Java Multithreading Cheat Sheet

1. Basic Threading

- Create Thread: class MyThread extends Thread or implements Runnable
- Start Thread: thread.start()
- Pause Thread: Thread.sleep(ms)
- Thread Name: Thread.currentThread().getName()

2. Synchronization

- Mutual Exclusion: synchronized methods or blocks
- Advanced Locking: ReentrantLock (manual lock/unlock)
- Atomic Operations: AtomicInteger, AtomicBoolean, etc.
- Volatile: ensures visibility between threads (no caching)

3. Inter-thread Communication

- wait(): Pauses thread and releases lock (in synchronized block)
- notify(): Wakes one waiting thread (in synchronized block)
- notifyAll(): Wakes all waiting threads
- Always use wait() in a while loop to avoid spurious wakeups

4. ExecutorService & Thread Pools

- Create Thread Pool: Executors.newFixedThreadPool(n)
- Submit Task: executor.submit(task)
- Callable vs Runnable: Callable returns value & can throw exceptions
- Future<V>: Holds result of Callable, future.get() waits for result
- Shutdown: executor.shutdown() to free resources

5. Runnable vs Callable

- Runnable: run() method, no return value, no checked exceptions
- Callable<T>: call() method, returns value, can throw checked exceptions

6. Thread-Safe Collections

- ConcurrentHashMap

- CopyOnWriteArrayList
- BlockingQueue
- AtomicInteger, AtomicBoolean

7. Common Interview Q&A

- Race Condition: Multiple threads accessing shared data simultaneously without sync
- Difference between wait() and sleep(): wait() releases lock, sleep() doesn't
- Difference between synchronized block and method: Block gives finer control
- Difference between notify() and notifyAll(): notify() wakes one, notifyAll() wakes all
- Difference between Callable and Runnable: Callable returns a result

Java Multithreading Recap & Cheat Sheet

Java Multithreading Final Quiz - 10 Questions

1. What is the purpose of the synchronized keyword in Java?
   Answer: C) To prevent concurrent access to critical sections

2. Which of the following is thread-safe?
   Answer: B) AtomicInteger

3. What is the main difference between wait() and sleep()?
   Answer: B) wait() releases lock; sleep() doesnt

4. Which interface allows a thread to return a value?
   Answer: B) Callable

5. What does ExecutorService.shutdown() do?
   Answer: C) Stops accepting new tasks, lets running tasks complete

6. Which collection is thread-safe by default?
   Answer: B) ConcurrentHashMap

7. Can notify() be called without a synchronized block?
   Answer: B) No

8. What does future.get() do?
   Answer: C) Blocks until result is available

9. What is a daemon thread?
   Answer: B) A system background thread that ends when main threads finish

10. Which of the following allows timed locking?
    Answer: B) ReentrantLock.tryLock()

Java Multithreading Cheat Sheet (for Interviews & Practice)

Basic Threading
- Create Thread: class MyThread extends Thread or implements Runnable
- Start Thread: thread.start()
- Pause Thread: Thread.sleep(ms)
- Thread Name: Thread.currentThread().getName()

Synchronization
- Mutual Exclusion: synchronized methods or blocks
- Advanced Locking: ReentrantLock (manual lock/unlock)
- Atomic Operations: AtomicInteger, AtomicBoolean, etc.
- Volatile: ensures visibility between threads (no caching)

Inter-thread Communication
- wait(): Pauses thread and releases lock (in synchronized block)
- notify(): Wakes one waiting thread (in synchronized block)
- notifyAll(): Wakes all waiting threads
- Always use wait() in a while loop to avoid spurious wakeups

ExecutorService & Thread Pools
- Create Thread Pool: Executors.newFixedThreadPool(n)
- Submit Task: executor.submit(task)
- Callable vs Runnable: Callable returns value & can throw exceptions
- Future<V>: Holds result of Callable, future.get() waits for result
- Shutdown: executor.shutdown() to free resources

Runnable vs Callable
- Runnable: run() method, no return value, no checked exceptions
- Callable<T>: call() method, returns value, can throw checked exceptions

Thread-Safe Collections
- ConcurrentHashMap
- CopyOnWriteArrayList
- BlockingQueue
- AtomicInteger, AtomicBoolean

Common Interview Q&A
- What is a race condition? When multiple threads access shared data simultaneously without proper synchronization.
- Difference between wait() and sleep()? wait() releases lock; sleep() doesnt.
- Difference between synchronized block and method? Block gives finer control, method locks the whole method.
- Difference between notify() and notifyAll()? notify() wakes one thread; notifyAll() wakes all waiting threads.
- Difference between Callable and Runnable? Callable returns a result and can throw checked exceptions.
- What is a deadlock? When two or more threads are blocked forever, each waiting on the other to release a lock.
- What is ReentrantLock? A flexible alternative to synchronized with advanced features like timed lock, tryLock, fairness.
- What is the use of volatile? Ensures changes to variables are visible across threads immediately.