考试指导：java多线程设计模式详解之四 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022__E8_80_83_E8_AF_95_E6_8C_87_E5_c97_138816.htm ReadWriteLock 多线程读写同一个对象的数据是很普遍的，通常，要避免读写冲突，必须保证任何时候仅有一个线程在写入，有线程正在读取的时候，写入操作就必须等待。 简单说，就是要避免“ 写-写 ”冲突和“ 读-写 ”冲突。但是同时读是允许的，因为“ 读-读 ”不冲突，而且很安全。 要实现以上的ReadWriteLock，简单的使用synchronized就不行，我们必须自己设计一个ReadWriteLock类，在读之前，必须先获得“ 读锁 ”，写之前，必须先获得“ 写锁 ”。举例说明：DataHandler对象保存了一个可读写的char[]数组：package com.crackj2ee.thread.public class DataHandler { // store data: private char[] buffer = "AAAAAAAAAA".toCharArray(). private char[] doRead() { char[] ret = new char[buffer.length]. for(int i=0. i ret[i] = buffer[i]. sleep(3). } return ret. } private void doWrite(char[] data) { if(data!=null) { buffer = new char[data.length]. for(int i=0. i buffer[i] = data[i]. sleep(10). } } } private void sleep(int ms) { try { Thread.sleep(ms). } catch(InterruptedException ie) {} }}doRead()和doWrite()方法是非线程安全的读写方法。为了演示，加入了sleep()，并设置读的速度大约是写的3倍，这符合通常的情况。为了让多线程能安全读写，我们设计了一个ReadWriteLock：package com.crackj2ee.thread.public class ReadWriteLock { private int readingThreads = 0. private int writingThreads = 0. private int waitingThreads = 0. // waiting for

write private boolean preferWrite = true. public synchronized void readLock() throws InterruptedException { while(writingThreads>0 || (preferWrite amp. waitingThreads>0)) this.wait(). readingThreads . } public synchronized void readUnlock() { readingThreads--. preferWrite = true. notifyAll(). } public synchronized void writeLock() throws InterruptedException { waitingThreads . try { while(readingThreads>0 || writingThreads>0) this.wait(). } finally { waitingThreads--. } writingThreads . } public synchronized void writeUnlock() { writingThreads--.