Atomic Variables & Non-blocking Synchronization

Why Not Just Use Locks?

- Lock contention causes OS scheduling and context switches.
- Suspending threads leads to delays and priority inversion.
- `volatile` gives visibility but not atomic compound actions.

Hardware Support

- CPUs expose atomic read-modify-write primitives.
- Compare and Swap (CAS): Compare expected vs actual value; update if matching.

Atomic Classes Overview

- `AtomicInteger`, `AtomicLong`, `AtomicBoolean`, `AtomicReference`
- Atomic arrays and field updaters

Locks vs Atomics Performance

- Low contention: atomics scale better
- High contention: locks suspend threads

Nonblocking Algorithms

- An algorithm is called nonblocking if failure or suspension of any thread cannot cause failure or suspension of another thread.
- An algorithm is called lock-free if, at each step, some thread can make progress.
- Nonblocking algorithms are immune to deadlock or priority inversion.

Summary

- Atomics expose hardware CAS from Java.
- Atomics scale better under typical contention.
- Non-blocking avoids deadlocks.