FAQs on Multithreading

1. Describe synchronization in respect to multithreading.

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources.

Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

2. Explain different way of using thread?

The thread could be implemented by using runable interface or by inheriting from the thread class. The former is more advantageous cause when you are going for multiple inheritances, the only interface can help.

3. What are the daemon threads and which method is used to create them?

The Daemon threads are the low priority threads that run intermittently in the background, performing the garbage collection operation for the java runtime system. “setDaemon” method is used to create a daemon thread.

4. Can applets communicate with each other?

Overridden At this point in time applets may communicate with other applets running in the same virtual machine. If the applets are of the same class, they can communicate via shared static variables. If the applets are of different classes, then each will need a reference to the same class with static variables. In any case the basic idea is to pass the information back and forth through a static variable.

An applet can also get references to all other applets on the same page using the getApplets() method of java.applet.AppletContext. Once you get the reference to an applet, you can communicate with it by using its public members.

It is conceivable to have applets in different virtual machines that talk to a server somewhere on the Internet and store any data that needs to be serialized there. Then, when another applet needs this data, it could connect to this same server. This implementation is non-trivial.

5. What method must be implemented by all threads?

All tasks must implement the run() method, whether they are a subclass of Thread or implement the Runnable interface.

6. What are synchronized methods and synchronized statements?

Overriding 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.

7. What is the difference between pre-emptive scheduling and time slicing?

Under pre-emptive 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.

8. What is synchronization and why is it important?

Synchronization is the mechanism that ensures that only one thread is accessed the resources at a time.

Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

9. Can a lock be acquired on a class?

Yes, a lock can be acquired on a class. This lock is acquired on the class's Class object.

10. What's new with the stop(), suspend() and resume() methods in JDK 1.2?

The stop(), suspend() and resume() methods have been deprecated in JDK 1.2.

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

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

12. What is the difference between yielding and sleeping?

When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep()method, it returns to the waiting state.

13. What is the difference between preemptive scheduling and time slicing?

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 reenters the pool of ready tasks.

The scheduler then determines which task should execute next, based on priority and other factors.

14. What is the catch or declare rule for method declarations?

If a checked exception may be thrown within the body of a method, the method must either catch the exception or declare it in its throws clause.

15. What is a task's priority and how is it used in scheduling?

A task's priority is an integer value that identifies the relative order in which it should be executed with respect to other tasks. The scheduler attempts to schedule higher priority tasks before lower priority tasks.

16. When a thread is created and started, what is its initial state?

A thread is in the ready state after it has been created and started.

17. What invokes a thread's run() method?

After a thread is started, via its start() method or that of the Thread class, the JVM invokes the thread'srun() method when the thread is initially executed.

18. What is the purpose of the wait(), notify(), and notifyAll() methods?

The wait(), notify(), and notifyAll() methods are used to provide an efficient way for threads to wait for a shared resource. When a thread executes an object's wait() method, it enters the waiting state. It only enters the ready state after another thread invokes the object's notify() or notifyAll() methods.

19. What are the high-level thread states?

The high-level thread states are ready, running, waiting, and dead.

20. What is an object's lock and which objects have locks?

An object's 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 object's lock.
All objects and classes have locks. A class's lock is acquired on the class's Class object.

21. What happens when a thread cannot acquire a lock on an object?

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.

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

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.

23. When is the finally clause of a try-catch-finally statement executed?

The finally clause of the try-catch-finally statement is always executed unless the thread of execution terminates or an exception occurs within the execution of the finally clause.

24. What happens when you invoke a thread's interrupt method while it is sleeping or waiting?

When a task's interrupt() method is executed, the task enters the ready state. The next time the task enters the running state, an Interrupted Exception is thrown.

25. What are three ways in which a thread can enter the waiting state?

A thread can enter the waiting state by invoking its sleep() method, by blocking on I/O, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's wait() method. It can also enter the waiting state by invoking its (deprecated) suspend() method.

26. What happens if a try-catch-finally statement does not have a catch clause to handle an exception that is thrown within the body of the try statement?

The exception propagates up to the next higher level try-catch statement (if any) or results in the program's termination.

27. What method must be implemented by all threads?

All tasks must implement the run() method, whether they are a subclass of Thread or implement theRunnable interface.

28. What are synchronized methods and synchronized statements?

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.

29. What are the two basic ways in which classes that can be run as threads may be defined?

A thread class may be declared as a subclass of Thread, or it may implement the Runnable interface.

30. What is the difference between process and thread?

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

31. What is multithreading and what are the methods for inter-thread communication and what is the class in which these methods are defined?

Multithreading is the mechanism in which more than one thread run independent of each other within the process. wait (), notify () and notifyAll() methods can be used for inter-thread communication and these methods are in Object class.

wait(): When a thread executes a call to wait() method, it surrenders the object lock and enters into a waiting state.

notify() or notifyAll(): To remove a thread from the waiting state, some other thread must make a call to notify() or notifyAll() method on the same object.

32. What is the class and interface in java to create thread and which is the most advantageous method?

Thread class and Runnable interface can be used to create threads and using Runnable interface is the most advantageous method to create threads because we need not extend thread class here.

33. What are the states associated in the thread?

Thread contains ready, running, waiting and dead states.

34. When you will synchronize a piece of your code?

When you expect your code will be accessed by different threads and these threads may change a particular data causing data corruption.

35. What is deadlock?

When two threads are waiting each other and can't precede the program is said to be deadlock.