1. You have thread T1, T2, and T3, how will you ensure that thread T2 run after T1 and thread T3 run after T2?

Answer: it can be achieved by using the join method of Thread class.

1. What are differences between wait and sleep method in Java?

Answer: The only major difference is that wait releases the lock or monitor while sleep doesn't release any lock or monitor while waiting. The wait is used for inter-thread communication while sleep is used to introduce pause on execution.

1. Describe the different states of a Thread and when do the state transitions occur.

answer: The state of a *Thread* can be checked using the *Thread.getState()* method. Different states of a *Thread* are described in the *Thread.State* enum. They are:

* ***NEW*** — a new *Thread* instance that was not yet started via *Thread.start()*
* ***RUNNABLE*** — a running thread. It is called runnable because at any given time it could be either running or waiting for the next quantum of time from the thread scheduler. A*NEW*thread enters the *RUNNABLE* state when you call *Thread.start()* on it
* ***BLOCKED*** — a running thread becomes blocked if it needs to enter a synchronized section but cannot do that due to another thread holding the monitor of this section
* ***WAITING*** — a thread enters this state if it waits for another thread to perform a particular action. For instance, a thread enters this state upon calling the *Object.wait()* method on a monitor it holds, or the *Thread.join()* method on another thread
* ***TIMED\_WAITING*** — same as the above, but a thread enters this state after calling timed versions of *Thread.sleep()*, *Object.wait()*, *Thread.join()* and some other methods
* ***TERMINATED*** — a thread has completed the execution of its *Runnable.run()* method and terminated

**Q6. What is the Thread’s interrupt flag? How can you set and check it? How does it relate to the InterruptedException?**

The interrupt flag, or interrupt status, is an internal *Thread* flag that is set when the thread is interrupted. To set it, simply call *thread.interrupt()* on the thread object*.*

If a thread is currently inside one of the methods that throw *InterruptedException* (*wait*, *join*, *sleep* etc.), then this method immediately throws InterruptedException. The thread is free to process this exception according to its own logic.

If a thread is not inside such method and *thread.interrupt()* is called, nothing special happens. It is thread’s responsibility to periodically check the interrupt status using *static Thread.interrupted()* or instance *isInterrupted()* method. The difference between these methods is that the *static Thread.interrupt()* clears the interrupt flag, while *isInterrupted()* does not.

**Q14. If two threads call a synchronized method on different object instances simultaneously, could one of these threads block? What if the method is static?**

If the method is an instance method, then the instance acts as a monitor for the method. Two threads calling the method on different instances acquire different monitors, so none of them gets blocked.

If the method is *static*, then the monitor is the *Class* object. For both threads, the monitor is the same, so one of them will probably block and wait for another to exit the *synchronized* method.