**Thread Lifecycle in Java**

A Java thread goes through the following **five major states**:

markdown

CopyEdit

1. New

2. Runnable

3. Running

4. Blocked / Waiting / Timed Waiting

5. Terminated (Dead)

These are based on the Thread.State enum introduced in Java 5.

**🔁 1. New State**

* A thread is in **New** state when it is **created** using the Thread class but **not yet started**.
* At this point, it’s just an object in memory.

Thread t = new Thread(); // New state

**🏃‍♂️ 2. Runnable State**

* When start() is called on the thread, it moves to the **Runnable** state.
* It means the thread is **ready to run**, and waiting for **CPU allocation** by the **Thread Scheduler**.
* It may or may not run immediately, depending on the OS scheduler.

t.start(); // Thread is now Runnable

**⚙️ 3. Running State**

* The thread enters the **Running** state when the **Thread Scheduler picks it** from the runnable pool.
* It starts executing the code inside the run() method.

public void run() {

System.out.println("Thread is running...");

}

**⏸ 4. Non-Runnable States**

This includes 3 sub-states: Blocked, Waiting, and Timed Waiting.

**a. Blocked**

* A thread goes into **Blocked** state when it tries to access a **synchronized block/method** which is currently held by another thread.
* It waits to **acquire the monitor lock**.

synchronized(obj) {

// Only one thread can execute here at a time

}

**b. Waiting**

* A thread is in **Waiting** state if it waits **indefinitely** for another thread to perform a specific action.

Example:

t.join(); // Main thread waits until 't' finishes

**c. Timed Waiting**

* A thread is in **Timed Waiting** if it waits for a specific time before going back to runnable.

Examples:

Thread.sleep(1000);

t.join(500);

wait(1000);

**☠️ 5. Terminated (Dead) State**

* A thread is in **Terminated** or **Dead** state when:
  + run() method completes,
  + or an **uncaught exception** terminates it.

