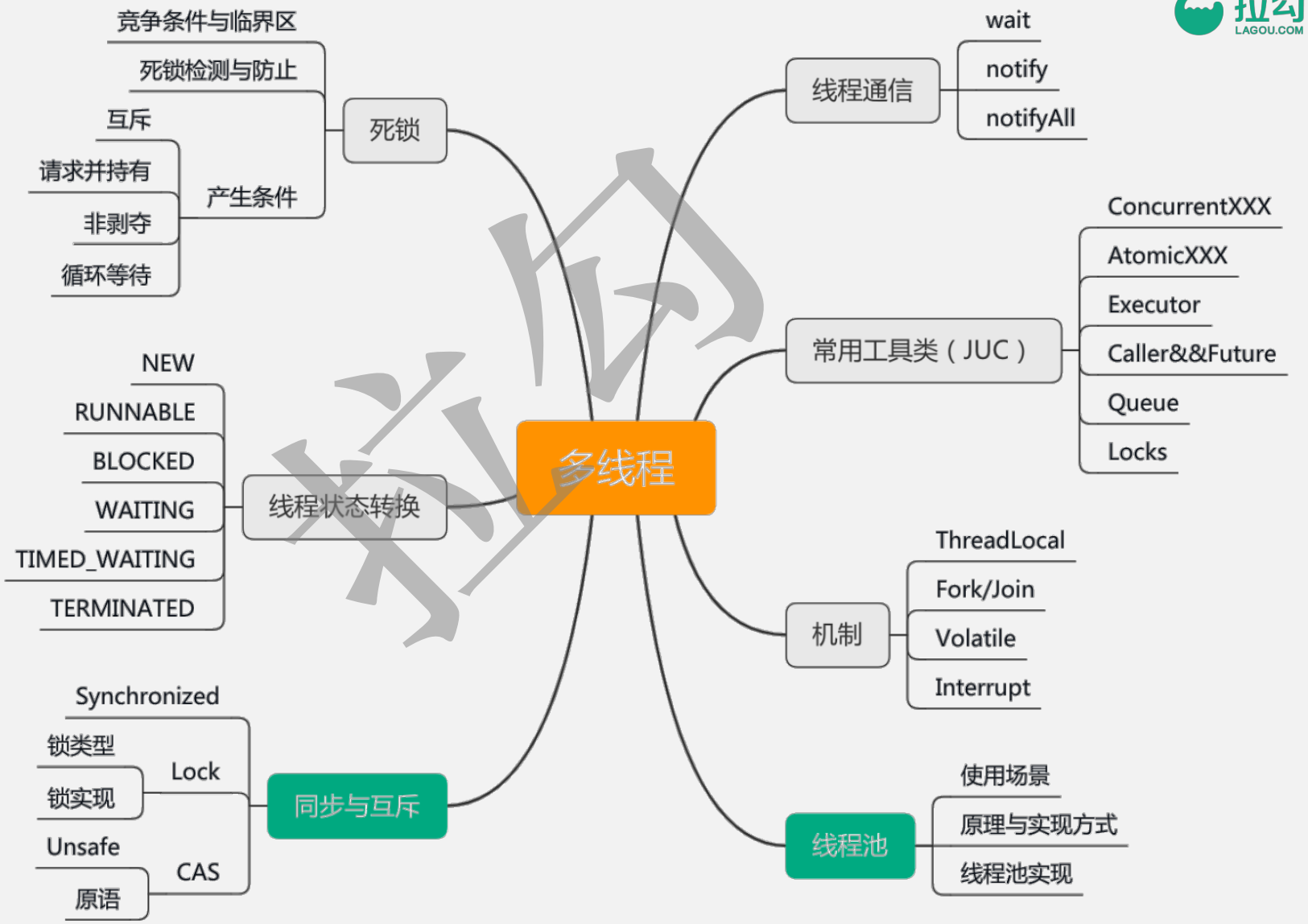
课时4

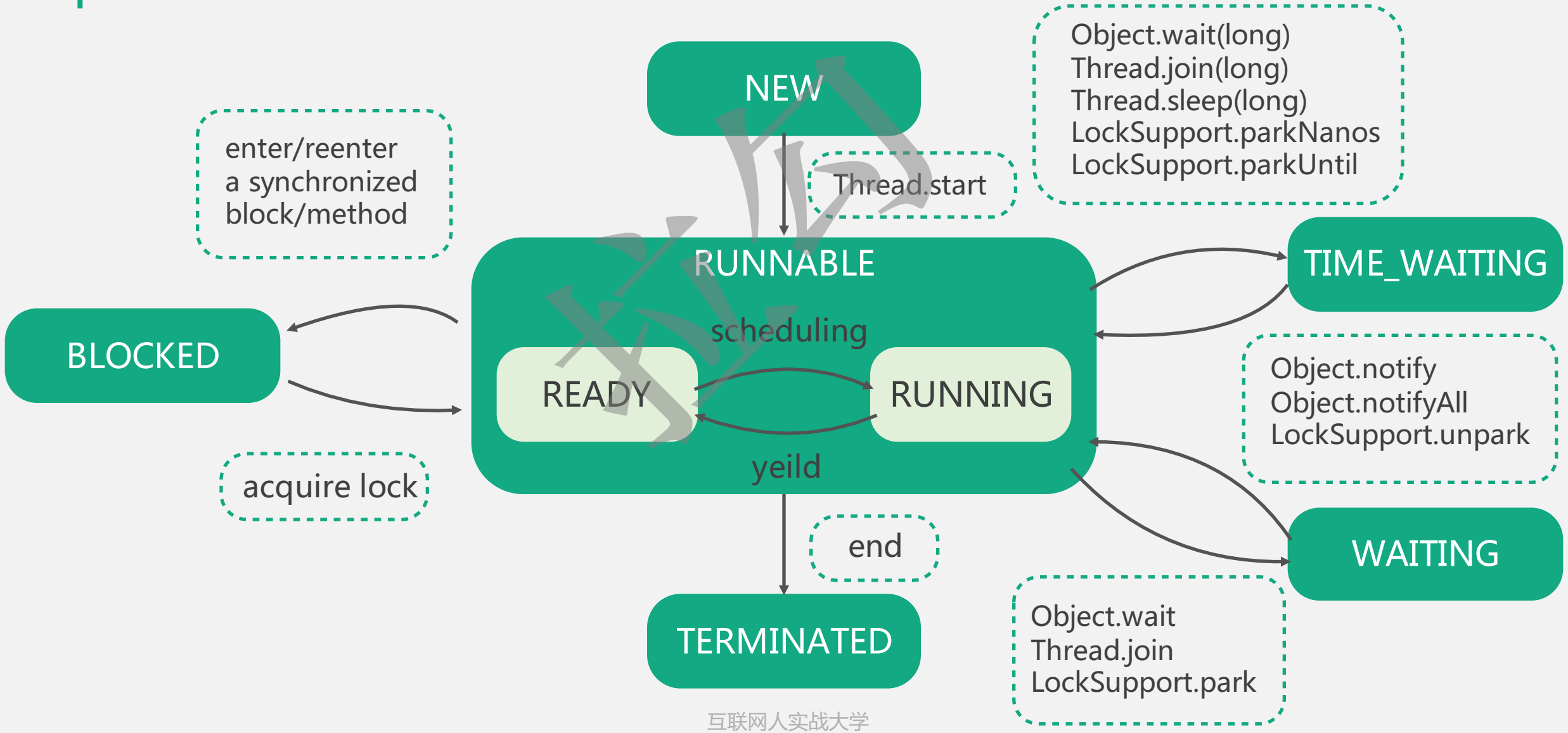
并发与多线程

1. 知识点汇总
2. 线程的状态转换
3. 线程同步与互斥
4. 线程池详解
5. JUC重点工具类实现
6. 考察点和加分项
7. 真题

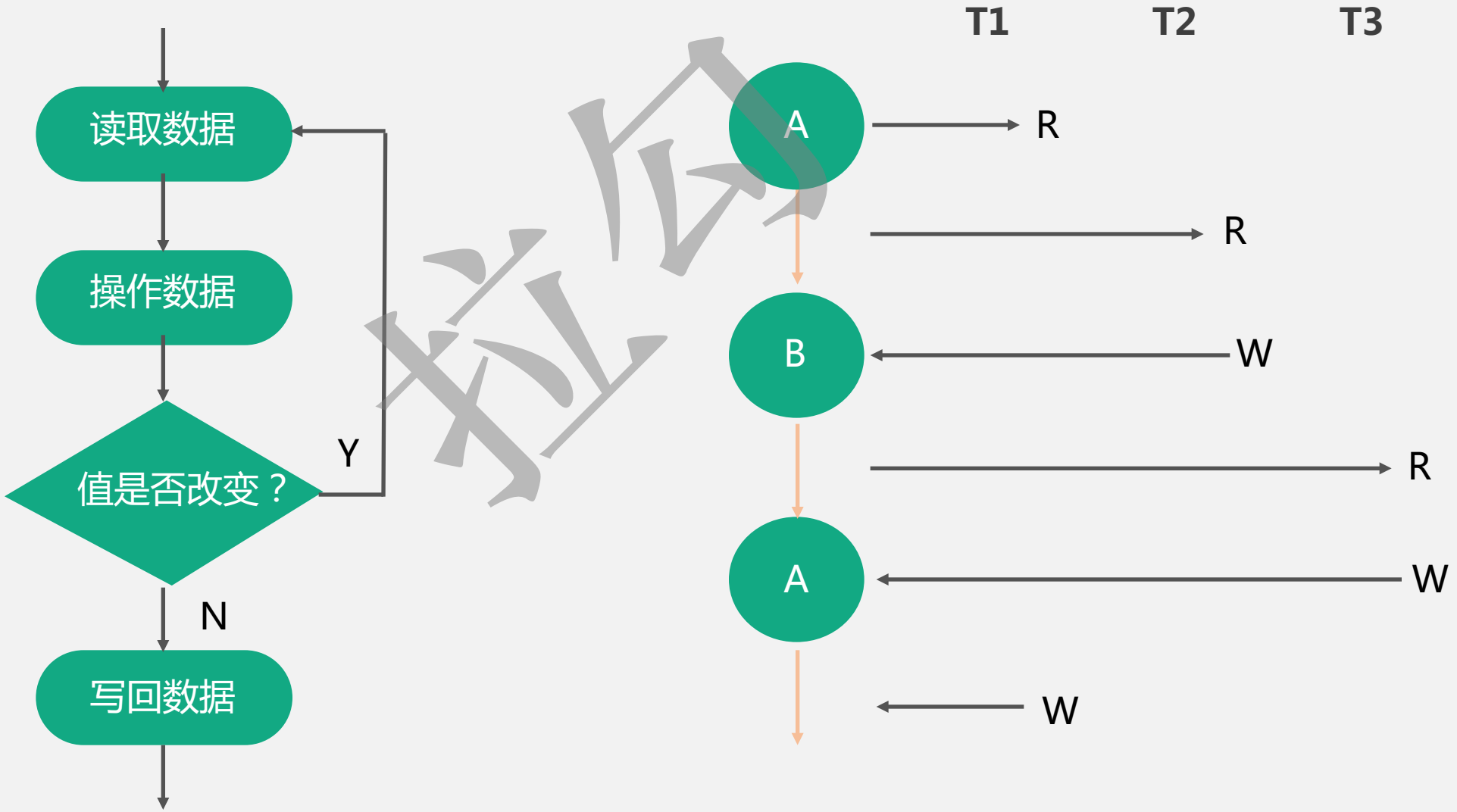
知识点汇总



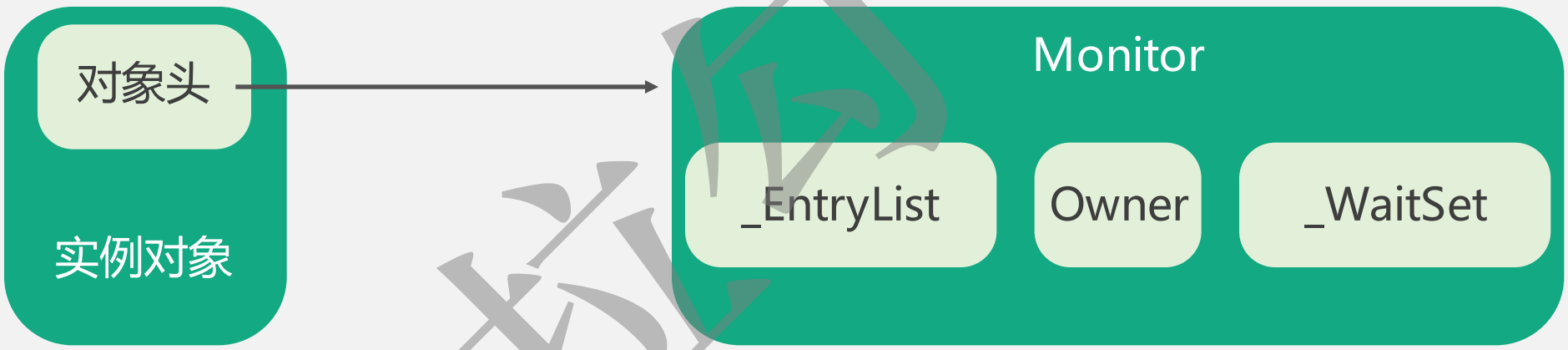
线程的状态转换


NEW
Thread.start
Object.wait(long)
Thread.join(long)
Thread.sleep(long)
LockSupport.parkNanos
LockSupport.parkUntil
enter/reenter
a synchronized
block/method
RUNNABLE
scheduling
READY
RUNNING
yeild
BLOCKED
acquire lock
TIME_WAITING
Object.notify
Object.notifyAll
LockSupport.unpark
WAITING
Object.wait
Thread.join
LockSupport.park
end
TERMINATED

CAS与ABA问题



Synchronized实现原理



方法同步：

- ACC_SYNCHRONIZED

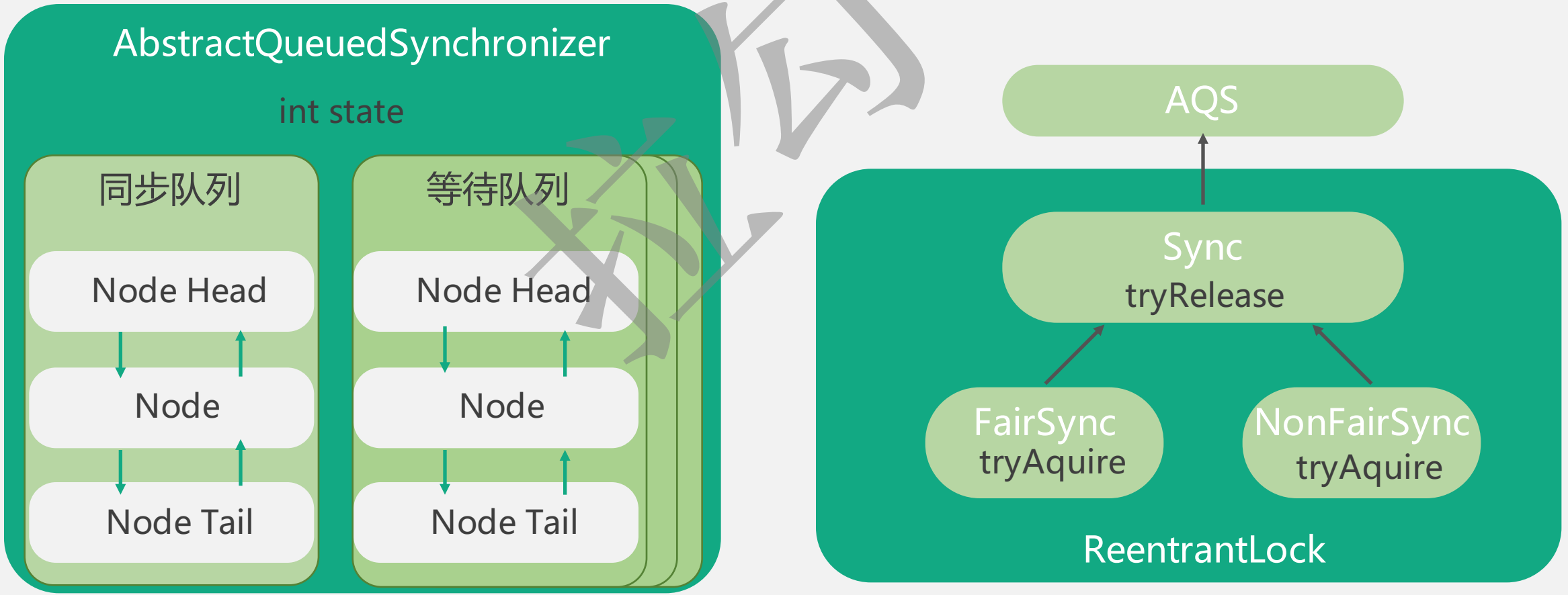
代码块同步：

- monitorenter
- monitorexit

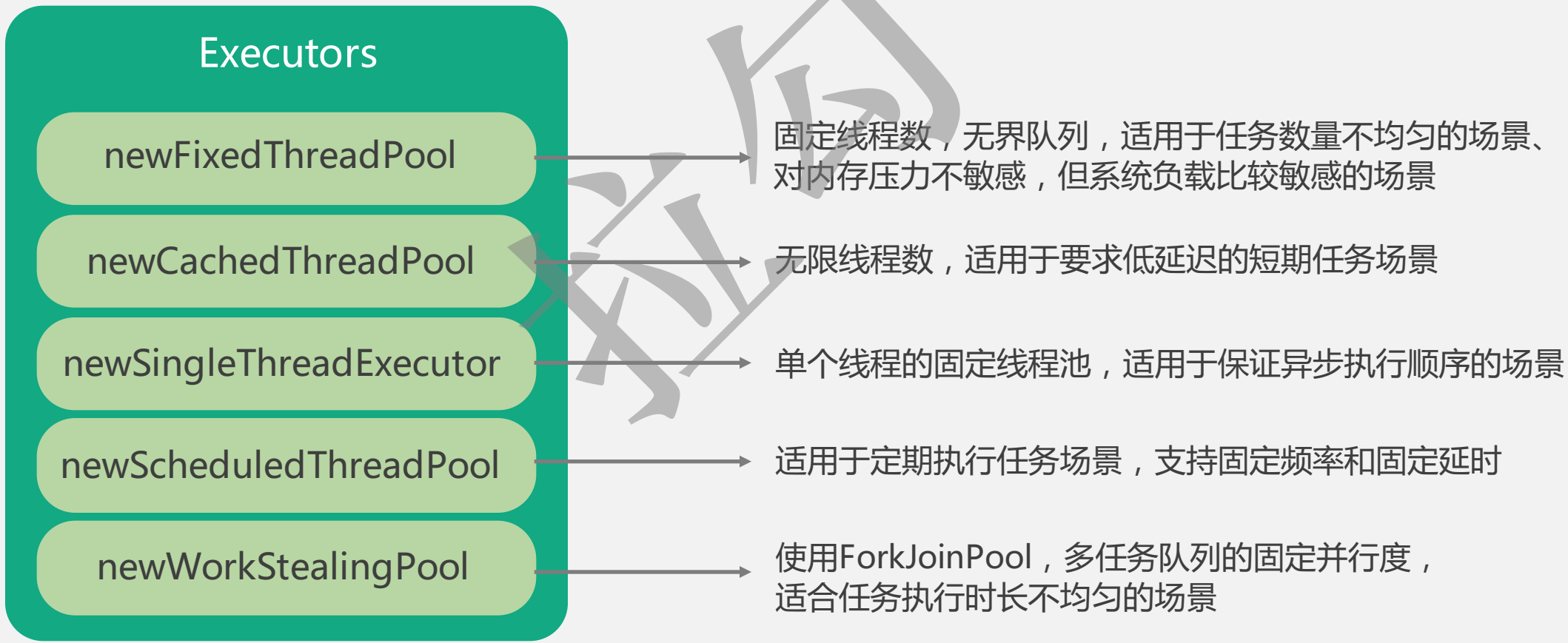
锁优化：

- 偏向锁
- 轻量锁
- 自旋锁

AQS与Lock


AbstractQueuedSynchronizer
int state
同步队列
Node Head
Node
Node Tail
等待队列
Node Head
Node
Node Tail
AQS
Sync
tryRelease
FairSync
tryAquire
NonFairSync
tryAquire
ReentrantLock

线程池适用场景


Executors
newFixedThreadPool
newCachedThreadPool
newSingleThreadExecutor
newScheduledThreadPool
newWorkStealingPool
固定线程数，无界队列，适用于任务数量不均匀的场景、
对内存压力不敏感，但系统负载比较敏感的场景
无限线程数，适用于要求低延迟的短期任务场景
单个线程的固定线程池，适用于保证异步执行顺序的场景
适用于定期执行任务场景，支持固定频率和固定延时
使用ForkJoinPool，多任务队列的固定并行度，
适合任务执行时长不均匀的场景

线程池参数介绍

ThreadPoolExecutor(int **corePoolSize**, int **maximumPoolSize**, long **keepAliveTime**, TimeUnit **unit**,
BlockingQueue<Runnable> **workQueue**, ThreadFactory **threadFactory**,
RejectedExecutionHandler **handler**)

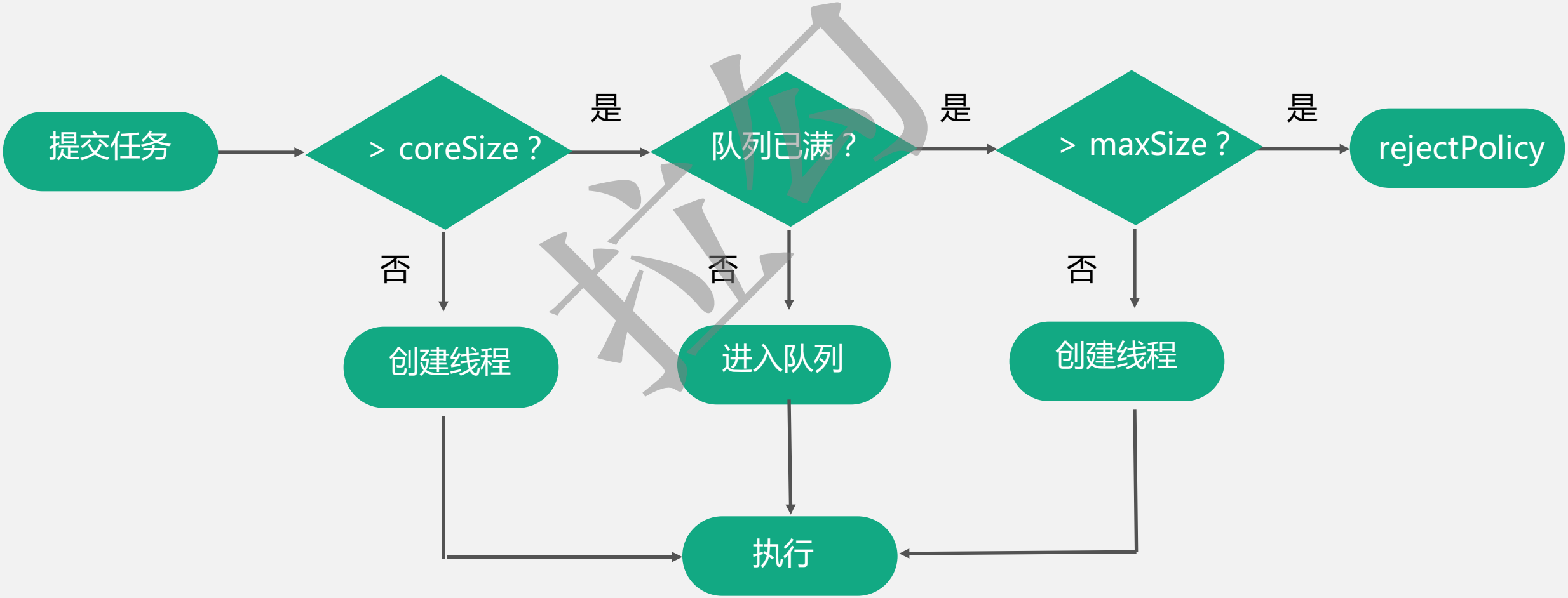
缓冲队列

- ArrayBlockingQueue(capacity)
- LinkedBlockingQueue
- SynchronousQueue

拒绝策略

- Abort
- Discard
- CallerRuns
- DiscardOldest

线程池任务执行流程



JUC常用工具

类名	特点
AtomicLong AtomicInteger AtomicBoolean **LongAdder** DoubleAdder LongAccumulator DoubleAccumulator	AtomicLong通过unsafe类实现，基于CAS LongAdder基于Cell，分段锁思想，空间换时间，更适合高并发场景
AtomicReference AtomicStampedReference AtomicMarkableReference	原子性对象读、写。 AtomicStampedReference和AtomicMarkableReference用来解决ABA问题，分别基于时间戳和标记位

JUC常用工具

类名	作用与特点
ReentrantLock **ReentrantReadWriteLock** **StampedLock** LockSupport	ReentrantLock是独占锁，Semaphore是共享锁 StampedLock是1.8改进的读写锁，CLH乐观锁，防止写饥饿
Executors ForkJoinPool FutureTask **CompletableFuture**	CompletableFuture支持流式调用，多future组合，可以设置完成时间 ForkJoinPool：分治思想+工作窃取

JUC常用工具

类名	特点
LinkedBlockingDeque ArrayBlockingQueue	LinkedBlockingDeque双端队列 ArrayBlockingQueue单端队列
CountDownLatch CyclicBarrier Semaphore	CountDownLatch：多线程任务汇总 CyclicBarrier：多线程并发执行 Semaphore：控制并发度（共享锁）
ConcurrentHashMap CopyOnWriteArrayList	COW适合读多写少，小数据量，高并发场景

面试考察点

考察点

- 理解线程的同步与互斥的原理
- 掌握线程安全相关机制
- 了解JUC工具的使用场景与实现原理
- 熟悉线程池的原理、使用场景、常用配置
- 理解线程的同步与异步、阻塞与非阻塞

加分项

加分项

- 结合实际项目经验或实际案例介绍原理
- 解决多线程问题的排查思路与经验
- 熟悉常用的线程分析工具与方法
- 了解Java8对JUC的增强
- 了解Reactive异步编程思想

真题汇总-1

拉勾 LAGOU.COM x msup®

1. 如何实现一个生产者与消费者模型？（锁、信号量、线程通信、阻塞队列等）
2. 如何理解线程的同步与异步、阻塞与非阻塞？
3. 线程池处理任务的流程是怎样的？
4. wait与sleep的有什么不同？
5. Synchronized 和 ReentrantLock 有什么不同？各适合什么场景？
6. 读写锁适用于什么场景？ReentrantReadWriteLock是如何实现的？

真题汇总-2

拉勾 LAGOU.COM x msup

7. 线程之间如何通信？
8. 保证线程安全的方法有哪些？
9. 如何尽可能提高多线程并发性能？
10. ThreadLocal用来解决什么问题？ThreadLocal是如何实现的？
11. 死锁的产生条件？如何分析是否有线程死锁？
12. 在实际工作中遇到过什么样的并发问题，如何发现（排查）并解决的？

关注订阅号：**IT进阶思维**，学习更多技术干货

Next：课时5《数据结构与算法》