Java Thread Lifecycle - Short Notes

1. Why Two Ways to Create Threads?

In Java, threads can be created in two ways:
1. By extending the Thread class
2. By implementing the Runnable interface

Reason:
- A class can extend only ONE class (no multiple inheritance).
- A class can implement MULTIPLE interfaces.

So, implementing Runnable is preferred as it's more flexible and separates task from thread mechanism.

2. Thread Lifecycle States

A thread goes through multiple states during its lifetime:

1. New - Thread is created but not started yet.
2. Runnable - Ready to run, waiting for CPU.
3. Running - Actively executing the run() method.
4. Blocked - Waiting to acquire a lock.
5. Waiting - Waiting indefinitely for another thread's action.
6. Timed Waiting - Waiting for a specified time.
7. Terminated - Execution of run() is complete.

3. State Transitions

- New -> Runnable: start()
- Runnable -> Running: Scheduler assigns CPU
- Running -> Runnable: yield()
- Running -> Waiting: wait(), join() (no timeout)
- Running -> Timed Waiting: sleep(time), wait(time), join(time)
- Running -> Blocked: Waiting for a lock
- Blocked/Waiting/Timed Waiting -> Runnable: Notified / time expired / lock acquired
- Running -> Terminated: run() completes

4. Summary Table

State	Description
New	Created but not started
Runnable	Waiting for CPU
Running	Executing run()
Blocked	Waiting to acquire lock

Java Thread Lifecycle - Short Notes

Waiting	Waiting indefinitely
Timed Waiting	Waiting with timeout
Terminated	Execution finished