Java Multi-Threading Interview Questions

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.

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.

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.

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.

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.
[sociallocker id="2713"]

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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());
    }

});
```

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.

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.

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.

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.

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.
TestThread.java

```
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
```