1、观察线程的所有状态

线程的状态是一个枚举类型 Thread.State

```
public class ThreadDemo {
 public static void main(String[] args) {
 for (Thread.State state : Thread.State.values()) {
 System.out.println(state);
 }
 }
}

 NEW
RUNNABLE
BLOCKED
WAITING
TIMED_WAITING
TERMINATED
```

NEW ; Thread对象创建出来了,但是内核的 PCB 还没有创建出来

RUNNABLE : 当前的 PCB 也创建出来了,同时这个 PCB 随时待命,或者可能正在 CPU 上运行,也　可能在就绪列表中排队 (可工作的. 又可以分成正在工作中和即将开始工作.)

TIMED_WAITING : 表示当前的 PCB 在阻塞队列中等待,而这样的等待是一个"带有结束时间"的等待

WAITING : 线程中如果调用了 wait 方法,也会阻塞等待,此时处在 WAITING 状态。

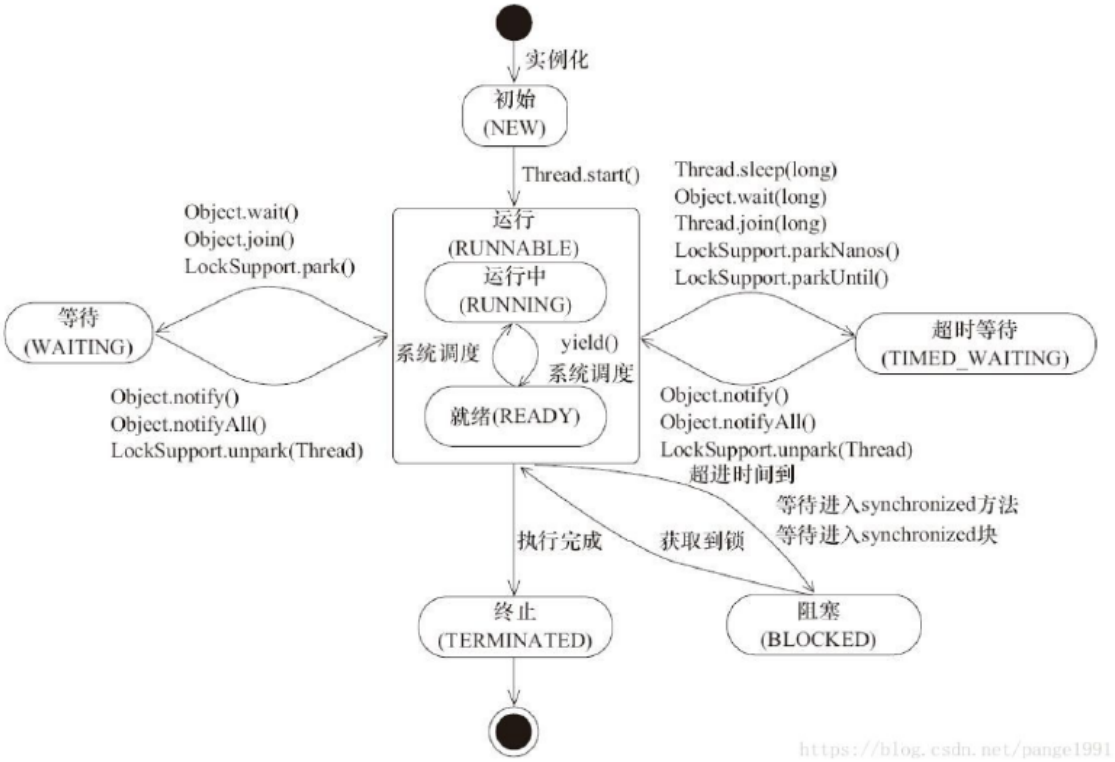
BLOCKED：线程尝试进行加锁，结果法向已经被其他线程占用了，此时线程也会阻塞等待，这个等待在其他线程释放锁之后，被唤醒

TERMINATED： 表示当前的 PCB 已经结束了，Thread 对象还在，此时调用获取状态，就是这个结果。

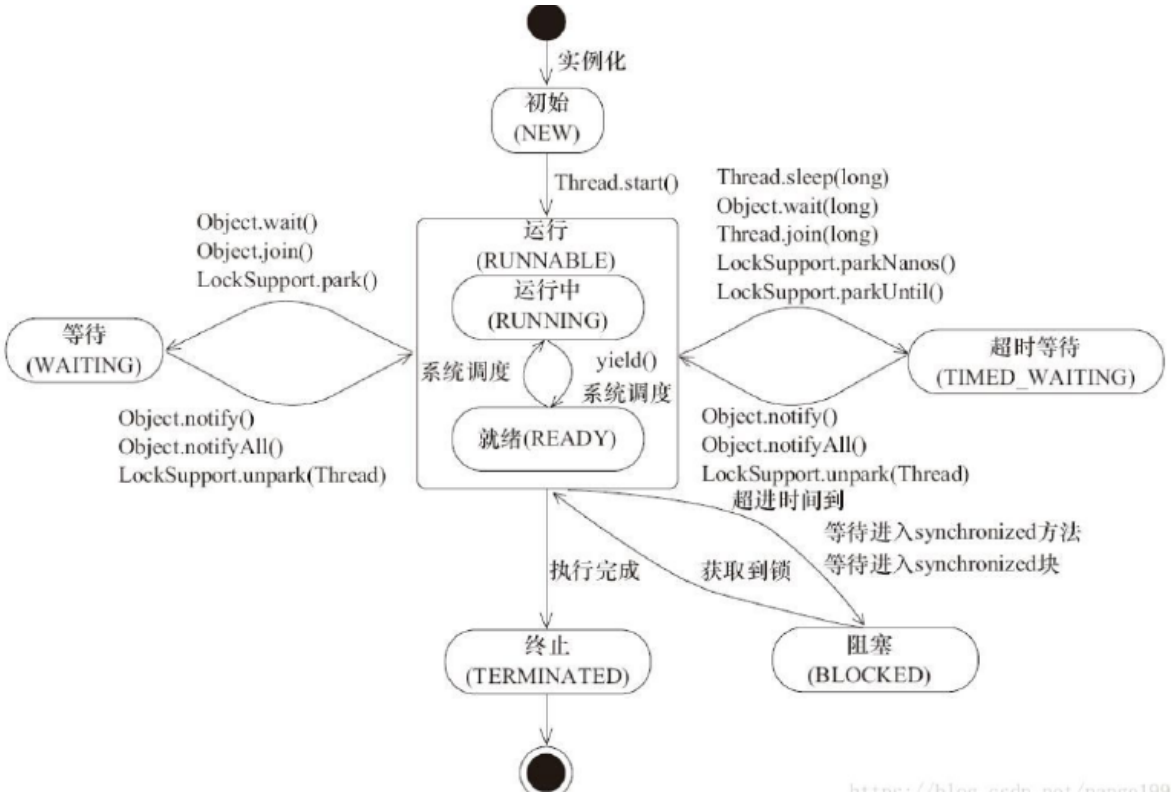
注：TIMED_WAITING、WAITING 、BLOCKED都表示阻塞等待了，结束阻塞等待的条件不一致。

2、线程状态和线程转移的意义

用于辅助系统对于线程进行调度这样的属性,理解线程的状态,最大的意义在于未来调试一些多线程的程序.



3、观察线程的状态和转移



观察1: 关注 NEW 、RUNNABLE 、TERMINATED 状态的转换

```
1 // 使用 isAlive 方法判定线程的存活状态.
2 public class ThreadDemo9 {
3  public static void main(String[] args) {
4  Thread t = new Thread(() -> {
5  for(int i = 0;i < 100; i ++){
```

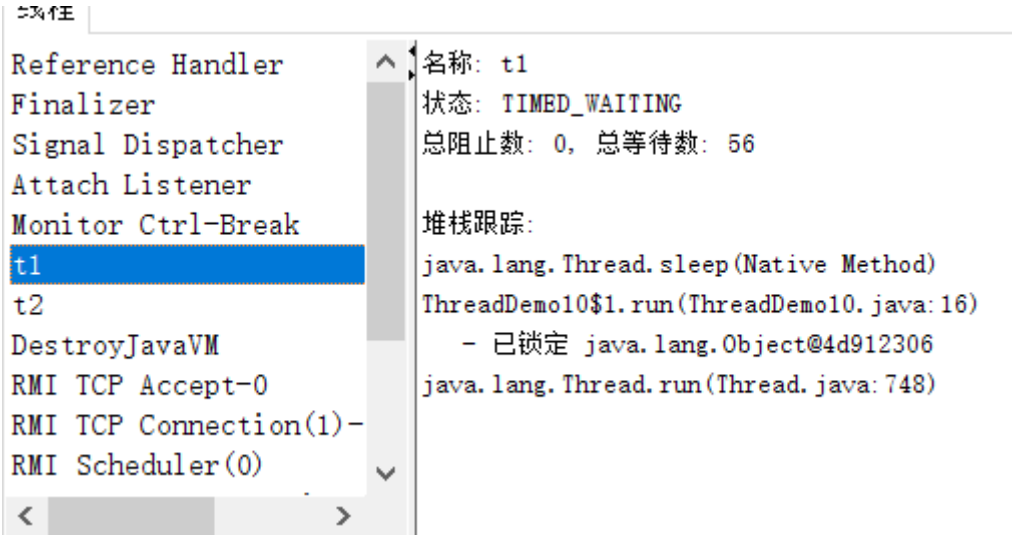
```
6
7   }
8   },"李四");
9   System.out.println(t.getName() + ":" + t.getState());
10   t.start();
11   while(t.isAlive()){
12   System.out.println(t.getName() + ":" + t.getState());
13   }
14   System.out.println(t.getName() + ":" + t.getState());
15   }
16  }
17
18  李四:NEW
19  李四:RUNNABLE
20  李四:RUNNABLE
21  李四:RUNNABLE
22  李四:RUNNABLE
23  李四:RUNNABLE
24  李四:TERMINATED
```

观察 2: 关注 WAITING 、BLOCKED 、TIMED_WAITING 状态的转换

```
1  public class ThreadDemo10 {
2   public static void main(String[] args) {
3   final Object object = new Object();
4   Thread t1 = new Thread(new Runnable() {
5   @Override
6   public void run() {
7   synchronized (object){
8   while(true){
9   try {
10   // Thread.sleep(1000);
11   object.wait();
12   } catch (InterruptedException e) {
13   e.printStackTrace();
14   }
```

```
15  }
16  }
17  }
18  },"t1");
19  t1.start();
20
21  Thread t2 = new Thread(new Runnable() {
22  @Override
23  public void run() {
24  synchronized (object){
25  System.out.println("hehe");
26  }
27  }
28  },"t2");
29  t2.start();
30  }
31 }
```

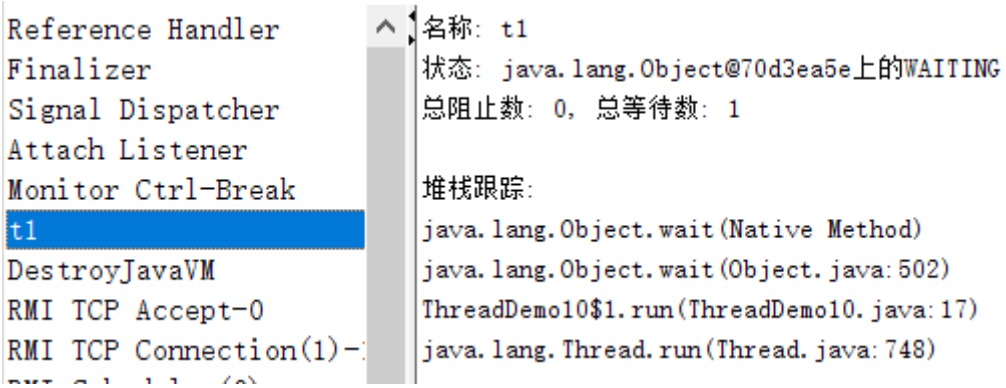
使用 jconsole 可以看到 t1 的状态是 TIMED_WAITING , t2 的状态是 BLOCKED



修改上面的代码, 把 t1 中的 sleep 换成 wait

```
public static void main(String[] args) {
 final Object object = new Object();
 Thread t1 = new Thread(new Runnable() {
 @Override
 public void run() {
 synchronized (object) {
 try {
 // [修改这里就可以了!!!!!]
 // Thread.sleep(1000);
  object.wait();
  } catch (InterruptedException e) {
  e.printStackTrace();
  }
  }
  }
  }, "t1");
  ...
}
```

使用 jconsole 可以看到 t1 的状态是 WAITING



总结:

1. BLOCKED 表示等待获取锁, WAITING 和 TIMED_WAITING 表示等待其他线程发来通知.

2. TIMED_WAITING 线程在等待唤醒，但设置了时限; WAITING 线程在无限等待唤醒

观察-3: yield() 大公无私，让出 CPU

```
1  public class ThreadDemo11 {
2   public static void main(String[] args) {
3   Thread t1 = new Thread(new Runnable() {
4   @Override
5   public void run() {
6   while (true) {
7   System.out.println("张三");
8   // 先注释掉, 再放开
9   Thread.yield();
10   }
11   }
12   },"t1");
13   t1.start();
14
15   Thread t2 = new Thread(new Runnable() {
16   @Override
17   public void run() {
18   while (true) {
19   System.out.println("李四");
20   }
21   }
```

```
22    }, "t2");
23    t2.start();
24    }
25  }
```

可以看到:

1. 不使用 yield 的时候, 张三李四大概五五开
2. 使用 yield 时, 张三的数量远远少于李四

结论:

yield 不改变线程的状态, 但是会重新去排队.