Java Multithreading Concurrency Interview Questions and Answers

APRIL 3, 2018 BY PANKAJ — 55 COMMENTS

Today we will go through Java Multithreading Interview Questions and Answers. We will also look into Concurrency interview questions and answers because both multithreading and concurrency go hand in hand. Thread is one of the popular topic in java interview questions. Here I am listing down most of the important java multithreading interview questions from interview perspective, but you should have good knowledge on java threads to deal with follow up questions.

Java Multithreading Interview Questions

1. What is the difference between Process and Thread?
2. What are the benefits of multi-threaded programming?
3. What is difference between user Thread and daemon Thread?
4. How can we create a Thread in Java?
5. What are different states in lifecycle of Thread?
6. Can we call run() method of a Thread class?
7. How can we pause the execution of a Thread for specific time?
8. What do you understand about Thread Priority?
9. What is Thread Scheduler and Time Slicing?
10. What is context-switching in multi-threading?
11. How can we make sure main() is the last thread to finish in Java Program?
12. How does thread communicate with each other?
13. Why thread communication methods wait(), notify() and notifyAll() are in Object class?
14. Why wait(), notify() and notifyAll() methods have to be called from synchronized method or block?
15. Why Thread sleep() and yield() methods are static?
16. How can we achieve thread safety in Java?
17. What is volatile keyword in Java
18. Which is more preferred – Synchronized method or Synchronized block?
19. How to create daemon thread in Java?
20. What is ThreadLocal?
21. What is Thread Group? Why it's advised not to use it?
22. What is Java Thread Dump, How can we get Java Thread dump of a Program?
23. What is Deadlock? How to analyze and avoid deadlock situation?
24. What is Java Timer Class? How to schedule a task to run after specific interval?
25. What is Thread Pool? How can we create Thread Pool in Java?
26. What will happen if we don't override Thread class run() method?

Java Concurrency Interview Questions

1. What is atomic operation? What are atomic classes in Java Concurrency API?
2. What is Lock interface in Java Concurrency API? What are it's benefits over synchronization?

Java Multithreading Interview Questions and Answers

1. What is the difference between Process and Thread?

A process is a self contained execution environment and it can be seen as a program or application whereas Thread is a single task of execution within the process. Java runtime environment runs as a single process which contains different classes and programs as processes. Thread can be called lightweight process. Thread requires less resources to create and exists in the process, thread shares the process resources.

2. What are the benefits of multi-threaded programming?

In Multi-Threaded programming, multiple threads are executing concurrently that improves the performance because CPU is not idle incase some thread is waiting to get some resources. Multiple threads share the heap memory, so it's good to create multiple threads to execute some task rather than creating multiple processes. For example, Servlets are better in performance than CGI because Servlet support multi-threading but CGI doesn't.

3. What is difference between user Thread and daemon Thread?

When we create a Thread in java program, it's known as user thread. A daemon thread runs in background and doesn't prevent JVM from terminating. When there are no user threads running, JVM shutdown the program and quits. A child thread created from daemon thread is also a daemon thread.

4. How can we create a Thread in Java?

There are two ways to create Thread in Java – first by implementing Runnable interface and then creating a Thread object from it and second is to extend the Thread Class. Read this post to learn more about creating threads in java.

5. What are different states in lifecycle of Thread?

When we create a Thread in java program, its state is New. Then we start the thread that change it's state to Runnable. Thread Scheduler is responsible to allocate CPU to threads in Runnable thread pool and change their state to Running. Other Thread states are Waiting, Blocked and Dead. Read this post to learn more about life cycle of thread.

6. Can we call run() method of a Thread class?

Yes, we can call run() method of a Thread class but then it will behave like a normal method. To actually execute it in a Thread, we need to start it using **Thread.start()** method.

7. How can we pause the execution of a Thread for specific time?

We can use Thread class sleep() method to pause the execution of Thread for certain time. Note that this will not stop the processing of thread for specific time, once the thread awake from sleep, it's state gets changed to runnable and based on thread scheduling, it gets executed.

8. What do you understand about Thread Priority?

Every thread has a priority, usually higher priority thread gets precedence in execution but it depends on Thread Scheduler implementation that is OS dependent. We can specify the priority of thread but it doesn't guarantee that higher priority thread will get executed before lower priority thread. Thread priority is an *int* whose value varies from 1 to 10 where 1 is the lowest priority thread and 10 is the highest priority thread.

9. What is Thread Scheduler and Time Slicing?

Thread Scheduler is the Operating System service that allocates the CPU time to the available runnable threads. Once we create and start a thread, it's execution depends on the implementation of Thread Scheduler. Time Slicing is the process to divide the available CPU time to the available runnable threads. Allocation of CPU time to threads can be based on thread priority or the thread waiting for longer time will get more priority in getting CPU time. Thread scheduling can't be controlled by java, so it's always better to control it from application itself.

10. What is context-switching in multi-threading?

Context Switching is the process of storing and restoring of CPU state so that Thread execution can be resumed from the same point at a later point of time. Context Switching is the essential feature for multitasking operating system and support for multi-threaded environment.

11. How can we make sure main() is the last thread to finish in Java Program?

We can use Thread join() method to make sure all the threads created by the program is dead before finishing the main function. Here is an article about Thread join method.

12. How does thread communicate with each other?

When threads share resources, communication between Threads is important to coordinate their efforts. Object class wait(), notify() and notifyAll() methods allows threads to communicate about the lock status of a resource. Check this post to learn more about thread wait, notify and notifyAll.

13. Why thread communication methods wait(), notify() and notifyAll() are in Object class?

In Java every Object has a monitor and wait, notify methods are used to wait for the Object monitor or to notify other threads that Object monitor is free now. There is no monitor on threads in java and synchronization can be used with any Object, that's why it's part of Object class so that every class in java has these essential methods for inter thread communication.

14. Why wait(), notify() and notifyAll() methods have to be called from synchronized method or block?

When a Thread calls wait() on any Object, it must have the monitor on the Object that it will leave and goes in wait state until any other thread call notify() on this Object. Similarly when a thread calls notify() on any Object, it leaves the monitor on the Object and other waiting threads can get the monitor on the Object. Since all these methods require Thread to have the Object monitor, that can be achieved only by synchronization, they need to be called from synchronized method or block.

15. Why Thread sleep() and yield() methods are static?

Thread sleep() and yield() methods work on the currently executing thread. So there is no point in invoking these methods on some other threads that are in wait state. That's why these methods are made static so that when this method is called statically, it works on the current executing thread and avoid confusion to the programmers who might think that they can invoke these methods on some non-running threads.

16. How can we achieve thread safety in Java?

There are several ways to achieve thread safety in java – synchronization, atomic concurrent classes, implementing concurrent Lock interface, using volatile keyword, using immutable classes and Thread

safe classes. Learn more at thread safety tutorial.

17. What is volatile keyword in Java

When we use volatile keyword with a variable, all the threads read it's value directly from the memory and don't cache it. This makes sure that the value read is the same as in the memory.

18. Which is more preferred – Synchronized method or Synchronized block?

Synchronized block is more preferred way because it doesn't lock the Object, synchronized methods lock the Object and if there are multiple synchronization blocks in the class, even though they are not related, it will stop them from execution and put them in wait state to get the lock on Object.

19. How to create daemon thread in Java?

Thread class setDaemon(true) can be used to create daemon thread in java. We need to call this method before calling start() method else it will throw IllegalThreadStateException.

20. What is ThreadLocal?

Java ThreadLocal is used to create thread-local variables. We know that all threads of an Object share it's variables, so if the variable is not thread safe, we can use synchronization but if we want to avoid synchronization, we can use ThreadLocal variables.
Every thread has it's own ThreadLocal variable and they can use it's get() and set() methods to get the default value or change it's value local to Thread. ThreadLocal instances are typically private static fields in classes that wish to associate state with a thread. Check this post for small example program showing ThreadLocal Example.

21. What is Thread Group? Why it's advised not to use it?

ThreadGroup is a class which was intended to provide information about a thread group. ThreadGroup API is weak and it doesn't have any functionality that is not provided by Thread. Two of the major feature it had are to get the list of active threads in a thread group and to set the uncaught exception handler for the thread. But Java 1.5 has added *setUncaughtExceptionHandler(UncaughtExceptionHandler eh)* method using which we can add uncaught exception handler to the thread. So ThreadGroup is obsolete and hence not advised to use anymore.

```
t1.setUncaughtExceptionHandler(new UncaughtExceptionHandler(){

@Override
public void uncaughtException(Thread t, Throwable e) {
    System.out.println("exception occured:"+e.getMessage());
```

```
    }

    });
```

22. What is Java Thread Dump, How can we get Java Thread dump of a Program?

Thread dump is list of all the threads active in the JVM, thread dumps are very helpful in analyzing bottlenecks in the application and analyzing deadlock situations. There are many ways using which we can generate Thread dump – Using Profiler, Kill -3 command, jstack tool etc. I prefer jstack tool to generate thread dump of a program because it's easy to use and comes with JDK installation. Since it's a terminal based tool, we can create script to generate thread dump at regular intervals to analyze it later on. Read this post to know more about generating thread dump in java.

23. What is Deadlock? How to analyze and avoid deadlock situation?

Deadlock is a programming situation where two or more threads are blocked forever, this situation arises with at least two threads and two or more resources.

To analyze a deadlock, we need to look at the java thread dump of the application, we need to look out for the threads with state as BLOCKED and then the resources it's waiting to lock, every resource has a unique ID using which we can find which thread is already holding the lock on the object.

Avoid Nested Locks, Lock Only What is Required and Avoid waiting indefinitely are common ways to avoid deadlock situation, read this post to learn how to analyze deadlock in java with sample program.

24. What is Java Timer Class? How to schedule a task to run after specific interval?

java.util.Timer is a utility class that can be used to schedule a thread to be executed at certain time in future. Java Timer class can be used to schedule a task to be run one-time or to be run at regular intervals.

java.util.TimerTask is an **abstract class** that implements Runnable interface and we need to extend this class to create our own TimerTask that can be scheduled using java Timer class.

Check this post for java Timer example.

25. What is Thread Pool? How can we create Thread Pool in Java?

A thread pool manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed.

A thread pool manages the collection of Runnable threads and worker threads execute Runnable from the queue.

java.util.concurrent.Executors provide implementation of java.util.concurrent.Executor interface to create the thread pool in java. Thread Pool Example program shows how to create and use Thread Pool in java. Or read ScheduledThreadPoolExecutor Example to know how to schedule tasks after certain delay.

26. What will happen if we don't override Thread class run() method?

Thread class run() method code is as shown below.

```
public void run() {
    if (target != null) {
        target.run();
    }
}
```

Above target set in the init() method of Thread class and if we create an instance of Thread class as `new TestThread()`, it's set to null. So nothing will happen if we don't override the run() method. Below is a simple example demonstrating this.

```
public class TestThread extends Thread {

        //not overriding Thread.run() method

        //main method, can be in other class too
        public static void main(String args[]){
                Thread t = new TestThread();
                System.out.println("Before starting thread");
                t.start();
                System.out.println("After starting thread");
        }
}
```

It will print only below output and terminate.

```
Before starting thread
After starting thread
```

Java Concurrency Interview Questions and Answers

1. What is atomic operation? What are atomic classes in Java Concurrency API?

Atomic operations are performed in a single unit of task without interference from other operations. Atomic operations are necessity in multi-threaded environment to avoid data inconsistency.

int++ is not an atomic operation. So by the time one threads read it's value and increment it by one, other thread has read the older value leading to wrong result.

To solve this issue, we will have to make sure that increment operation on count is atomic, we can do that using Synchronization but Java 5 java.util.concurrent.atomic provides wrapper classes for int and long that can be used to achieve this atomically without usage of Synchronization. Go to this article to learn more about atomic concurrent classes.

2. What is Lock interface in Java Concurrency API? What are it's benefits over synchronization?

Lock interface provide more extensive locking operations than can be obtained using synchronized methods and statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated Condition objects.
The advantages of a lock are

- it's possible to make them fair
- it's possible to make a thread responsive to interruption while waiting on a Lock object.
- it's possible to try to acquire the lock, but return immediately or after a timeout if the lock can't be acquired
- it's possible to acquire and release locks in different scopes, and in different orders

Read more at **Java Lock Example**.

3. What is Executors Framework?

In Java 5, Executor framework was introduced with the java.util.concurrent.Executor interface.

The Executor framework is a framework for standardizing invocation, scheduling, execution, and control of asynchronous tasks according to a set of execution policies.

Creating a lot many threads with no bounds to the maximum threshold can cause application to run out of heap memory. So, creating a ThreadPool is a better solution as a finite number of threads can be pooled and reused. Executors framework facilitate process of creating Thread pools in java. Check out this post to learn with example code to create thread pool using Executors framework.

4. What is BlockingQueue? How can we implement Producer-Consumer problem using Blocking Queue?

java.util.concurrent.BlockingQueue is a Queue that supports operations that wait for the queue to become non-empty when retrieving and removing an element, and wait for space to become available in the queue when adding an element.

BlockingQueue doesn't accept null values and throw NullPointerException if you try to store null value in the queue.

BlockingQueue implementations are thread-safe. All queuing methods are atomic in nature and use internal locks or other forms of concurrency control.

BlockingQueue interface is part of java collections framework and it's primarily used for implementing producer consumer problem.
Check this post for producer-consumer problem implementation using BlockingQueue.

5. What is Callable and Future?

Java 5 introduced java.util.concurrent.Callable interface in concurrency package that is similar to Runnable interface but it can return any Object and able to throw Exception.

Callable interface use Generic to define the return type of Object. Executors class provide useful methods to execute Callable in a thread pool. Since callable tasks run in parallel, we have to wait for the returned Object. Callable tasks return java.util.concurrent.Future object. Using Future we can find out the status of the Callable task and get the returned Object. It provides get() method that can wait for the Callable to finish and then return the result.
Check this post for Callable Future Example.

6. What is FutureTask Class?

FutureTask is the base implementation class of Future interface and we can use it with Executors for asynchronous processing. Most of the time we don't need to use FutureTask class but it comes real handy if we want to override some of the methods of Future interface and want to keep most of the base implementation. We can just extend this class and override the methods according to our requirements. Check out **Java FutureTask Example** post to learn how to use it and what are different methods it has.

7. What are Concurrent Collection Classes?

Java Collection classes are fail-fast which means that if the Collection will be changed while some thread is traversing over it using iterator, the iterator.next() will throw ConcurrentModificationException.

Concurrent Collection classes support full concurrency of retrievals and adjustable expected concurrency for updates.

Major classes are ConcurrentHashMap, CopyOnWriteArrayList and CopyOnWriteArraySet, check this post to learn how to avoid ConcurrentModificationException when using iterator.

8. ## What is Executors Class?

Executors class provide utility methods for Executor, ExecutorService, ScheduledExecutorService, ThreadFactory, and Callable classes.

Executors class can be used to easily create Thread Pool in java, also this is the only class supporting execution of Callable implementations.

9. ## What are some of the improvements in Concurrency API in Java 8?

Some important concurrent API enhancements are:

- ConcurrentHashMap compute(), forEach(), forEachEntry(), forEachKey(), forEachValue(), merge(), reduce() and search() methods.
- CompletableFuture that may be explicitly completed (setting its value and status).
- Executors newWorkStealingPool() method to create a work-stealing thread pool using all available processors as its target parallelism level.

Recommended Read: Java 8 Features

That's all for Java Thread and Concurrency interview questions, I have been adding more to this list. So bookmark the post for future reference.

« PREVIOUS
How to write XML file in Java (DOM Parser)

NEXT »
Java StAX Parser Example to read XML file

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: INTERVIEW QUESTIONS, JAVA

Comments

hardik Nai says

OCTOBER 5, 2017 AT 7:41 PM

add please Difference between sleep and wait?

most frequently asked.
Please update page base on comment it make more effective
Reply

hardik Nai says
OCTOBER 5, 2017 AT 7:38 PM
Add question “Difference between Runnable vs Thread”
most frequently asked
Reply

Hleb says
JULY 31, 2016 AT 2:07 AM
Hello, Pankaj
Thank you for your site and for all your posts
I have a question about Callable interface
You said that
“Executors … is the only class supporting execution of Callable implementations”
And I saw the post about FutureTask where you used Callable interface too.
So there are 2 ways how to execute Callable interface:
– Executors
– FutureTask
Are you agree ?
Reply

Pankaj says
JULY 31, 2016 AT 7:00 AM
If you check the FutureTask example at below URL, you will see that it’s using Executors to execute them.
https://www.journaldev.com/1650/java-futuretask-example-program
Reply

Dalia Kamal says
NOVEMBER 6, 2015 AT 8:57 AM
Many thanks,very Organized and clear tutorial!
Reply

Free Coder says
SEPTEMBER 7, 2015 AT 11:56 AM
Thanks for sharing the questions and answers. I am sure that you missed the point or mistakenly wrote the point.
"Synchronized block is more preferred way because it doesn't lock the Object, synchronized methods lock the Object "
As per as i know, the moment the synchronized keyword is used, monitor is involved, irrespective of whether it's at method or block level. In the case of synchronized block, lock needs to be specified explicitly, whereas, in the case of synchronized method, lock is used implicitly.
Reply

Joshi says
SEPTEMBER 7, 2015 AT 9:06 AM
hi sir,
i have 10 threads without using synchronization and any priority how we can we can execute 10threads sequentially.please reply me.
Reply

Ravi says
OCTOBER 26, 2015 AT 1:34 PM
Use cirular linked list of threads
Reply

Ish says
JULY 18, 2016 AT 9:16 AM
t1.start();
t1.join();
t2.start();
t2.join();
t3.start();
t3.join();
…….
Reply

lucky says
JANUARY 6, 2018 AT 10:20 AM
use Lock () method
Reentrantlock l = new Reentrantlock();
l . lock();
// do some operation here
l . unlock();
Reply

anju says
MAY 23, 2015 AT 12:26 AM
good
but urs Answers Link is not working!
Reply

Pankaj says
MAY 23, 2015 AT 12:32 AM
I tested and all the links are working fine, which link you are talking about.
Reply

krishna says
MAY 9, 2015 AT 7:38 AM
Nicely explained.
Thanks.
Reply

JavaCoder says
JANUARY 4, 2015 AT 1:02 AM
When do we use the Volatile variables in a program ? Is there any overhead by using volatile variables?
Reply

srujana says
DECEMBER 15, 2014 AT 3:52 PM
Hi Pankaj,
In one of the interview they asked me below questions.
1. We have 4 processed and single thread is running on each process. how is it possible all these these thread which are running in different process share common data?
2.In single process 4 threads are running, how all these threads access common data?
If you get a chance could please share these types of concepts.
Reply

Saurabh Asthana says
JULY 13, 2016 AT 1:24 PM
If yu go through java memory model, you will come to know that to share common data across thread it should be static data, so that they all point to same reference.
Reply

Danny says
SEPTEMBER 15, 2014 AT 2:50 AM
sorry, the ads were not on your website, it was my browser,
unknowingly installed offerwizar ads.
Reply

Pawan Shukla says
SEPTEMBER 1, 2014 AT 6:04 AM
i have a doubt about string class . I have studied from somewhere that whenever we create an object of string class using double quotes then only one object is created and that object is stored in string constant pool .But when we create the object using new operator then two object is created one is stored in heap and second one is stored in string constant pool.
please clear my doubts..
Reply

Pankaj says
SEPTEMBER 1, 2014 AT 6:52 AM
Please read Java String Pool, it will clear all your doubts in this area.
Reply

Pawan Shukla says
SEPTEMBER 1, 2014 AT 6:00 AM
very nice topics and really a well explained answers. Thanks pankaj keep it up.
Reply

Alz says
AUGUST 6, 2014 AT 11:24 PM
Great work! Simple and understandable explanation. You're the best dude! (y)
Reply

Awadhesh says
JULY 31, 2014 AT 7:42 AM
Its an excellent collection of Java Q&A . Keep going.
Reply

Subbukumararaja says
JULY 30, 2014 AT 10:35 PM
Sweet and simple. Awesome Pankaj.
Reply

Pavan Kulkarni says
JULY 15, 2014 AT 5:32 AM
What adds to your crisp explanation is the clean UI of this site. Really appreciate your contribution.
Reply

Satya says
MAY 31, 2014 AT 11:28 PM
Very nice Pankaj. Loved your detailed explanation. I am a big fan of your articles.
Reply

uttam says
MAY 20, 2014 AT 10:33 PM
Highly Impressed.. Keep up the good work.. □
Reply

Anonymous says
MAY 11, 2014 AT 1:16 PM
Nice question. Solid explanation about callable and future and executer framework.
Reply

Steve says
APRIL 27, 2014 AT 7:46 AM
Very nicely done – thanks for the info
Reply

Martin says
APRIL 23, 2014 AT 3:47 AM
Site doesn't work with ios. It says follow on twitter/facebook etc to access the rest, and when I do, it doesnt unlock.
Reply

nelesh says
APRIL 20, 2014 AT 9:14 AM
Thanks Pankaj for the great writeup, any tips for working with xml and java?
Reply

PurnaChandra says
APRIL 19, 2014 AT 3:35 PM
Very good and helpful.
Reply

Rafi says
APRIL 10, 2014 AT 5:16 AM
Pankaj,
I could only say along with a BIG THANKS is "GOD BLESS YOU!".
You are the Best.
– Rafi
Reply

Pankaj says
APRIL 10, 2014 AT 6:37 PM
Thanks Rafi, appreciate that you liked it.
Reply

Somesh says
APRIL 4, 2014 AT 8:58 AM
Can you please also provide explaination (with example) on semaphore and mutex in java ?
Reply

Somesh says
APRIL 4, 2014 AT 8:57 AM
Awesome article pankaj.. I' fan of ur writing..
Reply

Rad says
FEBRUARY 21, 2014 AT 7:57 AM
Good one!
Reply

Jitin says
FEBRUARY 10, 2014 AT 3:14 AM
Very well written article, Kudos for putting up so much effort. !
Good examples and links make it ons stop shop for re-visiting the grey area's of multi-threading
Reply

Sudhakar says
FEBRUARY 4, 2014 AT 10:51 PM
Hi Pankaj,
Nice explanation on concurrency and it helped me a lot to understand the new concurrency package introduced in java 5. I will certainly go through all of topics covered by you.
Reply

Jenny says
JANUARY 22, 2014 AT 4:44 PM
Your writing and examples are great. It is very easy to understand. Oracle Java should pay you to write their document I have many years java experience and a great developer. But My memory is not very good. So it is not easy for me to pass detail interview. Your writing can keep me read and help me to remember them. Thanks very much. I hope I can pass next interview.
Reply

Pankaj says
JANUARY 22, 2014 AT 6:11 PM
Wow, thats one of the best compliment I have ever got. Thanks for the kind words.
Wish you luck for your next interview.
Reply

puneetkagarwal says
JANUARY 21, 2014 AT 6:28 PM
Nice Stuff. Appreciate your efforts !!!!.
Keep on writing □
Reply

suresh says
DECEMBER 30, 2013 AT 11:08 AM
what is the difference between sleep() and wait() method and which scenario we should use sleep() and in which scenario we should use wait() method?
Reply

shubham says
JANUARY 4, 2014 AT 3:53 AM
sleep and wait, both methods put thread in hold/hung state. sleep method allows a thread to sleep(be in hold state) for a duration where as wait method allows a thread to wait(be in hold state) till other thread notify it to come out from sleep.
Reply

Maria Florencia Ortiz says
AUGUST 14, 2013 AT 4:24 PM
Pankaj,
I have a vaio s series sony 64 bit and I wonder if you can tell me what JAVA compiler to use and how I can install it.
Thanks
Maria
Reply

Pankaj says
AUGUST 14, 2013 AT 9:08 PM
Download JDK 64-bit installation file, follow instructions from https://www.journaldev.com/476/java-tutorial-1-setting-up-java-environment-on-windows
Reply

vivek says
JULY 3, 2013 AT 4:55 AM
Hi Pankaj,
Recently, in one of my interview i have been asked how can i initialize a variable before class initialization.I answered by making variable “static” but i am not sure. can you please help me in explaining about this ,
Also, when class is loaded it will be stored in perm gen area of heap isn’t it?
P.S. i surf internet alot but didnt get any satisfactory answer so i am asking you.
regards
vivek
Reply

Nick says

JULY 10, 2013 AT 4:24 AM

I believe final static variables must be initialized before class loaded into memory.

Reply

Yogendra says

JULY 31, 2013 AT 7:49 AM

You can mark the variables in a static block.. The idea of static blocks is that even before the class is loaded, the static block executes.

Reply

kamal says

SEPTEMBER 28, 2013 AT 8:29 AM

use static block to initialize variable bcz static block gets executed before loading the class into memory

Reply

Amay says

OCTOBER 9, 2013 AT 11:49 PM

As Nick said final static varialble is only ony way to intialize before any intialization. Because all other static block, static non final varible requires class to be loaded.

Reply

suresh says

JUNE 24, 2013 AT 10:38 AM

hiii,

It is really a nice explanation

Reply

Moorthy says

JANUARY 14, 2013 AT 3:46 PM

Hi Pankaj,

Good to see all the topics of muti-threading & concurrency in one page. However those topics should be

executed with atleast some samples to understand in a better way. Its easy to learn but very difficult to work in core java.
there are somebasics
http://www.nakov.com/inetjava/lectures/part-1-sockets/InetJava-1.3-Multithreading.html
Reply

Javin says
JANUARY 9, 2013 AT 2:47 PM
Threading is most confusing topic of Java and most asked topic also and confusion only increased with new concurrent package. thanks for your tips quite useful for beginners. I have also shared my experience as Top 10 multi-threading interview questions , do let me know how do you find it.
Reply

Pankaj says
JANUARY 9, 2013 AT 10:06 PM
Hi Javin, checked your questions and they are good and well written.
Reply

Leave a Reply

Your email address will not be published. Required fields are marked *
Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

CORE JAVA TUTORIAL

Java 10 Tutorials Java 9 Tutorials Java 8 Tutorials Java 7 Tutorials Core Java Basics OOPS Concepts Data Types and Operators String Manipulation Java Arrays Annotation and Enum Java Collections Java IO Operations Java Exception Handling MultiThreading and Concurrency Regular Expressions Advanced Java Concepts

IMPORTANT INTERVIEW QUESTIONS

Java Interview Questions

- Core Java Interview Questions
- String Interview Questions
- Multithreading Interview Questions
- Collections Interview Questions
- Exception Interview Questions
- Java Programming Interview Questions
- Java 8 Interview Questions Part 1
- Java 8 Interview Questions Part 2
- Servlet Interview Questions
- JSP Interview Questions
- Struts 2 Interview Questions
- JDBC Interview Questions
- Spring Interview Questions
- Hibernate Interview Questions
- JSF Interview Questions
- Web Services Interview Questions
- Scala Basic Interview Questions
- Scala Intermediate Interview Questions
- Scala Advanced Interview Questions
- Scala Interview Questions Summary
- Common Job Interview Questions

Miscellaneous

- Java ClassLoader
- String StringBuffer StringBuilder
- Java is Pass By Value
- Java Heap vs Stack Memory

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial

- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial