JAVA-ASSIGNMENT – 2

Explain Thread life cycle in java

In Java, the lifecycle of a thread goes through several distinct stages or states, controlled by methods that allow you to start, pause, resume, or stop the thread. Here's a breakdown of each state in the thread lifecycle and the primary methods associated with each stage:

1. New (or Born) State :

- **State Description**: When a thread is created using the `Thread` class or implementing the `Runnable` interface, it is in the "New" state. At this point, it's not yet started.
- **Key Method**: `Thread(Thread t, Runnable r)` (constructor)
- **Transition**: Calling `start()` on the thread will move it from the "New" state to the "Runnable" state.

2. Runnble State :

- **State Description**: After calling start(), the thread enters the "Runnable" state, meaning it's ready to run but is waiting for CPU time. The thread scheduler decides when the thread will actually run based on its priority and other factors.
- **Key Method**: start()
- **Transition**: When the thread scheduler picks this thread for execution, it transitions to the "Running" state.

3. Running State :

State Description: In the "Running" state, the thread is actively executing its task. The run() method is the entry point where the thread's job is defined.

- **Key Method**: run()
- **Transition**:
    - **To Blocked/Waiting State**: The thread may enter a waiting or blocked state if it needs resources, if it's waiting for synchronization, or if it explicitly calls wait/sleep/join.

- **To Terminated State**: If the run() method completes, the thread exits the "Running" state and moves to the "Terminated" state.

4. Blocked/ Waiting State :

- **State Description**: Threads enter this state when they are waiting for some resource to become available or if they call certain methods to pause themselves (e.g., sleep, join, wait).
- **Key Methods**:
  - **sleep(long millis)**: Puts the thread to sleep for a specified duration.
  - **wait() and wait(long timeout)**: Makes the thread wait for another thread to call notify() or notifyAll().
  - **join()**: Allows one thread to wait until another completes.
- **Transition**:
  - **To Runnable**: A thread moves back to "Runnable" if the waiting or blocking condition is resolved, e.g., the wait/sleep/join duration ends or it's notified.

5. Timed Waiting :

- **State Description**: A special state where the thread is in waiting, but with a specified time duration. Methods like sleep(long millis) and join(long millis) put a thread in this state.
- **Key Methods**:
  - **sleep(long millis)**
  - **join(long millis)**
  - **wait(long timeout)**
- **Transition**: After the specified waiting time elapses, the thread returns to the Runnable state.

6. Terminated (or Dead) State :

- **State Description**: Once the run() method completes execution, the thread enters the "Terminated" state, meaning it has finished its lifecycle and cannot be restarted.
- **Key Method**: No method can bring it back to life. Once terminated, the thread object can only be discarded.
- **Transition**: A thread is moved to the terminated state after its run() method completes, or if it encounters an unhandled exception.

Summary of Key Methods:

- **start()**: Starts the thread.
- **run()**: Defines the task performed by the thread.
- **sleep(long millis)**: Puts the thread to sleep for a specific period.
- **join()**: Makes one thread wait for another to finish.
- **wait() and notify()/notifyAll()**: Used for inter-thread communication.