线程的生命周期

线程的一生——6个状态（生命周期）

- 有哪6种状态？
- 每个状态是什么含义？
- 状态间的转化图示

每个状态是什么含义？

- New
- Runnable
- Blocked
- Waiting
- Timed Waiting
- Terminated

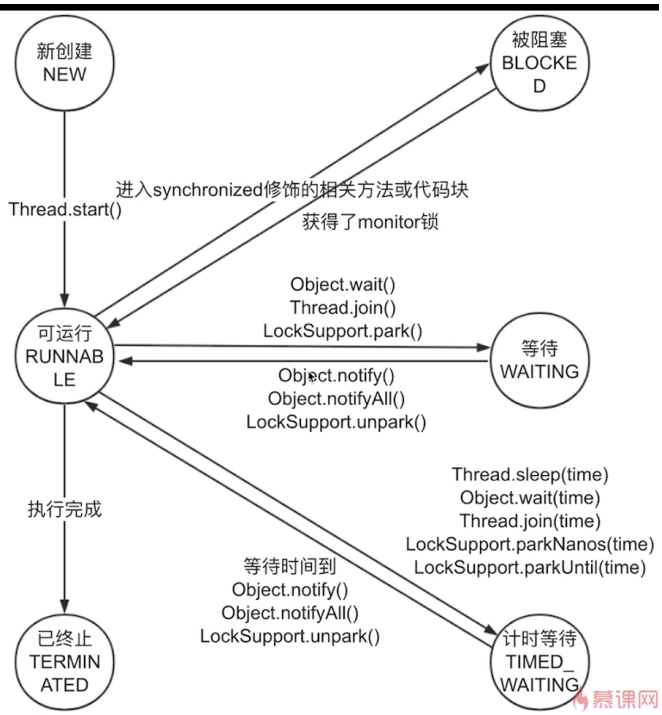
1： new 也就是在 new Thread之后，还没有 start d的状态

2： Runnable

3： Blocked 被 synchronize 修饰的，且不占有 锁的情况

状态间的转化图示

- New
- Runnable
- Blocked
- Waiting
- Timed Waiting
- Terminated



展示 new Runnable terminated

```
/**
 * 展示三种状态： new  RUNNABLE Terminated。即使正在运行，也是 Runable 状态，不是Running
 */
public class NewRunnableTerminated implements Runnable{

    public static void main(String[] args) {

        Thread thread = new Thread(new NewRunnableTerminated());
        // new 一个线程， 线程的状态是 new
        System.out.println(thread.getState());

        // 启动线程，线程的状态是 Runnable ，可运行的，可能没有抢到 CPU 资源
        thread.start();
        System.out.println(thread.getState());

        try {
            Thread.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        // 目前线程是正在运行，但是状态是 Runnable， 而不是 Running状态
        System.out.println(thread.getState());

        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        // 线程结束运行后的状态是 terminated
        System.out.println(thread.getState());


    }

    @Override
```

```
    public void run() {
        for (int i = 0; i < 1000; i++) {
            System.out.println(i);
        }
    }
}
```

展示 Time_waiting Blocked waiting

```
/**
 *  展示 blocked TimeWaiting Waiting状态
 */
public class BlockedWaitingTimedWaiting implements Runnable{

    public static void main(String[] args) {

        BlockedWaitingTimedWaiting runnable = new BlockedWaitingTimedWaiting();

        Thread thread1 = new Thread(runnable);
        thread1.start();  // 启动线程

        Thread thread2 = new Thread(runnable);
        thread2.start();

        try {
            Thread.sleep(200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // 该线程获取 锁，但是在执行 Thread.sleep(5000); 处于 TIMED_WAITING 状态
        System.out.println(thread1.getState());
        // 没有获取 锁，处于Blocked
        System.out.println(thread2.getState());

        try {
            Thread.sleep(6000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println(thread1.getState());
        System.out.println(thread2.getState());

    }

    // 被synchronize 修饰的方法
    private synchronized  void syn(){
        try {
            Thread.sleep(5000);
            wait();  // 导致线程 进入 waiting状态
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

```
    @Override
    public void run() {
        syn(); // 线程启动执行的方法
    }
}
```

阻塞状态

- 一般习惯而言，把Blocked(被阻塞）、Waiting(等待）、Timed_waiting(计时等待）都称为阻塞状态
- 不仅仅是Blocked

线程生命周期——常见面试问题

线程有哪几种状态？生命周期是什么？