive: <%@ page isErrorPage="true" %> Throwable object describing the exception may be accessed within the error page via the exception implicit object. Note: You must always use a relative URL as the value for the errorPage attribute.
5. **How do I prevent the output of my JSP or Servlet pages from being cached by the browser?** - You will need to set the appropriate HTTP header attributes to prevent the dynamic content output by the JSP page from being cached by the browser. Just execute the following scriptlet at the beginning of your JSP pages to prevent them from being cached at the browser. You need both the statements to take care of some of the older browser versions.

<%  
response.setHeader("Cache-Control","no-store"); //HTTP 1.1  
response.setHeader("Pragma","no-cache"); //HTTP 1.0  
response.setDateHeader ("Expires", 0); //prevents caching at the proxy server  
%>

1. **How do I use comments within a JSP page?** - You can use “JSP-style” comments to selectively block out code while debugging or simply to comment your scriptlets. JSP comments are not visible at the client. For example:
2. <%-- the scriptlet is now commented out
3. <%
4. out.println("Hello World");
5. %>
6. --%>

You can also use HTML-style comments anywhere within your JSP page. These comments are visible at the client. For example:

<!-- (c) 2004 -->

Of course, you can also use comments supported by your JSP scripting language within your scriptlets. For example, assuming Java is the scripting language, you can have:

<%

//some comment

/\*\*

yet another comment

\*\*/

%>

1. **Response has already been commited error. What does it mean?** - This error show only when you try to redirect a page after you already have written something in your page. This happens because HTTP specification force the header to be set up before the lay out of the page can be shown (to make sure of how it should be displayed, content-type=”text/html” or “text/xml” or “plain-text” or “image/jpg”, etc.) When you try to send a redirect status (Number is line\_status\_402), your HTTP server cannot send it right now if it hasn’t finished to set up the header. If not starter to set up the header, there are no problems, but if it ’s already begin to set up the header, then your HTTP server expects these headers to be finished setting up and it cannot be the case if the stream of the page is not over… In this last case it’s like you have a file started with <HTML Tag><Some Headers><Body>some output (like testing your variables.) Before you indicate that the file is over (and before the size of the page can be setted up in the header), you try to send a redirect status. It s simply impossible due to the specification of HTTP 1.0 and 1.1
2. **How do I use a scriptlet to initialize a newly instantiated bean?** - A jsp:useBean action may optionally have a body. If the body is specified, its contents will be automatically invoked when the specified bean is instantiated. Typically, the body will contain scriptlets or jsp:setProperty tags to initialize the newly instantiated bean, although you are not restricted to using those alone.  
   The following example shows the “today” property of the Foo bean initialized to the current date when it is instantiated. Note that here, we make use of a JSP expression within the jsp:setProperty action.
3. <jsp:useBean id="foo" class="com.Bar.Foo" >
4. <jsp:setProperty name="foo" property="today"
5. value="<%=java.text.DateFormat.getDateInstance().format(new java.util.Date()) %>"/ >
6. <%-- scriptlets calling bean setter methods go here --%>
7. </jsp:useBean >
8. **How can I enable session tracking for JSP pages if the browser has disabled cookies?** - We know that session tracking uses cookies by default to associate a session identifier with a unique user. If the browser does not support cookies, or if cookies are disabled, you can still enable session tracking using URL rewriting. URL rewriting essentially includes the session ID within the link itself as a name/value pair. However, for this to be effective, you need to append the session ID for each and every link that is part of your servlet response. Adding the session ID to a link is greatly simplified by means of of a couple of methods: response.encodeURL() associates a session ID with a given URL, and if you are using redirection, response.encodeRedirectURL() can be used by giving the redirected URL as input. Both encodeURL() and encodeRedirectedURL() first determine whether cookies are supported by the browser; if so, the input URL is returned unchanged since the session ID will be persisted as a cookie. Consider the following example, in which two JSP files, say hello1.jsp and hello2.jsp, interact with each other. Basically, we create a new session within hello1.jsp and place an object within this session. The user can then traverse to hello2.jsp by clicking on the link present within the page.Within hello2.jsp, we simply extract the object that was earlier placed in the session and display its contents. Notice that we invoke the encodeURL() within hello1.jsp on the link used to invoke hello2.jsp; if cookies are disabled, the session ID is automatically appended to the URL, allowing hello2.jsp to still retrieve the session object. Try this example first with cookies enabled. Then disable cookie support, restart the brower, and try again. Each time you should see the maintenance of the session across pages. Do note that to get this example to work with cookies disabled at the browser, your JSP engine has to support URL rewriting.
9. hello1.jsp
10. <%@ page session="true" %>
11. <%
12. Integer num = new Integer(100);
13. session.putValue("num",num);
14. String url =response.encodeURL("hello2.jsp");
15. %>
16. <a href='<%=url%>'>hello2.jsp</a>
17. hello2.jsp
18. <%@ page session="true" %>
19. <%
20. Integer i= (Integer )session.getValue("num");
21. out.println("Num value in session is "+i.intValue());
22. **How can I declare methods within my JSP page?** - You can declare methods for use within your JSP page as declarations. The methods can then be invoked within any other methods you declare, or within JSP scriptlets and expressions. Do note that you do not have direct access to any of the JSP implicit objects like request, response, session and so forth from within JSP methods. However, you should be able to pass any of the implicit JSP variables as parameters to the methods you declare. For example:
23. <%!
24. public String whereFrom(HttpServletRequest req) {
25. HttpSession ses = req.getSession();
26. ...
27. return req.getRemoteHost();
28. }
29. %>
30. <%
31. out.print("Hi there, I see that you are coming in from ");
32. %>
33. <%= whereFrom(request) %>
34. Another Example
35. file1.jsp:
36. <%@page contentType="text/html"%>
37. <%!
38. public void test(JspWriter writer) throws IOException{
39. writer.println("Hello!");
40. }
41. %>
42. file2.jsp
43. <%@include file="file1.jsp"%>
44. <html>
45. <body>
46. <%test(out);% >
47. </body>
48. </html>
49. Is there a way I can set the inactivity lease period on a per-session basis? - Typically, a default inactivity lease period for all sessions is set within your JSP engine admin screen or associated properties file. However, if your JSP engine supports the Servlet 2.1 API, you can manage the inactivity lease period on a per-session basis. This is done by invoking the HttpSession.setMaxInactiveInterval() method, right after the session has been created. For example:
50. <%
51. session.setMaxInactiveInterval(300);
52. %>

would reset the inactivity period for this session to 5 minutes. The inactivity interval is set in seconds.

1. **How can I set a cookie and delete a cookie from within a JSP page?** - A cookie, mycookie, can be deleted using the following scriptlet:
2. <%
3. //creating a cookie
4. Cookie mycookie = new Cookie("aName","aValue");
5. response.addCookie(mycookie);
6. //delete a cookie
7. Cookie killMyCookie = new Cookie("mycookie", null);
8. killMyCookie.setMaxAge(0);
9. killMyCookie.setPath("/");
10. response.addCookie(killMyCookie);
11. %>
12. **How does a servlet communicate with a JSP page?** - The following code snippet shows how a servlet instantiates a bean and initializes it with FORM data posted by a browser. The bean is then placed into the request, and the call is then forwarded to the JSP page, Bean1.jsp, by means of a request dispatcher for downstream processing.
13. public void doPost (HttpServletRequest request, HttpServletResponse response) {
14. try {
15. govi.FormBean f = new govi.FormBean();
16. String id = request.getParameter("id");
17. f.setName(request.getParameter("name"));
18. f.setAddr(request.getParameter("addr"));
19. f.setAge(request.getParameter("age"));
20. //use the id to compute
21. //additional bean properties like info
22. //maybe perform a db query, etc.
23. // . . .
24. f.setPersonalizationInfo(info);
25. request.setAttribute("fBean",f);
26. getServletConfig().getServletContext().getRequestDispatcher
27. ("/jsp/Bean1.jsp").forward(request, response);
28. } catch (Exception ex) {
29. . . .
30. }
31. }

The JSP page Bean1.jsp can then process fBean, after first extracting it from the default request scope via the useBean action.

jsp:useBean id="fBean" class="govi.FormBean" scope="request"

/ jsp:getProperty name="fBean" property="name"

/ jsp:getProperty name="fBean" property="addr"

/ jsp:getProperty name="fBean" property="age"

/ jsp:getProperty name="fBean" property="personalizationInfo" /

1. **How do I have the JSP-generated servlet subclass my own custom servlet class, instead of the default?** - One should be very careful when having JSP pages extend custom servlet classes as opposed to the default one generated by the JSP engine. In doing so, you may lose out on any advanced optimization that may be provided by the JSP engine. In any case, your new superclass has to fulfill the contract with the JSP engine by:  
   Implementing the HttpJspPage interface, if the protocol used is HTTP, or implementing JspPage otherwise Ensuring that all the methods in the Servlet interface are declared final Additionally, your servlet superclass also needs to do the following:
   * The service() method has to invoke the \_jspService() method
   * The init() method has to invoke the jspInit() method
   * The destroy() method has to invoke jspDestroy()

If any of the above conditions are not satisfied, the JSP engine may throw a translation error.  
Once the superclass has been developed, you can have your JSP extend it as follows:

<%@ page extends="packageName.ServletName" %>

1. **How can I prevent the word "null" from appearing in my HTML input text fields when I populate them with a resultset that has null values?** - You could make a simple wrapper function, like
2. <%!
3. String blanknull(String s) {
4. return (s == null) ? "" : s;
5. }
6. %>
7. then use it inside your JSP form, like
8. <input type="text" name="shoesize" value="<%=blanknull(shoesize)% >" >
9. **How can I get to print the stacktrace for an exception occuring within my JSP page?** - By printing out the exception’s stack trace, you can usually diagonse a problem better when debugging JSP pages. By looking at a stack trace, a programmer should be able to discern which method threw the exception and which method called that method. However, you cannot print the stacktrace using the JSP out implicit variable, which is of type JspWriter. You will have to use a PrintWriter object instead. The following snippet demonstrates how you can print a stacktrace from within a JSP error page:
10. <%@ page isErrorPage="true" %>
11. <%
12. out.println(" ");
13. PrintWriter pw = response.getWriter();
14. exception.printStackTrace(pw);
15. out.println(" ");
16. %>
17. **How do you pass an InitParameter to a JSP?** - The JspPage interface defines the jspInit() and jspDestroy() method which the page writer can use in their pages and are invoked in much the same manner as the init() and destory() methods of a servlet. The example page below enumerates through all the parameters and prints them to the console.
18. <%@ page import="java.util.\*" %>
19. <%!
20. ServletConfig cfg =null;
21. public void jspInit(){
22. ServletConfig cfg=getServletConfig();
23. for (Enumeration e=cfg.getInitParameterNames(); e.hasMoreElements();) {
24. String name=(String)e.nextElement();
25. String value = cfg.getInitParameter(name);
26. System.out.println(name+"="+value);
27. }
28. }
29. %>
30. **How can my JSP page communicate with an EJB Session Bean?** - The following is a code snippet that demonstrates how a JSP page can interact with an EJB session bean:
31. <%@ page import="javax.naming.\*, javax.rmi.PortableRemoteObject, foo.AccountHome, foo.Account" %>
32. <%!
33. //declare a "global" reference to an instance of the home interface of the session bean
34. AccountHome accHome=null;
35. public void jspInit() {
36. //obtain an instance of the home interface
37. InitialContext cntxt = new InitialContext( );
38. Object ref= cntxt.lookup("java:comp/env/ejb/AccountEJB");
39. accHome = (AccountHome)PortableRemoteObject.narrow(ref,AccountHome.class);
40. }
41. %>
42. <%
43. //instantiate the session bean
44. Account acct = accHome.create();
45. //invoke the remote methods
46. acct.doWhatever(...);
47. // etc etc...

%>

# [How to generate MD5 Hash in Java - String Byte Array digest Example](http://javarevisited.blogspot.in/2013/03/generate-md5-hash-in-java-string-byte-array-example-tutorial.html)

There are multiple ways to generate MD5 hash in Java program. Not only Java API provides convenient method to generate MD5 hash, you can also use popular open source frameworks, like Spring and Apache commons Codec to generate MD5 digest in Java. MD5 is popular Message Digest Algorithm, which is most commonly used to check data integrity e.g. comparing MD5 check-sum to see, if any file is altered or not. Though, MD5 was not considered a good cryptographic algorithm for security purpose due to several vulnerability found on it, it's still good enough or checking integrity of file. MD5 hashing algorithm generate a 128 bit or 16 byte long hash value. *MD5 hash values*, also known as *MD5 digest* is mostly represented as 32 character Hex String. You can generate MD5 hash from a [byte array](http://javarevisited.blogspot.sg/2012/12/how-to-compare-arrays-in-java-equals-deepequals-primitive-object.html), or [String](http://javarevisited.blogspot.com/2012/08/convert-inputstream-to-string-java-example-tutorial.html) directly using Java, Spring and Apache commons codec. Spring and Apache commons codec has identical API e.g. class name DigestUtils is same and allows you to directly generate MD5 hash as Hex String, while if you use Java than you need to convert byte array to Hex String, as java.security.MessageDigest.digest() method returns MD5 hash as byte array. Earlier we have seen, [How to encode and decode String in base64 encoding](http://javarevisited.blogspot.com/2012/02/how-to-encode-decode-string-in-java.html), and In this Java tutorial we will see,*How to generate MD5 hash or digest using Java*, Spring and Apache commons code library.

## How to Generate MD5 hash in Java - example

[How to create MD5 hash digest from String and byte array in Java](http://3.bp.blogspot.com/-K6q0DQ1v-tw/TWu8owBtc2I/AAAAAAAAADA/oBoHDBiJ8ag/s1600/17.jpg)In this part, we will see some example to generate MD5 hash in Java. Following Java program generates MD5 hash or digest of a String, by converting into byte array. Java Security package provides MessageDigest, which can generate MD5 hash. MessageDigest's digest() method accept a byte array, and return a byte array of hash value. Since many times we need MD5 hash as Hex String, I have converted that byte array into [Hex String](http://javarevisited.blogspot.sg/2011/10/convert-decimal-binary-octal-java.html). Apart from core Java example, which doesn't use any dependency, there are couple of more open source solution of generating MD5 digest. Two of the most popular open source library, Apache Commons Codec and Spring Framework provides utility method to create MD5 Hash in both byte array and Hex String format. Both Apache commons code and Spring provides, DigestUtils class with overloaded method md5() and md5Hex(), which can accept either [String](http://javarevisited.blogspot.sg/2012/10/10-java-string-interview-question-answers-top.html) or byte array and can return either 16 element byte array or 32 characters Hex String. If you are already using either Spring or Apache Commons Codec, than its best to use DigestUtil, as it only need a line of code to generate MD5 hash in Java. here is complete code example of generating MD5 digest in Java.

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.commons.codec.digest.DigestUtils;

*/\*\**

*\* Java program to generate MD5 hash or digest for String. In this example*

*\* we will see 3 ways to create MD5 hash or digest using standard Java API,*

*\* Spring framework and open source library, Apache commons codec utilities.*

*\* Generally MD5 has are represented as Hex String so each of this function*

*\* will return MD5 hash in hex format.*

*\**

*\* @author Javin Paul*

*\*/*

public class MD5Hash {

public static void main(String args[]) {

String password = "password";

System.out.println("MD5 hash generated using Java : " + md5Java(password));

System.out.println("MD5 digest generated using : " + md5Spring(password));

System.out.println("MD5 message created by Apache commons codec : " + md5ApacheCommonsCodec(password));

}

public static String md5Java(String message){

String digest = null;

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] hash = md.digest(message.getBytes("UTF-8"));

*//converting byte array to Hexadecimal String*

StringBuilder sb = new StringBuilder(2\*hash.length);

for(byte b : hash){

sb.append(String.format("%02x", b&0xff));

}

digest = sb.toString();

} catch (UnsupportedEncodingException ex) {

Logger.getLogger(StringReplace.class.getName()).log(Level.SEVERE, null, ex);

} catch (NoSuchAlgorithmException ex) {

Logger.getLogger(StringReplace.class.getName()).log(Level.SEVERE, null, ex);

}

return digest;

}

*/\**

*\* Spring framework also provides overloaded md5 methods. You can pass input*

*\* as String or byte array and Spring can return hash or digest either as byte*

*\* array or Hex String. Here we are passing String as input and getting*

*\* MD5 hash as hex String.*

*\*/*

public static String md5Spring(String text){

return DigestUtils.md5Hex(text);

}

*/\**

*\* Apache commons code provides many overloaded methods to generate md5 hash. It contains*

*\* md5 method which can accept String, byte[] or InputStream and can return hash as 16 element byte*

*\* array or 32 character hex String.*

*\*/*

public static String md5ApacheCommonsCodec(String content){

return DigestUtils.md5Hex(content);

}

}

Output:

MD5 hash generated using Java : 5f4dcc3b5aa765d61d8327deb882cf99

MD5 digest generated using : 5f4dcc3b5aa765d61d8327deb882cf99

MD5 message created by Apache commons code : 5f4dcc3b5aa765d61d8327deb882cf99

That's all on **How to generate MD5 hash or digest in Java**. As you have seen, there is more than one way to generate MD5 digest, both in byte array as well as Hex String format. Just remember to specify file encoding to String.getBytes() method, until you have not using any application wide [character encoding](http://javarevisited.blogspot.com/2012/01/get-set-default-character-encoding.html). Since String.getBytes() uses platform specific encoding which could be different in your windows development machine and production Linux Server. Also, consider using Spring or Apache Commons Code to generate MD5 Hash value, if you are already using these libraries. It's always best to reuse library code than writing your own MD5 hash function, to avoid testing overhead.

Read more: [http://javarevisited.blogspot.com/2013/03/generate-md5-hash-in-java-string-byte-array-example-tutorial.html#ixzz2OL07L9rL](http://javarevisited.blogspot.com/2013/03/generate-md5-hash-in-java-string-byte-array-example-tutorial.html" \l "ixzz2OL07L9rL)

# [3 Example to print array values in Java - toString and deepToString from Arrays](http://javarevisited.blogspot.in/2012/12/3-example-to-print-array-values-in-java.html)

Printing array values in Java or values of array element in Java would have been much easier if  arrays are allowed to directly prints its values whenever used inside System.out.println() or [format and printf method](http://javarevisited.blogspot.sg/2012/08/how-to-format-string-in-java-printf.html), Similar to various classes in Java do this by [overriding toString() method](http://javarevisited.blogspot.sg/2012/09/override-tostring-method-java-tips-example-code.html). Despite being an object, array in Java doesn't print any meaningful representation of its content when passed toSystem.out.println() or any other print methods. If you are using array in method argument or any other prominent place in code and actually interested in values of array then you don't have much choice than for loop until Java 1.4. Things has been changed since Java 5 because it introduced two extremely convenient methods for printing values of both primitive and [object arrays in Java](http://javarevisited.blogspot.sg/2012/12/what-is-object-in-java-or-oops-example.html). Arrays.toString(array) andArrays.deepToString(twoDimensionArray) can print values of any array. Main difference between Arrays.toString() and Arrays.deepToString  is that deepToString is used to print values of multidimensional array which is far more convenient than nesting of multiple for loops. In this Java tutorial we will see **3 different ways of printing array values in Java** or value of element from Array in Java.

## 3 ways to print array values in Java

As I said there is no direct way to print values of array in Java if you directly pass primitive or object array to System.out.println() you will receive following output:

**System**.out.println("Print array values in Java 1.4 :" + **Arrays**.asList(sArray));  
**System**.out.println("Print array values in Java 1.4 :" + **Arrays**.asList(iArray));  
  
Output:  
Printing **Integer** array in Java: [I@15b7986  
Printing **String** array in Java: [Ljava.lang.**String**;@87816d

You can't decipher anything until you are quite familiar of this array format and even then it doesn't tell anything about contents of array. It just print type of element and [hashcode](http://javarevisited.blogspot.sg/2011/02/how-to-write-equals-method-in-java.html). In order to print values of array you can use any of following 3 examples:

1) Use enhanced for loop or classic for loop with lenth of array.

2) Use Arrays.asList() to convert Array into [ArrayList](http://javarevisited.blogspot.sg/2011/05/example-of-arraylist-in-java-tutorial.html) and than print

3) Use Java 5 Arrays.toString() and Arrays.deepToString() methods

## Print Array Value Example 1: Using for loop

[How to print values of Array or values of elements from Array in Java](http://3.bp.blogspot.com/-K6q0DQ1v-tw/TWu8owBtc2I/AAAAAAAAADA/oBoHDBiJ8ag/s1600/17.jpg)[for loop](http://javarevisited.blogspot.sg/2012/05/break-continue-and-lablel-in-loop-java.html) is the classical way of printing or displaying values of both one dimension and multidimensional arrays in Java. before Java 5 you can use array.length to iterate over all array elements and printing values for each of them. From Java 5 onwards you can use much cleaner **enhanced for loop** which doesn't require any counter from moving one element to other in Java. Enhanced for loop in Java 5 is added with other popular language feature e.g. [Enum](http://javarevisited.blogspot.sg/2011/08/enum-in-java-example-tutorial.html),[Autoboxing](http://javarevisited.blogspot.sg/2012/07/auto-boxing-and-unboxing-in-java-be.html) and [Generics](http://javarevisited.blogspot.sg/2012/08/how-to-write-parametrized-class-method-Generic-example.html). Here is sample code example to print value of element from array using classical and enhanced for loop in Java:

**//** **Classic for loop before Java 5**

**private** **static** **int**[] iArray = **new** **int**[]{1, 2,3,4, 5};  
  
for(**int** i=0; i< iArray.length; i++){  
   **System**.out.print(iArray[i] +", ");  
}  
  
Output:  
1, 2, 3, 4, 5,  
  
//Enhanced for loop from Java 1.5  
for(**int** i : iArray){  
   **System**.out.print(i +", ");  
}

As you see using enhanced for loop for printing array values is more concise and clean.

## Print Array Values Example 2: Using Arrays.asList

Arrays.asList() method is used to [convert Array into ArrayList](http://javarevisited.blogspot.sg/2011/06/converting-array-to-arraylist-in-java.html) and as you know Collection classes overrides [toString method](http://javarevisited.blogspot.sg/2012/09/override-tostring-method-java-tips-example-code.html) to print there contents. By converting array into List we can leverage that property and print values from ArrayList instead of Array. Only limitation of this approach is it doesn't print contents of array if array is of primitive type like int, float or double but works well if Array contains objects like String. Arrays.asList() is also used to [create and initialize List in one line](http://javarevisited.blogspot.sg/2012/12/how-to-initialize-list-with-array-in-java.html). By the way here is simple code example of *displaying values form array* *in Java* using Arrays.asList() method:

**System**.out.println("Print String array values in Java 1.4 :" + **Arrays**.asList(sArray));  
**System**.out.println("Print int array values in Java 1.4 :" + **Arrays**.asList(iArray));  
  
Output:  
Print **String** array values in Java 1.4 :[abc, bcd, def, efg]  
Print **int** array values in Java 1.4 :[[I@15b7986]

## Print Array Value Example 3: using Arrays.toString and Arrays.deepToString

This is by far best and recommended way of printing values from Array in Java. Only caveat is that Arrays.toString() and Arrays.deepToString() are added from Java 5 onwards along with other features e.g. [Generics](http://javarevisited.blogspot.sg/2011/09/generics-java-example-tutorial.html), [varargs](http://javarevisited.blogspot.sg/2011/09/variable-argument-in-java5-varargs.html) or static [import](http://javarevisited.blogspot.sg/2012/10/what-is-static-import-in-java-5-example-tutorial.html). Use Arrays.toString() method to print both primitive and object single or one dimension array and use Arrays.deepToString() method to print values from two dimensional or multidimensional array (array of array in Java). Here is a simple example of printing array values using Arrays.toString() and Arrays.deepToString() in Java:

**System**.out.println("Print values of Integer array in Java: " + **Arrays**.toString(iArray));  
**System**.out.println("Print values of String array in Java: " + **Arrays**.toString(sArray));  
        
**int**[][] twoDimensionArray = **new** **int**[][]{  
                                    {1,2,3},  
                                    {10,20,30},  
                                    {100,200,300},  
                                    };  
**System**.out.println("Print two dimensional array in Java: " + **Arrays**.deepToString(twoDimensionArray));  
  
Output:  
Print values of **Integer** array in Java: [1, 2, 3, 4, 5]  
Print values of **String** array in Java: [abc, bcd, def, efg]  
Print two dimensional array in Java: [[1, 2, 3], [10, 20, 30], [100, 200, 300]]

## Java program to print array values

Here is the combined examples of printing value of elements from Array in Java using above three examples. Best way to print elements of Array is to use new methods added in java.util.Arrays class in Java 5 e.g. toString() and deepToString().

**import** java.util.Arrays;  
  
**public** **class** PrintArrayExample {  
  
    **private** **static** **int**[] intArray = **new** **int**[]{1, 2,3,4, 5};  
    **private** **static** **String**[] strArray = **new** **String**[]{"abc", "bcd", "def", "efg"};  
      
    **public** **static** **void** main(**String** args[]) {  
        **System**.out.println("Java Example to print int array in Java: " + intArray);  
        **System**.out.println("Java Example to print string array in Java: " + strArray);  
        
        *//generic way of printing values of array before java 5*  
        for(**int** i=0; i< intArray.length; i++){  
            **System**.out.print(intArray[i] +", ");  
        }  
        
        *//printing array values using enhanced for loop java 1.5*  
        for(**int** i : intArray){  
            **System**.out.print(i +", ");  
        }   
        
        *//another way to print array values in Java 1.4 is using Arrays.asList*  
        **System**.out.println("Java Example to print String array values in Java 1.4 :"

                            + **Arrays**.asList(strArray));  
        **System**.out.println("Java Example to int array values in Java 1.4 :"

                            + **Arrays**.asList(intArray));  
        
        *//better way of printing values of array in java 1.5*  
        **System**.out.println("Java Example to print values of array in Java: "

                            + **Arrays**.toString(intArray));  
        **System**.out.println("Java Example to print values of array in Java: "

                            + **Arrays**.toString(strArray));  
        
        **int**[][] twoDimensionArray = **new** **int**[][]{  
                                    {1,2,3},  
                                    {10,20,30},  
                                    {100,200,300},  
                                    };  
        **System**.out.println("Java Example to print two dimensional array in Java: "

                            + **Arrays**.deepToString(twoDimensionArray));  
  
    }  
}  
  
**Output:**  
Java Example to print **int** array in Java: [I@1820dda  
Java Example to print string array in Java: [Ljava.lang.**String**;@15b7986  
1, 2, 3, 4, 5, 1, 2, 3, 4, 5, Java Example to print **String** array values in Java 1.4 :[abc, bcd, def, efg]  
Java Example to **int** array values in Java 1.4 :[[I@1820dda]  
Java Example to print values of array in Java: [1, 2, 3, 4, 5]  
Java Example to print values of array in Java: [abc, bcd, def, efg]  
Java Example to print two dimensional array in Java: [[1, 2, 3], [10, 20, 30], [100, 200, 300]]

That's all on how to print array values in Java. we have seen three different example to display contents of array in Java and most easy and convenient approach is using Arrays.toString() and Arrays.deepToString(). if you are using Java 5.

Read more: [http://javarevisited.blogspot.com/2012/12/3-example-to-print-array-values-in-java.html#ixzz2OL0NNKFf](http://javarevisited.blogspot.com/2012/12/3-example-to-print-array-values-in-java.html" \l "ixzz2OL0NNKFf)

### Difference between HashMap and ConcurrentHashMap in Java Collection

**HashMap vs ConcurrentHashMap in Java**  
ConcurrentHashMap in Java is introduced as an alternative of [Hashtable in Java](http://javarevisited.blogspot.sg/2012/01/java-hashtable-example-tutorial-code.html), which is a synchronized collection class, that makes the main difference between HashMap and ConcurrentHashMap which is one is non synchronized , non [thread safe](http://javarevisited.blogspot.sg/2012/03/simpledateformat-in-java-is-not-thread.html)and not for use in Concurrent multi-threaded environment while ConcurrentHashMap is a thread-safe collection and intended to be used as primary Map implementation especially for multi-threaded and Concurrent environment. Apart from thread-safety there are some subtle differences between HashMap and ConcurrentHashMap which we will see in this article. By the way Difference between HashMap and ConcurrentHashMap as well as [ConcurrentHashMap vs Hashtable](http://javarevisited.blogspot.sg/2011/04/difference-between-concurrenthashmap.html) are two popular [core Java interview question](http://javarevisited.blogspot.sg/2011/04/top-20-core-java-interview-questions.html), mostly asked on senior level Java programmers.

### Difference between HashMap and ConcurrentHashMap in Java

[Difference between ConcurrentHashMap and HashMap in Java Collection](http://javarevisited.blogspot.sg/2011/08/enum-in-java-example-tutorial.html)In this section we will see some more details about HashMap and ConcurrentHashMap and compare them on various parameters like thread-safety, synchronization, performance, ease of use etc.  
  
1) As I said earlier first significant difference between HashMap and ConcurrentHashMap is that later is[thread-safe](http://javarevisited.blogspot.sg/2012/01/how-to-write-thread-safe-code-in-java.html) and can be used in concurrent environment without external synchronization. Though it doesn't provide same level of synchronization as achieved by using Hashtable but its enough for most practical purpose.  
  
2)You can make HashMap synchronized by wrapping it on Collections.synchornizedMap(HashMap) which will return a collection which is almost equivalent to Hashtable, where every modification operation on Map is locked on Map object while in case of ConcurrentHashMap, thread-safety is achieved by dividing whole Map into different partition based upon [Concurrency](http://javarevisited.blogspot.sg/2012/07/cyclicbarrier-example-java-5-concurrency-tutorial.html)level and only locking particular portion instead of locking whole Map.  
  
3) ConcurrentHashMap is more scalable and performs better than Synchronized HashMap in multi-threaded environment while in Single threaded environment both HashMap and ConcurrentHashMap gives comparable performance, where HashMap only slightly better.  
  
  
In Summary Main difference between ConcurrentHashMap and HashMap in [Java Collection](http://javarevisited.blogspot.sg/2011/11/collection-interview-questions-answers.html) turns out to be thread-safety, Scalability and Synchronization. ConcurrentHashMap is better choice than synchronized HashMap if you are using them as cache, which is most popular use case of a Map in Java application. ConcurrentHashMap is more scalable and outperform when number of reader threads outnumber number of writer threads.

# [Difference between transient and volatile keyword in Java](http://javarevisited.blogspot.in/2012/03/difference-between-transient-and.html)

Surprisingly "**Difference between transient and volatile keyword in Java**" has asked many times on [various java interview](http://javarevisited.blogspot.com/2011/04/top-20-core-java-interview-questions.html). volatile and transient are two completely different keywords from different areas of Java programming language. transient keyword is used during [serialization of Java object](http://javarevisited.blogspot.com/2011/04/top-10-java-serialization-interview.html) while volatile is related to visibility of variables modified by multiple thread during concurrent programming. Only similarity between volatile and transient is that they are less used or uncommon keywords and not as popular as public, [static](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html) or [final](http://javarevisited.blogspot.com/2011/12/final-variable-method-class-java.html). Anyway its good to know what transient keyword do in Java or how to use volatile keyword in Java. In this article we will couple of points between volatile and transient which can be used to answer this interview question.

[Difference between transient and volatile keyword in Java](http://2.bp.blogspot.com/-wrzDeQGAe1I/TWu8pLuLr4I/AAAAAAAAADE/V017G-6Q61w/s1600/java_logo_50_50.jpg)This article is in continuation of earlier interview question on serialization like [difference between Serializable and Externaliable](http://javarevisited.blogspot.com/2012/01/serializable-externalizable-in-java.html)  and  [Top 10 Java serialization interview question](http://javarevisited.blogspot.com/2011/04/top-10-java-serialization-interview.html). If you haven’t read them already you may find them useful and interesting.

## Difference between volatile and transient keyword in Java

1) transient keyword is used along with [instance variables](http://javarevisited.blogspot.com/2012/02/difference-between-instance-class-and.html) to exclude them from serialization process. if a field  is transient its value will not be persisted. see my post [what is transient keyword in java](http://javarevisited.blogspot.com/2011/09/transient-keyword-variable-in-java.html) for more details. On the other hand volatile keyword can also be used in variables to indicate compiler and JVM that always read its value from main memory and follow happens-before relationship on visibility of volatile variable among multiple thread. see my post how and [when to use volatile keyword in Java](http://javarevisited.blogspot.com/2011/06/volatile-keyword-java-example-tutorial.html) for more details.

2) transient keyword can not be used along with static keyword but volatile can be used along with static.

3) transient variables are initialized with default value during de-serialization and there assignment or restoration of value has to be handled by application code.

That’s all on **difference between transient and volatile keyword in java**. As I said this interview question doesn’t really test you and just try to find whether you are familiar with those less known keywords in java or not. Let us know if you come across any other difference between volatile and transient keyword in java.

Read more: <http://javarevisited.blogspot.com/2012/03/difference-between-transient-and.html#ixzz2OL2RCSRo>

# [Why multiple inheritances are not supported in Java](http://javarevisited.blogspot.in/2011/07/why-multiple-inheritances-are-not.html)

[Why multiple inheritence is not supported implemented in java](http://javarevisited.blogspot.com/2011/06/noclassdeffounderror-exception-in.html)Recently one of my friend appeared for an interview and after few so called easy questions he was asked **"Why multiple inheritance is not supported in Java"** , though he has a brief idea that in Java we can support multiple inheritance in java via interface but interviewer was keep pressing on why part , may be he was just read any blog post about it :). So after the interview my friend comes to me and in usual talk he told me about this questions and ask me the answer. Well this is very classical question like**[Why String is immutable in Java](http://javarevisited.blogspot.com/2010/10/why-string-is-immutable-in-java.html)**; similarity between these two questions is they are mainly driven by design decision taken by java's creator or designer. Though following two reason make sense to me on Why Java doesn't support multiple inheritances:

## Why Java doesn't support multiple inheritance

1) First reason is **ambiguity around Diamond problem**, consider a class A has foo() method and then B and C derived from A and has there own foo() implementation and now class D derive from B and C using multiple inheritance and if we refer just foo() compiler will not be able to decide which foo() it should invoke. This is also called Diamond problem becausestructure on this inheritance scenario is similar to 4 edge diamond, see below

           A foo()

           / \

          /   \

   foo() B     C foo()

          \   /

           \ /

            D

           foo()

In my opinion even if we remove the top head of diamond class A and allow multiple inheritances we will see this problem of ambiguity.  
  
Some times if you give this reason to interviewer he asks if C++ can support *multiple inheritance* than why not Java. hmmmmm in that case I would try to explain him the second reason which I have given below that its not because of technical difficulty but more to maintainable and clearer design was driving factor though this can only be confirmed by any of java designer and we can just speculate. [Wikipedia link](http://en.wikipedia.org/wiki/Diamond_problem)has some good explanation on how different language address problem arises due to diamond problem while using multiple inheritances.  
  
2) Second and more convincing reason to me is that **multiple inheritances does complicate the design and creates problem during casting, constructor chaining etc** and given that there are not many scenario on which you need multiple inheritance its wise decision to omit it for the sake of simplicity. Also java avoids this ambiguity by supporting single inheritance with interfaces. Since interface only have method declaration and doesn't provide any implementation there will only be just one implementation of specific method hence there would not be any ambiguity.

Read more: [http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html#ixzz2OL2cZ0aX](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html" \l "ixzz2OL2cZ0aX)

# [Why wait, notify and notifyAll is defined in Object Class and not on Thread class in Java](http://javarevisited.blogspot.in/2012/02/why-wait-notify-and-notifyall-is.html)

**Why wait, notify and notifyAll is declared in Object Class instead of Thread** is famous core java interview question which is asked during all levels of Java interview ranging from 2 years, 4years to quite senior level position on java development. Beauty of this question is that it reflect what does interviewee knows about wait notify mechanism, how does it sees whole wait and notify feature and whether his understanding is not shallow on this topic. Like [Why Multiple inheritance is not supported in Java](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html) or [why String is final in java](http://javarevisited.blogspot.com/2010/10/why-string-is-immutable-in-java.html) there could be multiple answers of *why wait and notify is defined in Object class* and every one could justify there reason.

[Why Wait notify method is declared in Object Class and not in Thread in Java](http://javarevisited.blogspot.com/2012/01/google-interview-questions-answers-top.html)In my all interview experience I found that wait and notify still remains most confusing for most of Java programmer specially up-to 2 to 3 years and if they asked to write code using wait and notify they often struggle. So if you are going for any Java interview make sure you have sound knowledge of wait and notify mechanism as well as you are comfortable writing code using wait and notify like Produce Consumer problem or implementing Blocking queue etc. by the way This article is in continuation of  my earlier article related to wait and notify e.g. [Why Wait and notify requires to be called from Synchronized block or method](http://javarevisited.blogspot.com/2011/05/wait-notify-and-notifyall-in-java.html) and  [Difference between wait, sleep and yield method in Java](http://javarevisited.blogspot.com/2011/12/difference-between-wait-sleep-yield.html) , if you haven’t read you may found interesting.

## Reason Why Wait , Notify and NotifyAll are in Object Class.

Here are some thoughts on why they should not be in Thread class which make sense to me :

1) Wait and notify is not just normal methods or synchronization utility, more than that they are **communication mechanism between two threads in Java**. And Object class is correct place to make them available for every object if this mechanism is not available via any java keyword like synchronized. Remember synchronized and wait notify are two different area and don’t confuse that they are same or related. Synchronized is to provide mutual exclusion and ensuring [thread safety of Java class](http://javarevisited.blogspot.com/2012/01/how-to-write-thread-safe-code-in-java.html) like race condition while wait and notify are communication mechanism between two thread.

2 )**Locks are made available on per Object basis**, which is another reason wait and notify is declared in Object class rather then Thread class.

3) In Java in order to enter critical section of code, Threads needs lock and they wait for lock, they don't know which threads holds lock instead they just know the lock is hold by some thread and they should wait for lock instead of knowing which thread is inside the synchronized block and asking them to release lock. this analogy fits with wait and notify being on object class rather than thread in Java.

These are just my thoughts on **why wait and notify method is declared in Object class rather than Thread in Java** and you have different version than me. In reality its another design decision made by Java designer like [not supporting Operator overloading in Java](http://javarevisited.blogspot.com/2011/08/why-java-does-not-support-operator.html). Anyway please post if you have any other convincing reason *why wait and notify method should be in Object class and not on Thread*.

Update:  
@Lipido has made an insightful comment , which is worth adding here. read his comment for full text

"Java is based on Hoare's monitors idea (http://en.wikipedia.org/wiki/Monitor\_%28synchronization%29). In Java all object has a monitor. Threads waits on monitors so, to perform a wait, we need 2 parameters:  
- a Thread  
- a monitor (any object)  
  
In the Java design, the thread can not be specified, it is always the current thread running the code. However, we can specify the monitor (which is the object we call wait on). This is a good design, because if we could make any other thread to wait on a desired monitor, this would lead to an "intrusion", posing difficulties on designing/programming concurrent programs. Remember that in Java all operations that are intrusive in another thread's execution are deprecated (e.g. stop())."

Read more: [http://javarevisited.blogspot.com/2012/02/why-wait-notify-and-notifyall-is.html#ixzz2OL2gdK4U](http://javarevisited.blogspot.com/2012/02/why-wait-notify-and-notifyall-is.html" \l "ixzz2OL2gdK4U)

[**Why character array is better than String for Storing password in Java**](http://javarevisited.blogspot.in/2012/03/why-character-array-is-better-than.html)

**Why character array is better than String for storing password in Java** was recent question asked to one of my friend in a java interview. he was interviewing for a Technical lead position and has over 6 years of experience.Both [Character array and String](http://javarevisited.blogspot.com/2012/02/how-to-convert-char-to-string-in-java.html) can be used to store text data but choosing one over other is difficult question if you haven't faced the situation already. But as my friend said any question related to Stringmust have a clue on special property of Strings like immutability and he used that to convince interviewer. here we will explore some reasons on why should you used char[] for storing password than String.  
  
[character array over string storing password java](http://javarevisited.blogspot.com/2011/08/convert-string-to-integer-to-string.html)This article is in continuation of my earlier interview question post on String e.g. [Why String is immutable in Java](http://javarevisited.blogspot.com/2010/10/why-string-is-immutable-in-java.html) or [How Substring can cause memory leak in Java](http://javarevisited.blogspot.com/2011/10/how-substring-in-java-works.html), if you haven't read those you may find them interesting.Here are few reasons which makes sense to believe that character array is better choice for storing password in Java than String:

1) Since **Strings are immutable in Java** if you store password as plain text it will be available in memory until Garbage collector clears it and since String are used in String pool for reusability there is pretty high chance that it will be remain in memory for long duration, which pose a security threat. Since any one who has access to memory dump can find the password in clear text and that's another reason you should always used an encrypted password than plain text. Since Strings are immutable there is no way contents of Strings can be changed because [any change will produce new String](http://javarevisited.blogspot.com/2011/07/string-vs-stringbuffer-vs-stringbuilder.html), while if you char[] you can still set all his element as blank or zero. So **Storing password in character array clearly mitigates security risk of stealing password**.

2) **Java itself recommends**using getPassword() method of JPasswordField which returns a char[] and deprecated getText() method which returns password in clear text stating security reason. Its good to follow advice from Java team and adhering to standard rather than going against it.

3) With String there is always a risk of printing plain text in [log file or console](http://javarevisited.blogspot.com/2011/05/top-10-tips-on-logging-in-java.html) but if use [Array](http://javarevisited.blogspot.com/2012/01/anonymous-array-example-java-create.html) you won't print contents of array instead its memory location get printed. though not a real reason but still make sense.

**[String](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html)** strPassword="Unknown";  
**char**[] charPassword= **new** **char**[]{'U','n','k','w','o','n'};  
**System**.out.println("String password: " + strPassword);  
**System**.out.println("Character password: " + charPassword);  
  
**String** password: Unknown  
**Character** password: [C@110b053

That's all on *why character array is better choice than String for storing passwords in Java*.  Though using char[] is not just enough you need to erase content to be more secure. I also suggest working with hash'd or [encrypted password](http://javarevisited.blogspot.com/2012/02/how-to-encode-decode-string-in-java.html) instead of plaintext and clearing it from memory as soon as authentication is completed.

Read more: [http://javarevisited.blogspot.com/2012/03/why-character-array-is-better-than.html#ixzz2OL2lMB8F](http://javarevisited.blogspot.com/2012/03/why-character-array-is-better-than.html" \l "ixzz2OL2lMB8F)

### Top 10 Servlet JSP Interview Questions for Java J2EE programmer - FAQ

Servlets and JSP Interview Questions are core of any J2EE Interview. Its expected from candidate to have good knowledge of Servlet, JSP, EJB , Struts or Spring in any J2EE interview. Consequently Servlets and [JSP Interview Questions](http://javarevisited.blogspot.sg/2011/10/jsp-interview-questions-answers-for.html) are very common in any J2EE Interviews along with some [questions from core Java](http://java67.blogspot.sg/2012/08/10-advanced-core-java-interview.html), Some interview [Questions from Spring](http://javarevisited.blogspot.sg/2011/09/spring-interview-questions-answers-j2ee.html) and Some [EJB interview questions](http://java67.blogspot.sg/2012/10/20-ejb-30-interview-questions-and.html). Since J2EE is main platform for building enterprise and web application using Java programming language, its important to prepare well for J2EE interview, especially If you are in application development space. Servlet and JSP forms web layer of J2EE framework where Java Server pages is main view technology and Servlets are main Controller component of standard MVC architecture. In this article we will see couple of most common or frequently asked JSP and [Servlet Interview questions and answers](http://javarevisited.blogspot.sg/2011/09/servlet-interview-questions-answers.html), which can help you on your J2EE interview preparation and understanding some key concept of Java web applications.

**JSP Servlet Interview Questions Answers**

[Servlet JSP Interview Questions Answers FAQ](http://3.bp.blogspot.com/-1lzFJzIgaHk/UF2Ci6kY5pI/AAAAAAAAAes/OYiM7r-DHzc/s1600/17.jpg)Here is my list of frequently asked JSP Servlet Interview questions to any Java programmer. Most of these Servlet JSP questions has appeared in various level of J2EE interviews including 2 to 3 and 2 to 4 years experience level. None of these questions require special question and neither of them are [tricky questions](http://java67.blogspot.sg/2012/09/top-10-tricky-java-interview-questions-answers.html), they all are focused on fundamentals which is key to clear any interview.

**Question 1: What is JSESSIONID in Java? When does JSESSIONID gets created ?**

One of my favourite Servlet JSP Interview question for 2 to 4 years experience programmers on web development. JSESSION id is a cookie which is used to manage session in Java web application. JSESSIONID is created by Web Container whenever a new session is created. See [What is JSESSIONID in Servlet JSP](http://javarevisited.blogspot.sg/2012/08/what-is-jsessionid-in-j2ee-web.html) for more details.

**Question 2: What is difference between include action and include directive in JSP?**

Another very popular JSP Interview questions, mostly asked to 2 to 3 years experienced J2EE programmer. There are couple of differences, most important of them is that include action is request  time inclusion while include directive is translation time inclusion of another resource e.g. JSP or html pages. This questions is also asked as difference between file include and page include. See [Include action vs Include directive](http://javarevisited.blogspot.sg/2012/01/difference-between-page-include-and.html) for more differences.

**Question 3: How do you define application wide error page in JSP?**

Almost in every Servlet JSP interview you will see question from error handling. You can define two kinds of error pages in Java web application, one is using tag <error-page> in web.xml and other is by using error page JSP which uses isErrorpage to declare that this jsp page can be used as error page. Other JSP uses that page by using attribute errorpage="error.jsp". Whenever you get an unhandled exception in JSP, request will be routed to error page. See [How to use error page in JSP](http://javarevisited.blogspot.sg/2012/01/error-page-in-java-web-application.html) for detailed answer of this question.

**Question 4: Difference between sendredirect and forward in Servlet ?**

One of the classical Interview Question from Servlet and JSP. This question is as old as [Vector vs ArrayList](http://java67.blogspot.sg/2012/09/arraylist-vs-vector-in-java-interview.html) in core Java. Anyway see[difference between sendredirect and forward](http://javarevisited.blogspot.sg/2011/09/sendredirect-forward-jsp-servlet.html) to answer this Servlet Interview question.

**Question 5: How do remove variable using <c:set> tag from JSTL ?**

This is one of the tricky Servlet JSP question. Many people assumes that <c:set> can only add or set variables in a particular scope but you can also remove any variable from any scope using JSTL <c:set> tag. See [How to use <c:set> JSTL tag in JSP](http://javarevisited.blogspot.sg/2012/02/jstl-tag-examples-in-jsp-java-j2ee.html) for exact way to remove any variable from any scope in JSP page.

**Question 6: What is difference between Web Server and Application Server ?**

This is rather simple Servlet JSP question to answer. If you have used EJB then you should know that , Web Server doesn't contain EJB container and EJB can not be deployed on that. Application Server is used to deploy and run EJB in J2EE environment. See [5 difference between Application and Web Server](http://javarevisited.blogspot.sg/2012/05/5-difference-between-application-server.html) to see more differences.

**Question 7: What is difference between URL Encoding and URL rewriting ?**

URL Encoding and URL rewriting is general web concept irrespective of Java programming language. URL Encoding refers to encoding URL e.g. replacing space with %20, you can see this when browser sends request to Server. On the other hand URL Rewriting is way to manage Session in web application. See [URL Encoding vs URL Rewriting](http://javarevisited.blogspot.sg/2012/01/url-rewriting-url-encoding-in-servlet.html) for more differences.

**Question 8: How do you get ServletContext reference inside Servlet ?**

ServletContext is very important object in Java Web application. ServletContext object is one per web application and serves as application scope and use to store common config and things which doesn't change on per session. Its easy to get reference ofServletContext in jsp using application implicit variable but in Servlet is not available in HttpServletRequest until version 3.0. You need HttpSession object to retrieve ServletContext in any Servlet. A good follow-up questions is How do you getServletContext in Struts Action class or Spring Controller classes. See [How to retrieve ServletContext in Servlet, Spring and Struts](http://java67.blogspot.sg/2012/09/difference-between-servletconfig-and-servletcontext-j2ee-jsp.html)for more details.

**Question 9: What is difference between ServletContext and ServletConfig in Java ?**

One of those classical Servlet Interview Questions which you can't afford to miss. Again ServletContext is used to provide application wide configure while ServletConfig is used to configure and provide initialization parameter to one Servlet. See[ServletContext vs ServletConfig](http://java67.blogspot.sg/2012/09/difference-between-servletconfig-and-servletcontext-j2ee-jsp.html) for more differences.

**Question 10: Which open source tag library have you used ?**

This is an interesting Servlet JSP questions and gives an opportunity to show how many tag library you are familiar with and which ones have you used. Most J2EE programmer answer this question with saying JSTL core tag library, Struts tag library , Spring tag library or [display tag](http://javarevisited.blogspot.sg/2011/09/displaytag-examples-tutorial-jsp-struts.html), which is quite popular tag library to display tabular data and provides lot of feature out of box e.g. paging, sorting and export functionality.

**Question 11: What is difference between GET and POST method in HTTP protocol?**

Another classical web interview question, not specific to Servlet or JSP but very important in context of web development which is based on HTTP protocol. There several differences between GET and POST method including length of data required to Send to Server. GET is less secure and can only send limited data hence not useful to transfer sensitive information. See [GET vs POST HTTP method](http://javarevisited.blogspot.sg/2012/03/get-post-method-in-http-and-https.html) for more differences.

**Question 12: What does load-on-start-up element in web.xml do?**

One of the tough Servlet JSP Interview question especially to 2 years experience guy, who may not be exposed to all tags of web.xml.load-on-startup is related to loading of Servlet. See [What is load-on-startup tag in web.xml](http://javarevisited.blogspot.sg/2011/12/load-on-startup-servlet-webxml-example.html) for more details.

These were some Servlet and JSP Interview questions and answer you can look for quick revision purpose. If you have faced any interesting Servlet JSP question or if you are looking answer for any Servlet JSP question than please share.

# [Java Best Practices to Follow while Overloading Method and Constructor](http://javarevisited.blogspot.in/2013/01/java-best-practices-method-overloading-constructor.html)

Method overloading in Java needs to be use carefully. Poorly overloaded method add not only add confusions among developers who use that but also they are error prone and leaves your program on compiler's mercy to select proper method. It's best to avoid issues related to [method overloading in Java](http://javarevisited.blogspot.sg/2011/08/what-is-polymorphism-in-java-example.html) by following some Java best practices. For those who doesn’t know What is method overloading in Java, methodoverloading means declaring two method with same name but different [method signatures](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html). This is generally done to create method which operates on different data types e.g. System.out.println() which is overloaded to accept different types of parameters like String, double, int etc, see this Java tutorial on [method overloading](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html) and [static vs dynamic binding](http://javarevisited.blogspot.sg/2012/03/what-is-static-and-dynamic-binding-in.html) for more details. By the way all of these Java best practices which are explained in context of method overloading are equally applicable to [constructor overloading in Java](http://javarevisited.blogspot.sg/2012/01/what-is-constructor-overloading-in-java.html), because in terms of overloading method and constructors are almost same.

## Java Best Practices -  Method Overloading

[Java best practices for method and constructor overloading in Java](http://3.bp.blogspot.com/-K6q0DQ1v-tw/TWu8owBtc2I/AAAAAAAAADA/oBoHDBiJ8ag/s1600/17.jpg)Here are some of the common things which you can remember while overloading method or [constructor in Java](http://javarevisited.blogspot.sg/2012/12/what-is-constructor-in-java-example-chainning-overloading.html). These Java best practices are completely based upon experience and you may have some more to add on this list. let’s see my list of Java best practices while overloading method in Java.

**1) Don't overload method which accept same number of parameter with similar types**

Two overloaded method which accept same number of argument with similar types i.e. which follow same [type hierarchy](http://javarevisited.blogspot.sg/2012/12/what-is-type-casting-in-java-class-interface-example.html) is most common mistake while overloading method  in Java. For example, find out which version of overloaded method will be invoked in following scenario :

**import** java.util.ArrayList;  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
/\*\*  
 \* Java program to demonstrate some best practice to following while overloading

 \* method in Java.This Java program shows a case of confusing method overloading in Java

 \*  
 \* @author Javin Paul  
 \*/  
**public** **class** OverloadingTest {  
    
    **public** **static** **void** main(**String** args[]){  
       **List** abc = **new** **ArrayList**();  
       **List** bcd = **new** **LinkedList**();  
        
       ConfusingOverloading co = **new** ConfusingOverloading();  
       co.hasDuplicates(abc); *//should call to ArryList overloaded method*  
       co.hasDuplicates(bcd); *//should call to LinkedList overloaded method*  
    }  
  
    
}  
  
**class** ConfusingOverloading{  
    
    **public** **boolean** hasDuplicates (**List** collection){  
        **System**.out.println("overloaded method with Type List ");  
        **return** **true**;  
    }  
    
    **public** **boolean** hasDuplicates (**ArrayList** collection){  
        **System**.out.println("overloaded method with Type ArrayList ");  
        **return** **true**;  
    }  
    
    
    **public** **boolean** hasDuplicates (**LinkedList** collection){  
        **System**.out.println("overloaded method with Type LinkedList ");  
        **return** **true**;  
    }  
    
}  
  
**Output**  
overloaded method with **Type** **List**  
overloaded method with **Type** **List**

To surprise of some programmers method with argument type List is called both the time, instead of expected method which takes [ArrayList](http://javarevisited.blogspot.sg/2011/05/example-of-arraylist-in-java-tutorial.html) and [LinkedList](http://javarevisited.blogspot.sg/2012/02/difference-between-linkedlist-vs.html), because method overloading is resolved at compile time using static binding in Java. This is  also one of the reason, why its important to clearly understand [difference between method overloading and overriding in Java](http://javarevisited.blogspot.sg/2011/12/method-overloading-vs-method-overriding.html). Here expected case is result of mistaking overloading as overriding, which work on actual object and happens at runtime. To know more about static and dynamic binding in Java , you can also see my post [difference between static and dynamic binding in Java](http://javarevisited.blogspot.com/2012/03/what-is-static-and-dynamic-binding-in.html).

**2) Use radically different types while overloading method in Java**

It's completely legal and there is no ambiguity when two overloaded method accepts radically different types like String and Integer. Though both overloaded method will accept only one parameter, it’s still clear which method is called because both types are completely different to each other. Both programmer and compiler both know which method will be invoked for a particular call. One of the example of this kind of overloading is constructor of java.util.Scanner class which accepts [File](http://javarevisited.blogspot.sg/2012/07/read-file-line-by-line-java-example-scanner.html), [InputStream](http://javarevisited.blogspot.sg/2012/08/convert-inputstream-to-string-java-example-tutorial.html) or String as parameter, as shown below :

Scanner(File source)

Scanner(InputStream source)

Scanner(String source)

**3) Beware of Autoboxing while overloading method in Java**

Prior to introduction of [Autoboxing and unboxing in Java 5](http://javarevisited.blogspot.sg/2012/07/auto-boxing-and-unboxing-in-java-be.html), method which accept primitive type and object type were radically different and it’s clear which method will be invoked. Now with autoboxing it's really confusing. Clasical example of this kind overloading mistake is ArrayList’s  remove() method, which is overloaded to accept index as well as Object. when you store Integer in ArrayList and call remove() method, It’s hard to find out which remove() method will be called, as shown in below example :

**List**<**Integer**> numbers = **new** **ArrayList**<**Integer**>();  
numbers.add(1);  
numbers.add(2);  
numbers.add(3);  
**System**.out.println("numbers: " + numbers);  
numbers.remove(1); *//should remove "1" as element or 2nd element from ArrayList*  
**System**.out.println("numbers: " + numbers);  
  
**Output:**  
numbers: [1, 2, 3]  
numbers: [1, 3]

Many Java programmer expect that Integer(1) object would  be removed but since remove() is overloaded, compiler choose remove(int) over remove(Object). Rules of which overloaded method gets chosen in case of [autoboxing](http://javarevisited.blogspot.sg/2010/10/what-is-problem-while-using-in.html) is complex and hard to remember, so Its best to avoid two [overloaded method](http://java67.blogspot.sg/2012/08/what-is-method-overloading-in-java-example.html) where one accept Object and other accept primitive type. If by any chance you must have to do this then make sure both of them perform identical function.

Read more: [http://javarevisited.blogspot.com/2013/01/java-best-practices-method-overloading-constructor.html#ixzz2OL3w66tJ](http://javarevisited.blogspot.com/2013/01/java-best-practices-method-overloading-constructor.html" \l "ixzz2OL3w66tJ)

# [Why main method is public static in Java](http://javarevisited.blogspot.in/2011/12/main-public-static-java-void-method-why.html)

**Main method in Java** is the first programming method a Java programmer knows when he starts learning Java programming language.have you ever thought about **why main method in Java is public, static and void**, of-course Yes, since most of us first learn C and C++ than we move to Java in our programming path we familiar with main method but in Java mainmethod is slightly different it doesn't return any value like in C it returns int, *main method is public static and void Why*? In this post we will try to find answer of these questions and have an idea of one of the most popular questions in Java why main method is declared Static.

### What is main method in Java?

Main method in Java is entry point for any core Java program. Remember we are not talking about Servlet, MIDlet or any other container managed Java program where life cycle methods are provided to control the execution. In core Java program, execution starts from main method when you type java main-class-name, JVM search for **public static void main(Stringargs[])** method in that class and if it doesn't find that method it throws error **NoSuchMethodError:main** and terminates.

**Signature of main method in Java**

Main method has to strictly follow its syntax; other wise JVM will not be able to locate it and your program will not run. Here is the exact signature of main method

**public static void main(String args[])**

This signature is classic signature and there from start of Java but with introduction of  [variable argument or varargs in Java5](http://javarevisited.blogspot.com/2011/09/variable-argument-in-java5-varargs.html) you can also declare main method in Java using varargs syntax as shown in below example:

**public static void main(String... args)**

Remember varargs version of java main method will only work in Java 1.5 or later version. Apart from public, static and void there are certain keywords like final, synchronized and strictfp which are permitted in signature of java main method.

### Why main method is static in Java

[why main method is public static void in Java](http://javarevisited.blogspot.com/2011/10/override-hashcode-in-java-example.html)Now come to the main point "Why main method is static in Java", there are quite a few reasons around but here are few reasons which make sense to me:

1. Since main method is static Java virtual Machine can call it without creating any instance of class which contains main method.

2. Since C and C++ also has similar main method which serves as entry point for program execution, following that convention will only help Java.

3. If main method were not declared static than JVM has to create instance of main Class and since constructor can be overloaded and can have arguments there would not be any certain and consistent way for **JVM to find main method in Java**.

4. Anything which is declared in [class in Java](http://javarevisited.blogspot.com/2011/10/class-in-java-programming-general.html) comes under reference type and requires object to be created before using them but static method and static data are loaded into separate memory inside JVM called context which is created when a class is loaded. If main method is static than it will be loaded in JVM context and are available to execution.

**Why main mehtod is public in Java**

Java specifies several access modifiers e.g. private, protected and public. Any method or variable which is declared public in Java can be accessible from outside of that class. Since main method is public in

Java, JVM can easily access and execute it.

**Why main method is void in Java**

Since main method in Java is not supposed to return any value, its made void which simply means main is not returning anything.

**Summary:**

1. Main method must be declared **public, static and void in Java** otherwise JVM will not able to run Java program.

2. JVM throws **NoSuchMethodException:main** if it doesn't find main method of predefined signature in class which is provided to Java command. E.g. if you run java Helloworld than JVM will search for public static void main String args[]) method in HelloWorld.class file.

3. Main method is entry point for any Core Java program. Execution starts from main method.

4. Main method is run by a special thread called ["main" thread in Java](http://javarevisited.blogspot.com/2011/02/how-to-implement-thread-in-java.html). Your Java program will be running until your main thread is running or any non-daemon thread spawned from main method is running.

5. When you see **"Exception in Thread main”** e.g.

**Exception in Thread main: Java.lang.NullPointerException** it means Exception is thrown inside main thread.

6. You can declare main method using varargs syntax from Java 1.5 onwards e.g.

**public static void main(String... args)**

7. Apart from static, void and public you can use final, synchronized and strictfp modifier in signature of main method in Java.

8. Main method in Java can be overloaded like any other method in Java but JVM will only call main method with specified signature specified above.

9. You can use throws clause in signature of main method and can throw any checked or unchecked Exception.

10. [Static initializer block](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html) is executed even before JVM calls main method. They are executed when a Class is loaded into Memory by JVM.

Read more: [http://javarevisited.blogspot.com/2011/12/main-public-static-java-void-method-why.html#ixzz2OL4AJCzk](http://javarevisited.blogspot.com/2011/12/main-public-static-java-void-method-why.html" \l "ixzz2OL4AJCzk)

# [Why main method is public static in Java](http://javarevisited.blogspot.in/2011/12/main-public-static-java-void-method-why.html)

**Main method in Java** is the first programming method a Java programmer knows when he starts learning Java programming language.have you ever thought about **why main method in Java is public, static and void**, of-course Yes, since most of us first learn C and C++ than we move to Java in our programming path we familiar with main method but in Java mainmethod is slightly different it doesn't return any value like in C it returns int, *main method is public static and void Why*? In this post we will try to find answer of these questions and have an idea of one of the most popular questions in Java why main method is declared Static.

### What is main method in Java?

Main method in Java is entry point for any core Java program. Remember we are not talking about Servlet, MIDlet or any other container managed Java program where life cycle methods are provided to control the execution. In core Java program, execution starts from main method when you type java main-class-name, JVM search for **public static void main(Stringargs[])** method in that class and if it doesn't find that method it throws error **NoSuchMethodError:main** and terminates.

**Signature of main method in Java**

Main method has to strictly follow its syntax; other wise JVM will not be able to locate it and your program will not run. Here is the exact signature of main method

**public static void main(String args[])**

This signature is classic signature and there from start of Java but with introduction of  [variable argument or varargs in Java5](http://javarevisited.blogspot.com/2011/09/variable-argument-in-java5-varargs.html) you can also declare main method in Java using varargs syntax as shown in below example:

**public static void main(String... args)**

Remember varargs version of java main method will only work in Java 1.5 or later version. Apart from public, static and void there are certain keywords like final, synchronized and strictfp which are permitted in signature of java main method.

### Why main method is static in Java

[why main method is public static void in Java](http://javarevisited.blogspot.com/2011/10/override-hashcode-in-java-example.html)Now come to the main point "Why main method is static in Java", there are quite a few reasons around but here are few reasons which make sense to me:

1. Since main method is static Java virtual Machine can call it without creating any instance of class which contains main method.

2. Since C and C++ also has similar main method which serves as entry point for program execution, following that convention will only help Java.

3. If main method were not declared static than JVM has to create instance of main Class and since constructor can be overloaded and can have arguments there would not be any certain and consistent way for **JVM to find main method in Java**.

4. Anything which is declared in [class in Java](http://javarevisited.blogspot.com/2011/10/class-in-java-programming-general.html) comes under reference type and requires object to be created before using them but static method and static data are loaded into separate memory inside JVM called context which is created when a class is loaded. If main method is static than it will be loaded in JVM context and are available to execution.

**Why main mehtod is public in Java**

Java specifies several access modifiers e.g. private, protected and public. Any method or variable which is declared public in Java can be accessible from outside of that class. Since main method is public in

Java, JVM can easily access and execute it.

**Why main method is void in Java**

Since main method in Java is not supposed to return any value, its made void which simply means main is not returning anything.

**Summary:**

1. Main method must be declared **public, static and void in Java** otherwise JVM will not able to run Java program.

2. JVM throws **NoSuchMethodException:main** if it doesn't find main method of predefined signature in class which is provided to Java command. E.g. if you run java Helloworld than JVM will search for public static void main String args[]) method in HelloWorld.class file.

3. Main method is entry point for any Core Java program. Execution starts from main method.

4. Main method is run by a special thread called ["main" thread in Java](http://javarevisited.blogspot.com/2011/02/how-to-implement-thread-in-java.html). Your Java program will be running until your main thread is running or any non-daemon thread spawned from main method is running.

5. When you see **"Exception in Thread main”** e.g.

**Exception in Thread main: Java.lang.NullPointerException** it means Exception is thrown inside main thread.

6. You can declare main method using varargs syntax from Java 1.5 onwards e.g.

**public static void main(String... args)**

7. Apart from static, void and public you can use final, synchronized and strictfp modifier in signature of main method in Java.

8. Main method in Java can be overloaded like any other method in Java but JVM will only call main method with specified signature specified above.

9. You can use throws clause in signature of main method and can throw any checked or unchecked Exception.

10. [Static initializer block](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html) is executed even before JVM calls main method. They are executed when a Class is loaded into Memory by JVM.

Read more: [http://javarevisited.blogspot.com/2011/12/main-public-static-java-void-method-why.html#ixzz2OL4HV0gn](http://javarevisited.blogspot.com/2011/12/main-public-static-java-void-method-why.html" \l "ixzz2OL4HV0gn)=====

# [Difference between Thread vs Runnable interface in Java](http://javarevisited.blogspot.in/2012/01/difference-thread-vs-runnable-interface.html)

**Thread vs Runnable in Java** is always been a confusing decision for beginners  in java. [Thread in Java](http://javarevisited.blogspot.com/2011/02/how-to-implement-thread-in-java.html) seems easy in comparison of Runnable because you just deal with one class **java.lang.Thread** while in case of using Runnable to implement Thread you need to deal with both Thread and Runnable two classes. though decision of using Runnable or Thread should be taken considering **differences between Runnable and Thread** and pros and cons of both approaches. This is also a very [popular thread interview questions](http://javarevisited.blogspot.com/2011/07/java-multi-threading-interview.html) and most of interviewer are really interested to know what is your point of view while choosing *Thread vs Runnable or opposite*. In this java article we will try to point out some*differences between Thread and Runnable in Java* which will help you to take an informed decision.

## Difference between Thread and Runnable interface in Java

### Thread vs Runnable in Java

[Difference between Thread vs Runnable in Java](http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html)Here are some of my thoughts on whether I should use **Thread or Runnable** for implementing task in Java, though you have another choice as "Callable" for implementing thread which we will discuss later.

1) [Java doesn't support multiple inheritance](http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html), which means you can only extend one class in Java so once you extended Thread class you lost your chance and can not extend or inherit another [class in Java](http://javarevisited.blogspot.com/2011/10/class-in-java-programming-general.html).

2) In Object oriented programming extending a class generally means adding new functionality, modifying or improving behaviors. If we are not making any modification on Thread than use Runnable interface instead.

3) **Runnable**interface represent a Task which can be executed by either plain Thread or Executors or any other means. so logical separation of Task as Runnable than Thread is good design decision.

4) Separating task as **Runnable**means we can reuse the task and also has liberty to execute it from different means. since you can not restart a Thread once it completes. again **Runnable vs Thread** for task, Runnable is winner.

5) Java designer recognizes this and that's why Executors accept Runnable as Task and they have worker thread which executes those task.

6) Inheriting all Thread methods are additional overhead just for representing a Task which can can be done easily with Runnable.

These were some of notable **difference between Thread and Runnable in Java**, if you know any other differences on Thread vs Runnable than please share it via comments. I personally use Runnable over Thread for this scenario and recommends to use Runnable or Callable interface based on your requirement.

Read more: [http://javarevisited.blogspot.com/2012/01/difference-thread-vs-runnable-interface.html#ixzz2OL4L4ZSr](http://javarevisited.blogspot.com/2012/01/difference-thread-vs-runnable-interface.html" \l "ixzz2OL4L4ZSr)

|  |  |
| --- | --- |
| Joined: Sep 28, 2004 Posts: 16429     16  [I like...](http://www.coderanch.com/how-to/java/BumperStickers) C++ Eclipse IDE Firefox Browser | posted Monday, November 16, 2009 19:23:51 private message  http://cache-www.coderanch.com/templates/default/images/thumbs-up.gif  Quote |
| The Hashtable class is a Map implementation that was part of Java since version 1.0. It was "obsoleted" with Hashmap (wrapped with synchronizedMap) in Java 1.2, and but was ported to work just as well as a synchronized Hashmap.   [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html) was added with Java 1.5, as part of the concurrency enhancements. Some features supported by it are ... (1) implemented using optimistic locking, (2) uses reader-writer locks to support simulataneous reads, and (3) uses segmentation to support simultaneous writes.   Henry |
| Books: [Java Threads, 3rd Edition](http://www.amazon.com/exec/obidos/ASIN/0596007825/ref=jranch-20), [Jini in a Nutshell](http://www.amazon.com/Jini-Nutshell-OReilly-Scott-Oaks/dp/1565927591/ref=jranch-20" \t "_new), and [Java Gems (contributor)](http://www.amazon.com/Java-Gems-Dwight-Deugo/dp/0521648246/ref=jranch-20) | |
| http://cache-www.coderanch.com/templates/default/images/spacer.gif | |  |
| **Kumar Jaya**  Ranch Hand   Joined: Jan 12, 2009 Posts: 45 | posted Monday, November 16, 2009 19:43:06 private message  http://cache-www.coderanch.com/templates/default/images/thumbs-up.gif  Quote |  |
| Hello Henry,   What is the purpose of optimistic locking in [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html" \o "Java API" \t "_new)?? Isnt [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html" \o "Java API" \t "_new) locked when accessed by any thread anyways?? Does the implementation take care of it or we have to externally use method to lock it optimistically??   Regards  Jaya |  |
| http://cache-www.coderanch.com/templates/default/images/spacer.gif | |  |
| **Henry Wong**  author  Sheriff   Joined: Sep 28, 2004 Posts: 16429     16  [I like...](http://www.coderanch.com/how-to/java/BumperStickers) C++ Eclipse IDE Firefox Browser | posted Monday, November 16, 2009 20:34:46 private message  http://cache-www.coderanch.com/templates/default/images/thumbs-up.gif  Quote |  |
| **Kumar Jaya wrote:**What is the purpose of optimistic locking in [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html" \o "Java API" \t "_new)?? Isnt [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html" \o "Java API" \t "_new) locked when accessed by any thread anyways?? Does the implementation take care of it or we have to externally use method to lock it optimistically??  Yeah, using optimistic locking probably doesn't gain much in this regard, as the design is to lock each method, but why not?   Regardless, its an implementation detail -- as a user of the [ConcurrentHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentHashMap.html" \o "Java API" \t "_new), you simply know that the class is threadsafe, even though it doesn't use the synchronization mechanism.   Henry |  |