🔹 Process Lifecycle (in OS)

A process = independent program under execution (with its own memory, PCB, resources).

Typical process states:

1. **New** – Process is being created.
2. **Ready** – Waiting to be assigned to CPU.
3. **Running** – Instructions are being executed.
4. **Waiting/Blocked** – Waiting for I/O or event.
5. **Terminated/Exit** – Execution finished.

📌 In some OS diagrams, you may also see:

- **Suspended Ready / Suspended Blocked** (swapped to disk).

🔹 Thread Lifecycle (in Java/OS)

A thread = lightweight unit of a process (shares memory/resources with parent process).

Typical thread states:

1. **New (Created)** – Thread object created, not started yet.
2. **Runnable** – Ready to run, waiting for CPU (like process "Ready").
3. **Running** – CPU executing thread code.
4. **Waiting** – Waiting indefinitely until notified (via `notify()`).
5. **Timed Waiting** – Waiting for a fixed time (`sleep()`, `join(timeout)`).
6. **Terminated/Dead** – Execution finished.

🔹 Key Differences

Aspect	Process Lifecycle	Thread Lifecycle
Unit	Program in execution	Smallest execution unit inside process
States	New, Ready, Running, Waiting, Terminated (+Suspended)	New, Runnable, Running, Waiting, Timed Waiting, Terminated
Memory	Each process has separate memory (PCB, address space)	Threads share same memory of process
Overhead	Heavy (context switch = save PCB, memory maps)	Light (context switch only saves registers/stack)
Creation	Slower (involves OS, PCB allocation)	Faster (within same process, no separate PCB)