

作者：JavaGieGie

微信公众号：Java开发零到壹

前言

某一天清晨（9:00）,装逼界俩泰斗早早到了公司。

狗剩：龙Gie，去不去蹲坑，趁现在没人，去占个好位置，然后给你看个好宝贝。

我：臭不要脸，大清早就想着蹲坑划水，我平是怎么教育你的，能不能向我学习。



我：真香.....

狗剩子：花Gie，听说隔壁组的毛孩离职进了那个姓阿的公司，这尼玛简直太气人了。

我：你还别不服气，人家肚里真的有货，不信我问你几个问题，看看你能不能答上来？

正文

我：你用过多线程吗？创建线程的方式有几种？

这种问题还拿出来问，你是来侮辱俺的智商么，创建线程的典型方式有两种，分别是实现Runnable接口和继承Thread类，此外还可以通过线程池、定时器、匿名内部类等来进行创建。

我：还有其他的吗？

心中嘀咕：这特么话问的想给我下套啊。啊..那个..有的。

虽然说了这么多创建线程的方式，但我们查看源码就会发现，其实本质上只有一种，那就是通过新建Thread类来创建线程，并最终通过start方法来新建线程，只是run方法的实现有两种，第一种继承Thread类是重写父类run方法，而第二种实现Runnable是对接口的run方法进行实现，然后将Runnable实例传递给Thread类。

- 新建Thread类，run方法实现

```
//实现方式
class ThreadTest extends Thread{
    @Override
    public void run() {
        System.out.println("我爱杰伦");
    }
}

Thread thread = new Thread(new ThreadTest());
thread.start();

//线程最终调用target.run()
private Runnable target;
@Override
public void run() {
    if (target != null) {
        target.run();
    }
}
```

- 实现Runnable，run方法实现

```
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        System.out.println("大家好，我叫狗剩子");
    }
});
thread.start();
```

至于线程池、定时器等工具类本质上也是上述的一种实现。

小样，是不是偷看过花Gie的笔记了,那你知道继承Thread和实现Runnable接口哪种更好？

实现Runnable接口更好，主要原因有以下几个：

- Java是单继承，如果继承Thread类后，会限制扩展性。
- 实现Runnable接口将具体的任务（run方法）和创建线程（Thread类）分开实现，这样可以使同一个任务类传递给不同的线程，任务类不负责创建线程等工作，两者各司其职，从而实现解耦。
- 使用继承Thread类创建线程，每次新建任务，只能创建一个新的线程，即使该任务只是打印一行日志，也要完成线程的创建销毁等过程，造成资源的严重浪费。而Runnable和线程池就能避免这种浪费。

那你知道线程有哪几种状态？它的生命周期是什么吗？

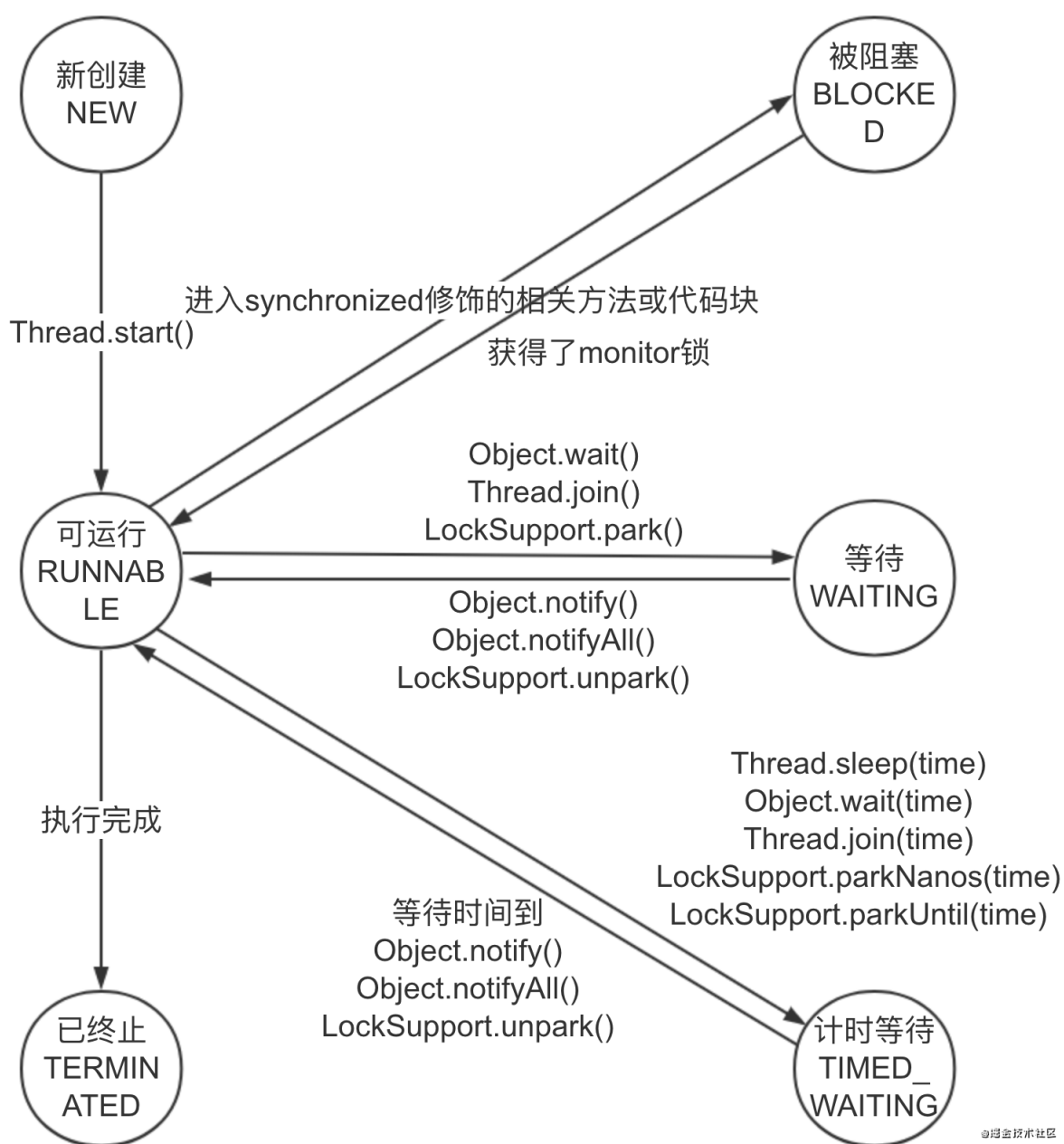
花Gie，你把手伸过来，我给你看个宝贝。



来来来，我给你看个宝贝。

什么东西热乎乎的，你特么....讨厌！

不好意思，情不自禁，把我热乎乎的大包子拿出来了，在这...是这个



这个图是我见过画的最好的图，我们可以看到java线程共计包含六个生命周期。

- **新建(New):** 创建一个线程对象。

1. JVM为其分配内存，并初始化其成员变量的值；
2. 该状态下线程也不会得到调度。

- **就绪状态 (Runnable)**：当线程对象调用start方法之后，就会进入就绪状态，需要注意的是线程获取到时间片后依旧会处以Runnable状态。

1. JVM创建方法调用栈和程序计数器；
2. 该状态的线程一直处于线程就绪队列（尽管采用的是队列形式，事实上，把它成为可运行池而不是可运行队列。因为CPU的调度不一定是按照先进先出的顺序来调度的），线程并没有开始运行；
3. 线程并不是说执行了start()方法就立即执行，需要等待系统为其分配CPU时间片。

- **阻塞状态 (Blocked)**：处于运行状态的线程，遇到某些情况会让出自己的CPU执行权，进入阻塞状态。

进入阻塞状态的情况：

1. 线程试图获取synchronized方法/代码块，但monitor锁被其他线程占用；
2. 线程调用sleep方法，主动进入休眠，待休眠结束进入就绪状态等待获取CPU执行权；
3. 线程调用持有锁的wait方法；
4. 线程调用阻塞式IO方法，在方法返回前，该线程一直被阻塞。

- **死亡 (Terminated)**：线程执行完成或者抛出异常，线程资源被回收。

线程进入死亡状态3种方法：

1. run()或call()方法执行完成，线程正常结束；
2. 线程抛出一个未捕获的异常；
3. 直接调用该线程的stop()方法——容易导致死锁，不推荐使用。

敲黑板：下图是Java官方文档，Java线程没有Running状态，其Runnable包含了操作系统中ready和running状态。

Enum Constant Summary

Enum Constants

Enum Constant and Description

BLOCKED

Thread state for a thread blocked waiting for a monitor lock.

NEW

Thread state for a thread which has not yet started.

RUNNABLE

Thread state for a runnable thread.

TERMINATED

Thread state for a terminated thread.

TIMED_WAITING

Thread state for a waiting thread with a specified waiting time.

WAITING

Thread state for a waiting thread.

哟，小伙子有点东西啊，上面你说线程调用start()方法最终调用的是run()方法，

那我们为什么不直接调用run()方法呢？

因为start()是用来启动线程，run()方法只是执行线程运行时的代码，如果直接调用run()方法，也仅仅是调用一个普通的方法而已，和线程的生命周期是没有关系的，还有我们需要注意start()方法调用第二次会报运行时异常。

那你知道sleep和wait/notify的区别是什么吗？

这个也休想难倒咱家。

- 相同：
 1. 他们都可以让线程进入阻塞状态。
 2. 他们都可以响应中断Thread.interrupt。
- 不同点
 1. sleep() 来自 Thread，wait() 来自 Object。
 2. sleep() 不释放monitor锁，但wait() 会释放。
 3. sleep() 时间到会自动恢复，wait() 可以使用 notify()/notifyAll()直接唤醒。
 4. wait方法需要在同步方法中执行，而sleep不需要。

你知道yield和sleep有什么区别吗？

好家伙，还好我准备的充足，yield的作用是让出自己的时间片，线程依旧处于Runnable状态，依然有可能被再次调度，而sleep被调用后会进入阻塞状态，在等待时间内不会被线程调度器调用，

回答的还可以，都快赶上我了，那还有一个问题，守护线程和普通线程的区别你有了解吗？

午餐时间到



握草，午饭时间到了，不和你说了，屁股都风干了，今天暂且到这，等老衲补充一下营养，明天再会会你。


总结

以上就是今天介绍的多线程基础知识，这块知识点比较多，这里花Gie也会分多个篇幅来介绍，我敢向狗剩子保证，只要你认真跟好花Gie的这一系列，面试必能小虐面试官，如果不能...那我把狗剩子送给你。

万丈高楼平地起，花Gie认为只有打好基础，认真修炼内功，才能走的更轻松，也更远。

点关注，防走丢

以上就是本期全部内容，如有纰漏之处，请留言指教，非常感谢。我是花GieGie，有问题大家随时留言讨论，我们下期见👋。

文章持续更新，可以微信搜一搜「**Java开发零到壹**」第一时间阅读，后续会持续更新java面试和各类知识点，有兴趣的小伙伴欢迎关注，一起学习，一起哈哈。

原创不易，你怎忍心白嫖，如果你觉得这篇文章对你有点用的话，感谢老铁为本文**点个赞、评论或转发一下**，因为这将是我输出更多优质文章的动力，感谢！