1. 多线程的简介
2. 概述

多线程能够充分利用CPU的空闲时间片，提升进程的整体运行效率。同一进程的所有线程时共享同一内存的，不需要特殊的数据传送机制，不需要建立共享存储区或者共享文件。

1. 进程与线程
2. 进程

每个应用程序可以运行更多个进程，每个进程都有一个独立的内存空间，进程是程序的基本单位。

1. 线程

进程内部的一个独立执行单元，一个进程可以并发运行多个线程。

1. 进程与线程的区别
2. 多线程的执行顺序是随机的，程序员无法控制。
3. Java程序的进程中至少有两个线程，主线程main和垃圾回收线程。每执行一个类时，都会启动一个JVM，每个JVM就是启动了一个线程，且Java具有垃圾回收机制，因此垃圾回收线程会作为守护线程一直运行。
4. 创建一个线程的开销比创建一个进程的开销小得多，因此多任务并发时，通常都是创建多线程。
5. 多线程的创建
6. 三种创建方式
7. 继承Thread类

创建线程类来继承Thread类，重写run方法的逻辑。

class myThread extends Thread{

public void run(){}

}

1. 实现Runnable接口

创建类实现Runnable接口，重写run方法，在创建线程类时传入该类即可。

class myRunnable implements Runnable{

public void run(){}

}

1. 匿名内部类

直接通过在创建新的线程时传入一个匿名的内部类实现Runnable接口，可以快速地实现每个线程执行不同的任务。

new Thread(new Runnable(){

public void run(){}

})

1. 实现Runnable接口的优势
2. 多个线程可以共享Runnable接口的类中的资源变量
3. 避免Java中类的单继承的局限性，可以实现多个接口
4. 一个Runnable类可以被多个线程共享，实现代码和数据的独立，实现解耦操作，增强程序的健壮性
5. 线程池中只能放入实现Runnable接口或callable接口的类线程，不能直接放入继承Thread类的线程。
6. 守护线程

Java中有两种线程，一种是用户线程，另一种是守护线程。用户线程是用户自定义的线程，当主线程停止时，用户线程不会停止；守护线程是后台运行的线程，当进程不存在或者主线程停止时，守护线程也会停止。

1. 线程状态
2. 线程状态介绍

Java的线程共有六种状态，分别是NEW, RUNNABLE, BLOCKED, WAITING, TIMED\_WAITING, TERMINATED。

1. NEW（新建）

线程刚被创建，但是并没有启动。

1. RUNNABLE(可运行)

线程可在JAVA虚拟机中运行的状态，可能正在运行，也可能没有，取决于处理器。因此，这个状态内部也有两种状态，一种是准备运行，另一种是正在运行。

1. BLOCKED（锁阻塞）

当该线程想要获取的对象锁被其他的线程持有时，该线程进入阻塞状态，当该线程获取到对象锁后，变为RUNNABLE状态。

1. WAITING（无限等待）

一个线程在等待另一个线程执行唤醒操作，这个线程就会进入WAITING状态，这个状态是无法自动唤醒的，只能通过另一个线程的notify()或者notifyAll()方法唤醒。

1. TIMED\_WAITING(计时等待)

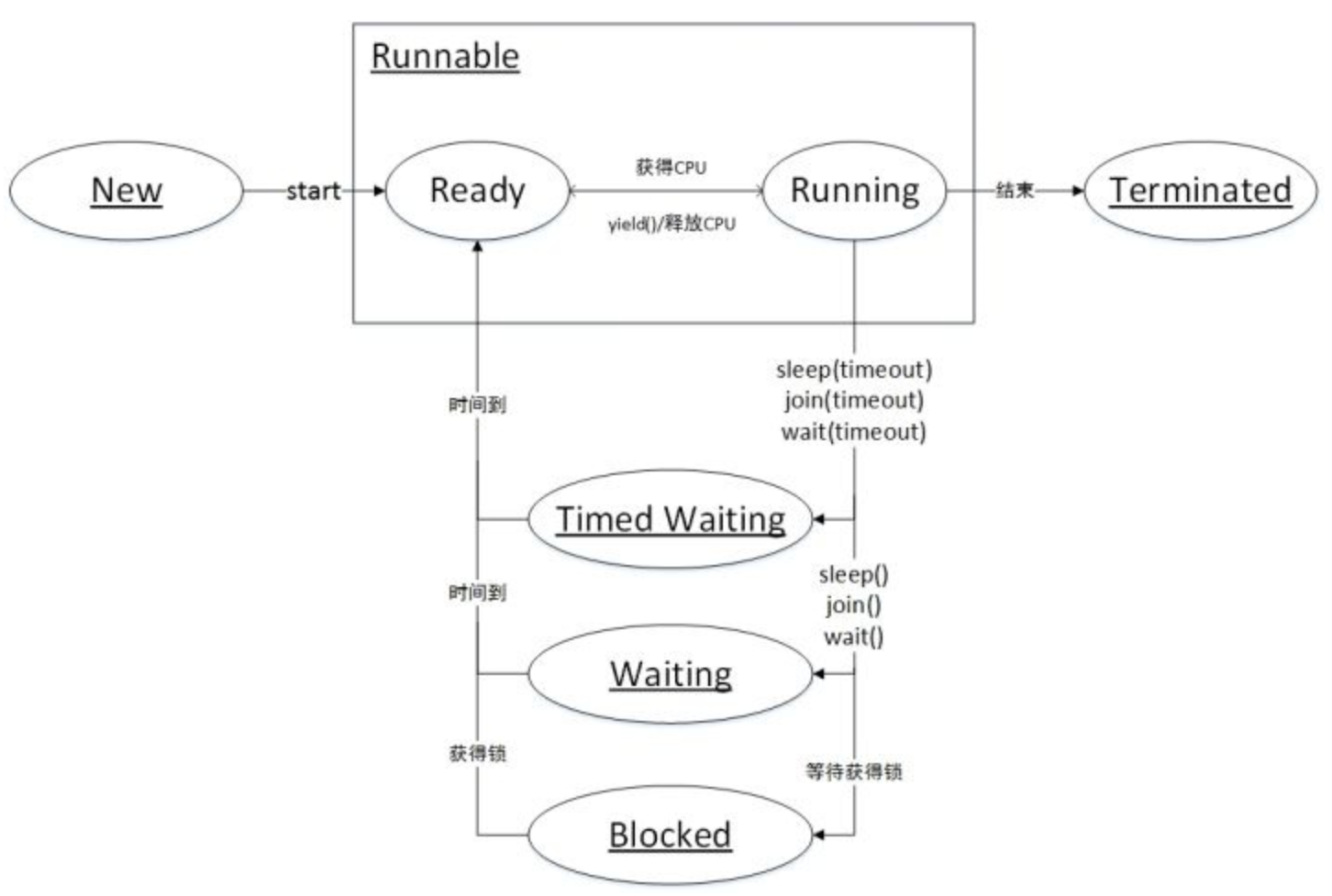
与WAITING状态相同，不过是有计时参数，这个状态会一直保持着知道计时期满或者接受到唤醒通知，带有计时参数的方法有Thread.sleep()，Object.wait()。

1. TERMINATED（被终止）

线程由于run方法正常退出而死亡，或者因为没有捕获的异常终止了run方法而死亡。

1. 线程状态图

以下是六个线程状态的示意图：



1. wait(), notify()

wait(), notify(), notifyAll()这三个方法是定义在Object类中的，用来控制线程的状态。wait()方法使持有该对象的线程将该对象的控制权交出去，然后进入WAITING状态；notify()方法通知某个处于正在等待这个对象的控制权的线程；notifyAll()方法会通知所有正在等待这个对象的控制权的线程。

注意，这三个方法都只能在线程同步中使用，并且wait()和notify()对应的都是同一个锁的资源，不同对象锁之间唤醒的话就会有线程安全问题。

1. wait()和sleep()区别
2. sleep()是Thread类的方法，wait()方法是Object类的方法
3. 调用sleep()方法时，线程不会释放对象锁；调用wait()方法时，线程会放弃该对象锁
4. sleep()方法导致了程序暂停执行指定的时间，让出CPU给其他的线程，当时间到了时就会恢复运行；wait()方法是将对象锁和资源都交了出去，该线程进入此对象的等待池中处于等待状态，只有针对此对象调用notify()方法或者notifyAll()方法时，该线程才能够争抢对象锁，如果抢到了，就进入运行状态。