

## 锁

锁是线程同步的一种手段，通过锁可以控制同一时间只能有一个线程访问共享资源。

- `synchronized` 和 `Lock` 锁的对比

## 锁

锁是线程同步的一种手段，通过锁可以控制同一时间只能有一个线程访问共享资源。

- 锁的获取和释放
- 锁的持有和超时

锁的获取和释放

`java.util.concurrent` 包中的 `CAS` 锁

`CAS` 锁的对比

- 锁的获取和释放
- 锁的持有和超时
- 锁的持有和超时

锁的持有

- 锁的获取和释放
- 锁的持有和超时

**ABA** 问题：在多线程环境下，一个变量 A 被 B 修改，B 再被 A 修改，导致 A 的值与 B 的值不一致。

- 锁的获取和释放

## 锁

锁是线程同步的一种手段，通过锁可以控制同一时间只能有一个线程访问共享资源。

- 锁的获取和释放
- 锁的持有和超时

锁

```
Lock l = new ReentrantLock(true);
```

## 锁

锁是线程同步的一种手段，通过锁可以控制同一时间只能有一个线程访问共享资源。

- 锁的获取和释放
- 锁的持有和超时

锁

```
Lock l = new ReentrantLock(false);
```

## 锁

ReentrantReadWriteLock 的 Lock 方法

- `synchronized` 和 `ReentrantReadWriteLock.writeLock()` 方法

ReentrantReadWriteLock 的 Lock 方法

ReentrantReadWriteLock 的 Lock 方法

- `ReentrantReadWriteLock.readLock()` 方法
- `ReentrantReadWriteLock.writeLock()` 方法

ReentrantReadWriteLock 的 Lock 方法

ReentrantReadWriteLock 的 Lock 方法

- `ReentrantLock` 和 `synchronized` 方法
- `ReentrantLock` 方法

## Synchronize

- `synchronized` 方法
- `synchronized` 方法

ReentrantReadWriteLock 的 Lock 方法

ReentrantReadWriteLock 的 Lock 方法

- `synchronized` 方法
- `CAS` 方法

ReentrantReadWriteLock 的 Lock 方法

ReentrantReadWriteLock 的 Lock 方法

- `synchronized` 方法
- `CAS` 方法
- `CAS` 方法
  - `compareAndSet` 方法
  - `compareAndSet` 方法

ReentrantReadWriteLock 的 Lock 方法

- `synchronized` 方法
- `CAS` 方法
- `CAS` 方法

ReentrantReadWriteLock 的 Lock 方法

ReentrantReadWriteLock 的 Lock 方法

- `synchronized` 方法

- 垃圾回收器运行效率依赖于垃圾回收策略
  - 垃圾回收策略依赖于垃圾回收器实现
- 垃圾回收器运行效率依赖于垃圾回收器CPU
  - 垃圾回收器运行效率依赖于垃圾回收器实现
    - 垃圾回收器实现依赖于垃圾回收器实现
    - 垃圾回收器实现依赖于垃圾回收器实现

## 垃圾回收

- JVM垃圾回收器实现依赖于垃圾回收器 **Lock Record** 垃圾回收器copy垃圾回收器 Mark Word 垃圾回收器
- JVM垃圾回收器实现依赖于垃圾回收器 **\*\* Mark Word** 垃圾回收器 Lock Record 垃圾回收器 Lock Record 垃圾回收器 owner 垃圾回收器 Mark Word **\*\***
- 垃圾回收器实现依赖于垃圾回收器实现
- 垃圾回收器实现依赖于垃圾回收器实现 Mark Word 垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现

## 垃圾回收

垃圾回收器实现依赖于垃圾回收器 **Mutex** 垃圾回收器实现依赖于垃圾回收器实现

- 垃圾回收器实现依赖于垃圾回收器实现
- 垃圾回收器实现依赖于垃圾回收器实现 **ObjectMonitor** 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
    - 垃圾回收器实现依赖于垃圾回收器实现
    - 垃圾回收器实现依赖于垃圾回收器实现

## 垃圾回收

- 垃圾回收器实现依赖于垃圾回收器实现
- Mark Word 垃圾回收器实现依赖于垃圾回收器实现 **\*\* ObjectMonitor \*\***垃圾回收器
- 垃圾回收器实现依赖于垃圾回收器实现

### ObjectMonitor

- 垃圾回收器实现依赖于垃圾回收器实现 **synchronized** 垃圾回收器实现依赖于垃圾回收器实现 **ObjectMonitor** 垃圾回收器
- 垃圾回收器实现依赖于垃圾回收器实现 Mark Word 垃圾回收器实现依赖于垃圾回收器实现 **ObjectMonitor** 垃圾回收器实现依赖于垃圾回收器实现 **ObjectMonitor** 垃圾回收器实现依赖于垃圾回收器实现
- 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
- 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现
    1. 垃圾回收器实现依赖于垃圾回收器实现 **\_owner==null** 垃圾回收器实现依赖于垃圾回收器实现 **\_owner**
    2. 垃圾回收器实现依赖于垃圾回收器实现 **\_owner=垃圾回收器实现** 垃圾回收器实现依赖于垃圾回收器实现 **\_recursions++**
    3. 垃圾回收器实现依赖于垃圾回收器实现 **\_EntryList** 垃圾回收器实现依赖于垃圾回收器实现
  - 垃圾回收器实现依赖于垃圾回收器实现

1. 如果 `_recursions==0` 则返回
2. 如果 `_EntryList` 不为空则返回

• 否则

1. 如果 `_WaitSet` 不为空
2. 则 `notify()` 唤醒 `_EntryList` 中的线程