自拟线程池总结：

简单说下线程的运行规则：

线程是进程的最小单位，线程是任务的载体，把任务带到cpu上执行相应的逻辑，一个cpu同一时刻只能运行一个线程，所以说真正意义上的多线程是在多核cup上才能实现的，别的线程只能等待cpu的时间片，等线程执行完或没有执行完用完时间片就会跳出来，人后其他线程就会抢占cpu的资源，切换上下文，系统就会变卡，所以线程太多也不是好事。

线程池实现原理：

线程池会在任务执行时批量生成多个线程，避免一个一个生成造成资源的浪费

几个要点：

任务仓库BlockingQueue<Runnable>，用于存储任务（阻塞队列）调用其task()拿任务，调用offer（Runnable）加入任务到队列

存储线程的集合用Collections.synchronizedList(new ArrayList<Thread>());存储

每个任务的执行方法：

public static class worker extends Thread{

private threadpooldemo threadpooldemo;

public worker(threadpooldemo threadpooldemo){

this.threadpooldemo = threadpooldemo;

}

@Override

public void run() {

while(this.threadpooldemo.BlockingQueue.size()>0){

//声明任务

Runnable take = null;

try {

//判断是否是在工作状态下，如果是则阻塞，不是则不阻塞（我理解不阻塞就是线程执行完就释放掉了）

if(this.threadpooldemo.isshutdown){

take = this.threadpooldemo.BlockingQueue.take();

}else{

take = this.threadpooldemo.BlockingQueue.poll();

}

} catch (InterruptedException e) {

e.printStackTrace();

}

if (take!=null){

take.run();

System.out.println("一个线程开始执行"+Thread.currentThread().getName());

}

}

}

}

关闭线程池

用public volatile boolean isshutdown = true;可见参数控制