任务列表分派给多个线程的策略和方法Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E4_BB_BB_E5_8A_A1_E5_88_97_E8_c104_644502.htm 多线程下载由来已久，如 FlashGet、NetAnts 等工具，它们都是依懒于 HTTP 协议的支持(Range 字段指定请求内容范围)，首先能读取出请求内容 (即欲下载的文件) 的大小，划分出若干区块，把区块分段分发给每个线程去下载，线程从本段起始处下载数据及至段尾，多个线程下载的内容最终会写入到同一个文件中。 只研究有用的，工作中的需求：要把多个任务分派给多个线程去执行，这其中就会有一个任务列表指派到线程的策略思考：已知：1. 一个待执行的任务列表，2. 指定要启动的线程数. 问题是：每个线程实际要执行哪些任务。 策略是：任务列表连续按线程数分段，先保证每线程平均能分配到的任务数，余下的任务从前至后依次附加到线程中--只是数量上，实际每个线程执行的任务都还是连续的。如果出现那种僧多(线程) 粥(任务) 少的情况，实际启动的线程数就等于任务数，一挑一。这里只实现了每个线程各扫自家门前雪，动作快的完成后眼见别的线程再累都是爱莫能助。 实现及演示代码如下：由三个类实现，写在了一个 java 文件中：TaskDistributor 为任务分发器，Task 为待执行的任务，WorkThread 为自定的工作线程。代码中运用了命令模式，如若能配以监听器，用上观察者模式来控制 UI 显示就更绝妙不过了，就能实现像下载中的区块着色跳跃的动感了，在此定义下一步的着眼点了。代码中有较为详细的注释，看这些注释和执行结果就很容易理解的。main() 是测试方法 package com.unmi.common. import

java.util.ArrayList. import java.util.List. public class TaskDistributor { /** * 测试方法 * @param args */ public static void main(String[] args) { //初始化要执行的任务列表 List taskList = new ArrayList(). for (int i = 0. i 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com