# Java多线程的学习

目录

[Java多线程的学习 1](#_Toc4232728)

[前文提要: 1](#_Toc4232729)

[问题1：什么是线程？ 2](#_Toc4232730)

[线程的定义： 2](#_Toc4232731)

[线程的几种状态： 2](#_Toc4232732)

[线程新建的两种方式： 3](#_Toc4232733)

[线程的中止和中断： 4](#_Toc4232734)

[线程的等待和唤醒 5](#_Toc4232735)

[线程等待和休眠有什么区别？ 5](#_Toc4232736)

[等待线程结束（join）和谦让（yield）: 5](#_Toc4232737)

[volatile（可见变量说明）： 6](#_Toc4232738)

[线程组（ThreadGroup）： 6](#_Toc4232739)

[守护线程（Daemon） 6](#_Toc4232740)

[线程安全的概念与synchronized： 6](#_Toc4232741)

## 前文提要:

对于Java多线程的学习主要还是从《Java高并发程序设计》入门起，只看过一遍，在参加了春招面试才发觉自己只学了个皮毛，因此来重新学过，以此来记录下来。

本次复习采取结合面试题的方式来进行。先提出一些问题，做出解答，并补充知识点。

## 问题1：什么是线程？

线程的定义：在讲线程之前，我们都会先结合一下进程来讲，首先，什么是进程。

1，进程是什么，进程是指一个正在执行中的程序，一个程序运行活动，是系统进行资源和调度的基本单位。当你看到一个.exe文件的时候，他就是一个静态的程序，而当你点击运行他的时候，他就是一个进程了。一个进程中可以包含多个线程。进程是线程的一个容器。

2，前面我们说到了进程是线程的容器，所以说，一个进程中就可以包含有多个线程，线程是程序执行的的最小单位，我们可以通过多线程去进行并发程序的执行，因为线程的调度和切换的成本远远小于进程。

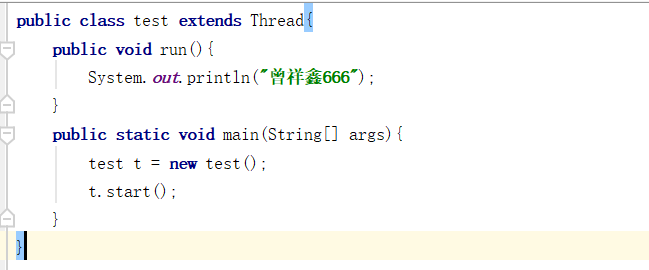
### 线程的几种状态：

1. 新建，我们会先创建出一个线程出来,这个线程是还没有start的。
2. 就绪，我们要知道，线程是不会自己运行的，所以就需要我们去启动他，我们用start 去启动它， 这个时候线程就会处于就绪状态，但是，因为系统的cpu在每一时刻都是有限的，所以就算你开始了，你也不能保证你的线程就能马上得到cpu的资源，这种转台我们称为就绪状态。
3. 运行，获取到了我们的cpu资源，开始线程的运行。
4. 阻塞，当线程遇到一些特殊情况下，就会处于阻塞的状态，在这种情况下，线程就会暂时地让出cpu的资源，让其他处于就绪状态的线程进行运行状态。由阻塞的原因分为以下几种情况。
   1. 等待写入阻塞：由于在等待io 的过程中，所以导致的线程阻塞的
   2. 锁的阻塞：当线程在等待一个锁，而这个锁被别的线程占用着，因此导致的线程阻塞。
   3. 当线程处于sleep的状态下，也会导致阻塞。（等下我们就补充以下sleep和wait 的区别）称为睡眠阻塞。
   4. 当线程还没有被触发某个条件的时候。称为条件阻塞

### 线程新建的两种方式：

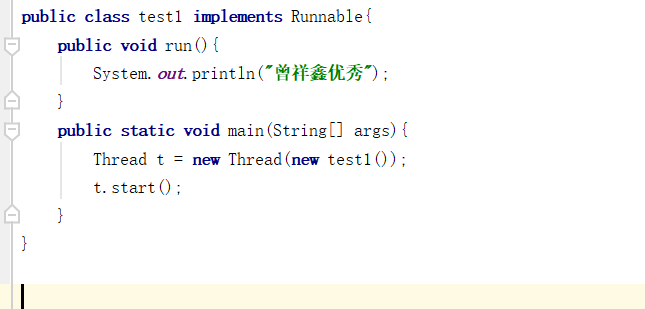
1. 通过继承Thread的方式；

也就是我们创建一个类去继承Thread ，然后在里面去重写我们的run方法，而这个继承了Thread的类，也就编程了一个线程。而我们要做的就是去新建一个线程然后start它。



1. 通过实现runnable的方式；

在这种方式中，我们通过新建一个类去实现Runnable接口，而这个接口就是定义我们的Run的内容，也就是线程的运行内容，然后我们新建一个线程，也就是new Thread，然后将我们的线程的执行内容传进去，并start它。这是因为呢，Thread 有一个构造方法，public Thread(Runnable target);当调用start的时候，就会去执行target里面的run犯法。



### 线程的中止和中断：

线程如何去终止呢，这里我们有两种方式，其中一种使用stop（）方法， 但是我们已经不用这个了，另外一个就是设置一个flag，在run()方法的while循环中去终止这个线程。

#### 为什么不能用stop()方法呢？

因为stop（）会强制地中断掉线程，包括它的锁，这样有可能会导致我们的数据出现不一致的问题，具体的场景如下：假如我正在设置两个变量，而且还是锁着的情况，当我刚设置完一个变量，但是这个时候，我被stop给中断了，锁也没有了，这个时候别人在读 的时候，就会显示两个变量的值不一致了。这就是为什么我们不用stop方法了。

#### 使用标记去中断run里面的while循环：

中断和中止是不一样的，前面说的中止只是一种自由的机制，但是中断比中止要更加完善和友好，当你要中断的时候，就是发送一个通知给线程，而线程何时中断，就由它自己去决定。

下面呢，我们讲一下关于中断的两个注意事项：

1，中断如何去使用，我们首先要先了解几个方法；Thread.interrupt() //中断线程；

Thread.isInterrupt() //判断线程是否中断；

一般情况，我们可以对线程执行中断操作，但是，如果我们没有在程序中设置中断之后需要进行的操作的话，那就是没有用的了，所以呢，我们要在一个if判断中判断是否已经中断，然后如果中断的话，我们就退出while循环。这个其实跟上面的标记方法类似，但是呢。它有一个好处就在于sleep（）方法了。也就是第二个注意事项；

2，当我们在使用sleep 的时候，我们要时刻注意sleep（）方法是否会被中断，抛出中断异常，所以我们需要一个try-catch,当我们捕捉到了之后,我们需要在catch中重新地中断一次，因为当抛出异常的时候，我们的程序是会清除中断标记的，因此我们必须要设置中断，不然下次循环的时候，我们就无法捕获这个中断异常了。

### 线程的等待和唤醒

线程的等待和唤醒是一个重点我们需要掌握的知识点好吧。

首先，我们必要要知道的一点就是：等待wait和唤醒notify是对象上的方法，而sleep呢是线程Thread上的方法，那么等待唤醒机制究竟是怎么样的呢，

首先，我们需要先获得一个锁以及锁的对象，也就是说，我们要先用synchronized（object） 的方式，这样的话，我们就可以将object对象进行wait，这样的话，也就是说我们将对象进行了等待，线程就会进入等待状态，而进而将锁给释放出来，给其他线程中，那么假如其他的线程也调用了这个对象的wait 方法的话，那么就进入等待队列，那么当我使用唤醒notify的时候，将随机地唤醒一个正在等待中的线程。使用notify 的话，则会唤醒所有在wait 的线程。

### 线程等待和休眠有什么区别？

1. 首先，sleep是线程的方法，而wait是对象的方法。
2. wait会释放cpu资源和锁，而sleep会释放资源但是不会释放锁。
3. sleep需要指定休眠的时间，而wait可以不用指定等待的时间
4. sleep可以在任何地方使用，而wait只能在同步锁的地方使用。

### 等待线程结束（join）和谦让（yield）:

在这里呢，我们要强调说明以下join是什么意思？所谓join呢，就是当我们这个线程调用另一个线程的join的时候，也就是..join前面的线程对象的时候，我们的这个线程就必须等到join前面的那个线程执行完毕之后，我们才能够继续执行下去，而join（）的本质其实就是一个wait（）方法，也就是说，我将自己的本身的这个线程给等待掉，而当join前面的线程执行完之后，我再去唤醒自己本身这个线程。

而谦让yield就是我这个线程已经完成我最基本的操作啦，我可以释放我的cpu资源执行权，但不是放弃，而是大家重新一起来参与抢夺中。

### volatile（可见变量说明）：

当我们在变量的前面加上这个volatile的时候，就是在告诉程序，我的这个变量有可能会在线程执行的过程中变化，所以你要保证他的可见性，如果他变化了的话，其他线程要知道这件事情。

### 线程组（ThreadGroup）：

将相同功能的线程归到一个线程组里面去。可以进行统一的配置和调度。

### 守护线程（Daemon）

当我们的线程设置为守护线程的时候，就是将线程设置为守护我们的这个线程的线程，当守护的对象消失了，那么我们的这个守护线程也会随之消失，如果不设置的话，我的主线程

消失的时候，我的这个线程还会一直执行下去。

### 线程安全的概念与synchronized：

关键字synchronized的多种用法：

1. 指定加锁对象，对给定对象进行加锁；
2. 直接作用于实例方法，相当于对当前实例加锁，
3. 直接作用于静态方法，相当于对当前类枷锁。

## JDK并发（多线程入门）

### synchronized的功能扩展：重入锁；

重入锁的实现是用ReentrantLock来实现的，先声明一个重入锁，然后在run（）方法里面就可以进行加锁，记得最后要把锁给解开，还有就是为什么叫做重入锁，就是因为这个锁是可以反复进入的，当然这个进入是指同一个线程的。