Java Threads Job Interview Questions And Answers

Interview Questions Answers

http://interviewquestionsanswers.org/

About Interview Questions Answers

Interview Questions Answers . ORG is an interview preparation guide of thousands of Job Interview Questions And Answers, Job Interviews are always stressful even for job seekers who have gone on countless interviews. The best way to reduce the stress is to be prepared for your job interview. Take the time to review the standard interview questions you will most likely be asked. These interview questions and answers on Java Threads will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts.

If you find any **question or answer** is incorrect or incomplete then you can **submit your question or answer** directly with out any registration or login at our website. You just need to visit Java Threads Interview Questions And Answers to add your answer click on the *Submit Your Answer* links on the website; with each question to post your answer, if you want to ask any question then you will have a link *Submit Your Question*; that's will add your question in Java Threads category. To ensure quality, each submission is checked by our team, before it becomes live. This Java Threads Interview preparation PDF was generated at **Saturday 4th February, 2017**

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter for latest Jobs and interview preparation guides.
http://twitter.com/InterviewQA

If you need any further assistance or have queries regarding this document or its material or any of other inquiry, please do not hesitate to contact us.
http://interviewquestionsanswers.org/Contact-Us

Best Of Luck.

Interview Questions Answers.ORG Team
http://InterviewQuestionsAnswers.ORG/
Support@InterviewQuestionsAnswers.ORG

Java Threads Interview Questions And Answers Guide.

Question - 1:
What is a working thread?

Ans:
A working thread, more commonly known as a worker thread is the key part of a design pattern that allocates one thread to execute one task. When the task is complete, the thread may return to a thread pool for later use. In this scheme a thread may execute arbitrary tasks, which are passed in the form of a Runnable method argument, typically execute(Runnable). The runnable tasks are usually stored in a queue until a thread host is available to run them.
The worker thread design pattern is usually used to handle many concurrent tasks where it is not important which finishes first and no single task needs to be coordinated with another. The task queue controls how many threads run concurrently to improve the overall performance of the system. However, a worker thread framework requires relatively complex programming to set up, so should not be used where simpler threading techniques can achieve similar results.

View All Answers

Question - 2:
What is a green thread?

Ans:
A green thread refers to a mode of operation for the Java Virtual Machine (JVM) in which all code is executed in a single operating system thread. If the Java program has any concurrent threads, the JVM manages multi-threading internally rather than using other operating system threads.
There is a significant processing overhead for the JVM to keep track of thread states and swap between them, so green thread mode has been deprecated and removed from more recent Java implementations. Current JVM implementations make more efficient use of native operating system threads.

View All Answers

Question - 3:
Which class is the wait() method defined in?

Ans:
The wait() method is defined in the Object class, which is the ultimate superclass of all others. So the Thread class and any Runnable implementation inherit this method from Object. The wait() method is normally called on an object in a multi-threaded program to allow other threads to run. The method should should only be called by a thread that has ownership of the object's monitor, which usually means it is in a synchronized method or statement block.

View All Answers

Question - 4:
What is the volatile modifier for?

Ans:
The volatile modifier is used to identify variables whose values should not be optimized by the Java Virtual Machine, by caching the value for example. The volatile modifier is typically used for variables that may be accessed or modified by numerous independent threads and signifies that the value may change without synchronization.

View All Answers

Question - 5:
What is the SwingUtilities.invokeLater(Runnable) method for?

Ans:
The static utility method invokeLater(Runnable) is intended to execute a new runnable thread from a Swing application without disturbing the normal sequence of event dispatching from the Graphical User Interface (GUI). The method places the runnable object in the queue of Abstract Windowing Toolkit (AWT) events that are due to be processed and returns immediately. The runnable object's run() method is only called when it reaches the front of the queue.
The deferred effect of the invokeLater(Runnable) method ensures that any necessary updates to the user interface can occur immediately, and the runnable work will begin as soon as those high priority events are dealt with. The invoke later method might be used to start work in response to a button click that also requires a significant change to the user interface, perhaps to restrict other activities, while the runnable thread executes.

View All Answers

Question - 6:

Is there a separate stack for each thread in Java?

Ans:

Yes. Every thread maintains its own separate stack, called Runtime Stack but they share the same memory. Elements of the stack are the method invocations, called activation records or stack frame. The activation record contains pertinent information about a method like local variables.

View All Answers

Question - 7:

How would you implement a thread pool?

Ans:

public class ThreadPool implements ThreadPoolInt
This class is an generic implementation of a thread pool, which takes the following input
a) Size of the pool to be constructed
b) Name of the class which implements Runnable and constructs a thread pool with active threads that are waiting for activation. Once the threads have finished processing they come back and wait once again in the pool.
This thread pool engine can be locked i.e. if some internal operation is performed on the pool then it is preferable that the thread engine be locked. Locking ensures that no new threads are issued by the engine. However, the currently executing threads are allowed to continue till they come back to the passivePool.

View All Answers

Question - 8:

What state does a thread enter when it terminates its processing?

Ans:

When a thread terminates its processing, it enters the dead state.

View All Answers

Question - 9:

What is an objects lock and which objects have locks?

Ans:

An objects lock is a mechanism that is used by multiple threads to obtain synchronized access to the object. A thread may execute a synchronized method of an object only after it has acquired the objects lock. All objects and classes have locks. A class's lock is acquired on the class's Class object.

View All Answers

Question - 10:

Why would you use a synchronized block vs. synchronized method?

Ans:

Synchronized blocks place locks for shorter periods than synchronized methods.

View All Answers

Question - 11:

When you will synchronize a piece of your code?

Ans:

When you expect that your shared code will be accessed by different threads and these threads may change a particular data causing data corruption, then they are placed in a synchronized construct or a synchronized method.

View All Answers

Question - 12:

What do you understand by Synchronization?
Or
What is synchronization and why is it important?
Or
Describe synchronization in respect to multithreading?
Or
What is synchronization?

Ans:

With respect to multithreading, Synchronization is a process of controlling the access of shared resources by the multiple threads in such a manner that only one thread can access a particular resource at a time. In non synchronized multithreaded application, it is possible for one thread to modify a shared object while another thread is in the process of using or updating the object's value. Synchronization prevents such type of data corruption which may otherwise lead to dirty reads and significant errors.
E.g. synchronizing a function:
public synchronized void Method1 () {
// method code.
}
E.g. synchronizing a block of code inside a function:
public Method2 (){
synchronized (this) {
// synchronized code here.
}
}

View All Answers

Question - 13:

What is daemon thread and which method is used to create the daemon thread?

Ans:

Daemon threads are threads with low priority and runs in the back ground doing the garbage collection operation for the java runtime system. The setDaemon() method is used to create a daemon thread. These threads run without the intervention of the user. To determine if a thread is a daemon thread, use the accessor method isDaemon()
When a standalone application is run then as long as any user threads are active the JVM cannot terminate, otherwise the JVM terminates along with any daemon threads which might be active. Thus a daemon thread is at the mercy of the runtime system. Daemon threads exist only to serve user threads.

View All Answers

Question - 14:

What is the difference between process and thread?

Ans:

A thread is a separate path of execution in a program. A Process is a program in execution.

View All Answers

Question - 15:

What is the difference between the methods sleep() and wait()?

Ans:

The sleep method is used when the thread has to be put aside for a fixed amount of time. Ex: sleep(1000), puts the thread aside for exactly one second. The wait method is used to put the thread aside for up to the specified time. It could wait for much lesser time if it receives a notify() or notifyAll() call. Ex: wait(1000), causes a wait of up to one second. The method wait() is defined in the Object and the method sleep() is defined in the class Thread.

View All Answers

Question - 16:

Can Java object be locked down for exclusive use by a given thread?
Or
What happens when a thread cannot acquire a lock on an object?

Ans:

Yes. You can lock an object by putting it in a "synchronized" block. The locked object is inaccessible to any thread other than the one that explicitly claimed it. If a thread attempts to execute a synchronized method or synchronized statement and is unable to acquire an object's lock, it enters the waiting state until the lock becomes available.

View All Answers

Question - 17:

What are synchronized methods and synchronized statements?

Ans:

Synchronized methods are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class. Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

View All Answers

Question - 18:

How does multithreading take place on a computer with a single CPU?

Ans:

The operating system's task scheduler allocates execution time to multiple tasks. By quickly switching between executing tasks, it creates the impression that tasks execute sequentially.

View All Answers

Question - 19:

What is deadlock?

Ans:

When two threads are waiting for each other and can't proceed until the first thread obtains a lock on the other thread or vice versa, the program is said to be in a deadlock.

View All Answers

Question - 20:

What is thread? What are the high-level thread states?
Or
What are the states associated in the thread?

Ans:

A thread is an independent path of execution in a system. The high-level thread states are ready, running, waiting and dead.

View All Answers

Question - 21:
What is the purpose of the wait(), notify(), and notifyAll() methods?

Ans:
The wait(), notify() and notifyAll() methods are used to provide an efficient way for thread inter-communication.
View All Answers

Question - 22:
What invokes a threads run() method?

Ans:
After a thread is started, via its start() method of the Thread class, the JVM invokes the thread's run() method when the thread is initially executed.
View All Answers

Question - 23:
What is the difference between preemptive scheduling and time slicing?

Ans:
Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then re-enters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.
View All Answers

Question - 24:
What is mutual exclusion? How can you take care of mutual exclusion using Java threads?

Ans:
Mutual exclusion is a phenomenon where no two processes can access critical regions of memory at the same time. Using Java multithreading we can arrive at mutual exclusion. For mutual exclusion, you can simply use the synchronized keyword and explicitly or implicitly provide an Object, any Object, to synchronize on. The synchronized keyword can be applied to a class, to a method, or to a block of code. There are several methods in Java used for communicating mutually exclusive threads such as wait( ), notify( ), or notifyAll( ). For example, the notifyAll( ) method wakes up all threads that are in the wait list of an object.
View All Answers

Question - 25:
How to create multithreaded program? Explain different ways of using thread? When a thread is created and started, what is its initial state?
Or
Extending Thread class or implementing Runnable Interface. Which is better?

Ans:
You have two ways to do so. First, making your class "extends" Thread class. The other way is making your class implement "Runnable" interface. The latter is more advantageous, cause when you are going for multiple inheritance, then only interface can help. . If you are already inheriting a different class, then you have to go for Runnable Interface. Otherwise you can extend Thread class. Also, if you are implementing interface, it means you have to implement all methods in the interface. Both Thread class and Runnable interface are provided for convenience and use them as per the requirement. But if you are not extending any class, better extend Thread class as it will save few lines of coding. Otherwise performance wise, there is no distinguishable difference. A thread is in the ready state after it has been created and started.
View All Answers

Question - 26:
What is the difference between yielding and sleeping?

Ans:
When a task invokes its yield() method, it returns to the ready state, either from waiting, running or after its creation. When a task invokes its sleep() method, it returns to the waiting state from a running state.
View All Answers

Question - 27:
What are three ways in which a thread can enter the waiting state?
Or
What are different ways in which a thread can enter the waiting state?

Ans:
A thread can enter the waiting state by the following ways:
1. Invoking its sleep() method,
2. By blocking on I/O
3. By unsuccessfully attempting to acquire an object's lock
4. By invoking an object's wait() method.
5. It can also enter the waiting state by invoking its (deprecated) suspend() method.
View All Answers

Question - 28:

How do I create a Runnable with inheritance?

Ans:

To introduce a Runnable type to an existing class hierarchy, you need to create a sub-class that declares that it implements the Runnable interface, and provide a run method to fulfil the interface. This combination of interface and inheritance means that runnable implementations can be very minor extensions of existing classes

View All Answers

Question - 29:

Do I need to use synchronized on setValue(int)?

Ans:

It depends whether the method affects method local variables, class static or instance variables. If only method local variables are changed, the value is said to be confined by the method and is not prone to threading issues.

View All Answers

Question - 30:

Can I implement my own start() method?

Ans:

The Thread start() method is not marked final, but should not be overridden. This method contains the code that creates a new executable thread and is very specialised. Your threaded application should either pass a Runnable type to a new Thread, or extend Thread and override the run() method.

View All Answers

Question - 31:

What is the difference between a threads start() and run() methods?

Ans:

The separate start() and run() methods in the Thread class provide two ways to create threaded programs. The start() method starts the execution of the new thread and calls the run() method. The start() method returns immediately and the new thread normally continues until the run() method returns.
The Thread class' run() method does nothing, so sub-classes should override the method with code to execute in the second thread. If a Thread is instantiated with a Runnable argument, the thread's run() method executes the run() method of the Runnable object in the new thread instead.
Depending on the nature of your threaded program, calling the Thread run() method directly can give the same output as calling via the start() method, but in the latter case the code is actually executed in a new thread.

View All Answers

Question - 32:

Why not override Thread to make a Runnable?

Ans:

There is little difference in the work required to override the Thread class compared with implementing the Runnable interface, both require the body of the run() method. However, it is much simpler to make an existing class hierarchy runnable because any class can be adapted to implement the run() method. A subclass of Thread cannot extend any other type, so application-specific code would have to be added to it rather than inherited.
Separating the Thread class from the Runnable implementation also avoids potential synchronization problems between the thread and the run() method. A separate Runnable generally gives greater flexibility in the way that runnable code is referenced and executed.

View All Answers

Question - 33:

A Thread is runnable, how does that work?

Ans:

The Thread class' run method normally invokes the run method of the Runnable type it is passed in its constructor. However, it is possible to override the thread's run method with your own.

View All Answers

Question - 34:

How does the run() method in Runnable work?

Ans:

It may help to think of the run method like the main method in standard single threaded applications. The run method is a standard entry point to run or execute a class. The run method is normally only executed in the context of an independent Thread, but is a normal method in all other respects.

View All Answers

Question - 35:

What is the difference between Thread and Runnable types?

Ans:

A Java Thread controls the main path of execution in an application. When you invoke the Java Virtual Machine with the java command, it creates an implicit thread in which to execute the main method. The Thread class provides a mechanism for the first thread to start-up other threads to run in parallel with it.

View All Answers

Question - 36:

Why are there separate wait and sleep methods?

Ans:

The static Thread.sleep(long) method maintains control of thread execution but delays the next action until the sleep time expires. The wait method gives up control over thread execution indefinitely so that other threads can run.

View All Answers

Question - 37:

Why are wait(), notify() and notifyall() methods defined in the Object class?

Ans:

These methods are detailed on the Java Software Development Kit JavaDoc page for the Object class, they are to implement threaded programming for all subclasses of Object.

View All Answers

Question - 38:

What is threaded programming and when is it used?

Ans:

Threaded programming is normally used when a program is required to do more than one task at the same time. Threading is often used in applications with graphical user interfaces; a new thread may be created to do some processor-intensive work while the main thread keeps the interface responsive to human interaction.
The Java programming language has threaded programming facilities built in, so it is relatively easy to create threaded programs. However, multi-threaded programs introduce a degree of complexity that is not justified for most simple command line applications.

View All Answers

Java Programing Most Popular & Related Interview Guides