Java Concurrency & Multithreading
Industry-Level + Interview Roadmap

This document covers practical, production-grade Java concurrency topics along with interview-focused concepts commonly expected by product and service-based companies.

1. Threading Fundamentals

- Process vs Thread
- Thread lifecycle and states
- Thread class vs Runnable interface
- start() vs run()
- sleep(), join(), yield(), interrupt()
- Thread priority and daemon threads

2. Synchronization & Intrinsic Locks

- synchronized keyword (method and block)
- Object lock vs Class lock
- Reentrancy in Java
- wait(), notify(), notifyAll()
- Producer–Consumer problem

3. Java Memory Model (JMM)

- Visibility, atomicity, ordering
- Happens-before relationship
- volatile keyword
- volatile vs synchronized
- Double-checked locking problem

4. Lock Framework (java.util.concurrent.locks)

- Lock interface
- ReentrantLock features
- Fair vs Non-fair locking
- tryLock() and lockInterruptibly()
- ReadWriteLock and ReentrantReadWriteLock

5. Executor Framework & Thread Pools

- Executor and ExecutorService

- Fixed, Cached, Single, Scheduled thread pools
- ThreadPoolExecutor internal working
- Core pool size, max pool size, queue
- Rejection policies and tuning strategies

6. Callable, Future & CompletableFuture

- Callable vs Runnable
- Future blocking behavior
- CompletableFuture async programming model
- thenApply, thenCompose, thenCombine
- Exception handling in CompletableFuture

7. Concurrent Collections

- Why legacy collections are slow
- ConcurrentHashMap internal working
- CopyOnWriteArrayList and CopyOnWriteArraySet
- BlockingQueue implementations
- ConcurrentModificationException avoidance

8. Atomic Variables & CAS

- AtomicInteger, AtomicLong, AtomicReference
- Compare-And-Swap (CAS) mechanism
- Lock-free programming basics
- ABA problem

9. Fork/Join Framework & Parallelism

- ForkJoinPool architecture
- RecursiveTask and RecursiveAction
- Work-stealing algorithm
- Parallel streams internals
- When not to use parallel streams

10. Deadlocks, Livelocks & Starvation

- Deadlock conditions and examples
- Deadlock prevention techniques
- Livelock vs Deadlock

- Thread starvation
- Debugging using jstack

11. Performance & Best Practices

- Minimize synchronization scope
- Prefer immutability
- CPU-bound vs IO-bound thread pool sizing
- Avoid blocking operations
- Context switching overhead

12. Interview-Oriented Coding Scenarios

- Producer–Consumer implementation
- Thread-safe Singleton
- Custom ThreadPoolExecutor
- Rate limiter
- Parallel API calls using CompletableFuture
- Thread-safe cache design