**中软国际考题-笔试**

1. 单选题

1、启动多线程的方式是调用如下哪一个方法（ B ）

A.start B.do C.run D.sleep

2、Java提供了一个\_\_\_D\_\_\_\_\_\_\_线程，自动回收动态分配的内存。（D ）

A．异步 B．消费者 C．守护 D．垃圾收集

3、当▁方法终止时，能使线程进入死亡状态。（A ）

A．run B．setPrority//更改线程优先级

C．yield//暂停当前线程的执行 执行其他线程 D．sleep//线程休眠

4、线程通过▁▁方法可以使具有相同优先级线程获得处理器。（ C）

A．run B．setPrority C．yield D．sleep

5、哪个关键字可以对对象加互斥锁？  ( B )

   A．transient  B．synchronized  C．serialize  D．static

6. 以下哪个方法用于定义线程的执行体？( C   )

A、 start() B、init() C、run() D、synchronized()

7、线程通过（ D ）方法可以休眠一段时间，然后恢复运行。

A. run B. setPrority C. yield D. sleep

8、resume() 方法负责恢复哪些线程的执行？（B ）

1. 通过调用 wait() 方法而停止运行的线程
2. 通过调用 suspend() 方法而停止运行的线程
3. 通过调用 sleep() 方法而停止运行的线程
4. 通过调用 stop() 方法而停止运行的线程

9、运行在操作系统上的每个应用都会独占一个独立的？（ B）

A.线程 B.process C.thread D.Runnable

10、下面关于线程的叙述中，正确的是（C ）

A  不论是系统支持线程还是用户级线程，其切换都需要内核的支持

B  线程是资源的分配单位，进程是调度和分配的单位

C  不管系统中是否有线程，进程都是拥有资源的独立单位

D  在引入线程的系统中，进程仍是资源分配和调度分派的基本单位

11、下列哪个关键字可以用于实现同步?( C )

(A) Native (B) static (C) Synchronized (C) abstract

12、请问wait()方法在以下哪个代码中被调用？（C ）

A.一个while()循环体中 B.run方法中 C.同步化代码块中 D. 代码的任何地方

1. 多选题
2. 以下关于ThreadLocal的作用描述正确的是？（ ADE ）
3. ThreadLocal用来解决多线程程序的并发

B. 多线程类需要继承ThreadLocal类;

C. 多线程类需要实现ThreadLocal接口

1. ThreadLocal并不是一个Thread,而是Thread的局部变量
2. ThreadLocal为每个使用该变量的线程提供独立的变量副本
3. 在java中实现多线程的方式有哪几种？（ AD ）
4. 继承Thread类 B. 调用sleep方法等待; C. 继承Runnable类
5. 实现Runnable接口 E. 实现Thread接口

3、有三种原因可以导致线程不能运行，它们是▁。（ ABC ）

A．等待 B．阻塞 C．休眠 D．挂起及由于I/O操作而阻塞

4、下列关于Java多线程并发控制机制的叙述中，错误的是：( BC )

A、Java中对共享数据操作的并发控制是采用加锁技术

B、线程之间的交互，提倡采用suspend()/resume()方法

C、共享数据的访问权限都必须定义为private

D、Java中没有提供检测与避免死锁的专门机制，但应用程序员可以采用某些策略防止死锁的发生

5、关于sleep()和wait()，以下描述正确的是（ ABC ）

A. sleep是线程类（Thread）的方法，wait是Object类的方法；

B. sleep不释放对象锁，wait放弃对象锁；

C. sleep暂停线程、但监控状态仍然保持，结束后会自动恢复；

D. wait后进入等待锁定池，只有针对此对象发出notify方法后获得对象锁进入运行状态。

6、有关线程的哪些描述是正确的？（BC ）

A. 一旦一个线程被创建，它就立即开始运行

B. 使用 start() 方法可以使一个线程成为可运行的，但是它不一定立即开始运行

C. 一个线程可能因为不同的原因停止 (cease) 并进入就绪状态

D. 当一个线程因为抢先机制而停止运行时，它被放在可运行队列的前面

7、 以下哪几种是Java实现多线程的方式？（ AC ）

A. Thread类 B. Runnable类 C. Runnable接口

D. Thread接口

8、有三种原因可以导致线程不能运行，它们是 （ ABC ）

A．等待 B．阻塞 C．休眠 D．挂起及由于I/O操作而阻塞

9、 下列哪些方法可以启动线程？（AC ）

A. start()方法 B.go() C. run()方法

D. First（）方法

10、final、finally和finalize的区别（ABC ）

A. final是修饰符，可以修饰类、方法、常量

B. finally在异常处理时的操作，可以提供finally来执行清除操作

C. finalize是方法名，在垃圾收集器在对象从内存中清除之前做必要的清理工作

D. finally和finalize都是异常处理的方法

1. 填空题
2. 如果线程类实现了Runnable接口，则需要实现该接口的\_\_\_\_run\_\_\_\_\_\_ 方法，在启动线程时需要调用\_\_\_\_start\_\_\_\_\_方法。
3. Java中的线程有四种状态分别是：新建、\_\_运行\_\_\_\_\_\_\_\_ 、\_\_\_\_\_等待\_\_\_\_、结束。
4. 对处于可运行状态的多个线程对象进行系统级的协调，防止多个线程争用有限资源而导致系统\_\_\_\_崩溃\_\_\_\_\_\_。
5. java 的调度策略是基于线程的\_\_\_\_\_优先\_\_\_\_\_抢先式调度。意思就是，谁的优先级高那我就先给谁使用系统资源。
6. 创建线程时需要实现 Runnable 接口
7. 线程同步的关键字是：\_\_\_\_\_ synchronized \_\_\_\_\_ 。
8. Java中的线程有五种状态分别是：新建、\_\_\_\_\_\_\_可运行\_\_\_\_ 、\_\_\_\_\_运行 \_ 、\_\_\_\_\_\_等待\_\_\_\_ 、终止。
9. 线程同步与通讯有哪几种方式：信号量、\_\_\_ (互斥)锁机制\_\_\_\_\_\_\_ 、\_\_\_\_信号机制\_\_\_\_\_。
10. Java中的线程有四种状态分别是：新建、\_\_\_\_运行\_\_\_\_\_\_ 、\_\_\_等待\_\_\_\_\_\_、结束。判断题
11. ArrayList、Vector、Map等对象都是线程安全的。（ F ）
12. 在Java中，高优先级的可运行线程会抢占低优先级线程。（ T）
13. 线程使用sleep方法去休眠后可以使用notify方法唤醒（F）
14. 线程使用sleep方法休眠是会释放该线程同步锁定的对象。( F )
15. 线程可以用yield方法使低优先级的线程运行。（ F ）
16. 一个线程在调用它的 start() 方法之前，该线程将一直处于新建状态。（F ）
17. 一个线程可以调用 yield() 方法使其他线程有机会运行。（T ）
18. 如果线程终止，它便不能运行。（T ）
19. 多个程序同时运行使用的是时间片轮转算法。（ T ）
20. 当今的操作系统绝大部分是基于单任务的操作系统（F）
21. 线程池中的所有线程都是处于时时刻刻唤醒状态（ F）
22. 如果线程死亡，它便不能运行。（T）
23. Stop()方法是终止当前线程的一种状态( T )
24. 互斥条件：一个资源每次只能被一个进程使用。（ ）
25. 多线程是Java语言独有的。（ ）
26. 多线程需要多CPU。（ F ）

五、简答题

1、什么是线程池？ 为什么要使用它？

答案：

1、什么是线程池：  java.util.concurrent.Executors提供了一个 java.util.concurrent.Executor接口的实现用于创建线程池,多线程技术主要解决处理器单元内多个线程执行的问题，它可以显著减少处理器单元的闲置时间，增加处理器单元的吞吐能力。

2.线程池的作用：

线程池作用就是限制系统中执行线程的数量。  
    根据系统的环境情况，可以自动或手动设置线程数量，达到运行的最佳效果；少了浪费了系统资源，多了造成系统拥挤效率不高。用线程池控制线程数量，其他线程排队等候。一个任务执行完毕，再从队列的中取最前面的任务开始执行。若队列中没有等待进程，线程池的这一资源处于等待。当一个新任务需要运行时，如果线程池中有等待的工作线程，就可以开始运行了；否则进入等待队列。

2、什么是线程安全？

答案：

1. 线程安全的代码是多个线程同时执行也能工作的代码
2. 如果一段代码可以保证多个线程访问的时候正确操作共享数据，那么它是线程安全的
3. 当一个类被多个线程进行访问并且正确运行，它就是线程安全的

3、什么是线程？

 是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。线程有就绪、阻塞和运行三种基本状态。

1. Runnable接口和Callable接口的区别？

两者最大的不同点是：实现Callable接口的任务线程能返回执行结果；而实现Runnable接口的任务线程不能返回结果；

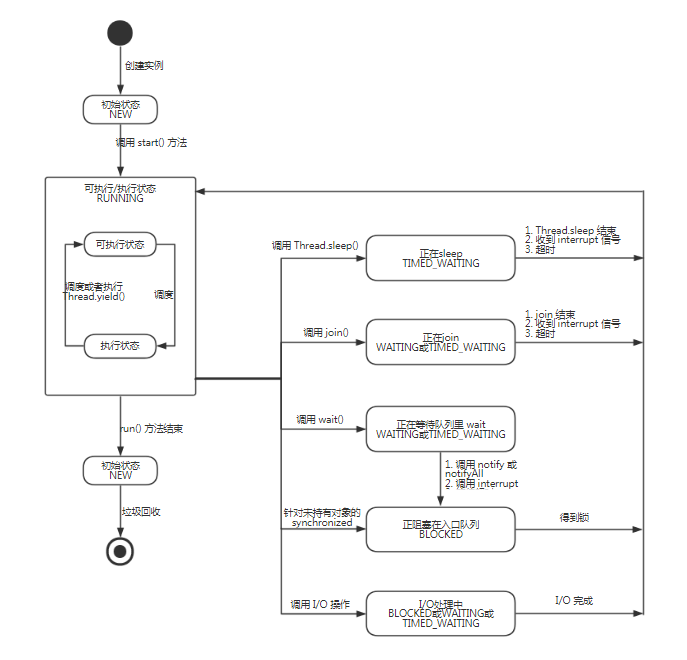
Callable接口的call()方法允许抛出异常；而Runnable接口的run()方法的异常只能在内部消化，不能继续上抛；

5. start()方法和run()方法的区别

线程的run()方法是由java虚拟机直接调用的，如果我们没有启动线程（没有调用线程的start()方法）而是在应用代码中直接调用run()方法，那么这个线程的run()方法其实运行在当前线程（即run()方法的调用方所在的线程）之中，而不是运行在其自身的线程中，从而违背了创建线程的初衷；

6、画图描述线程状态转换（5分）

答案：



7、试简述Thread类的子类或实现Runnable接口两种方法的异同？

答案：

1）定义一个类继承自Thread类，重写run方法，然后创建这个类的对象，然后通过对象调用start方法启动线程。

2）定义一个类实现Runnable接口，重写run方法，然后创建一个这个类的子类对象，然后建Thread类的对象，将子类对象作为参数进行传递，然后通过start方法启动线程

这两个方法的共同点就是子类重写run方法，但是继承是重写Thread中的run方法，实现接口是Runnable中的run方法。而且都需要start来启动线程，继续是通过其创建的子类对象来调用的，接口是将子类对象作为参数来传递给Thread，然后通过Thread的对象来调用start启动的。

9、 请简述ThreadLocal 的作用是什么？

答案： 为每一个使用变量的线程提供独立的变量副本，每个线程都可以修改自己的独立的线程副本，但是并不会影响其他的线程副本，从而保证了线程安全。变量副本存在于ThreadLocal的虚拟Map中。

1. 请简述 sleep() 方法和 yield() 方法的区别。

sleep方法给其他线程机会时不考虑线程优先级(优先级低的也有可能)；而yield方法只会给相同优先级或者更高优先级线程机会(如果没有相同或者更高优先级的线程，该线程会继续运行)

线程执行sleep方法进入阻塞状态，执行yield方法进入就绪状态

12、试说明在Java中 wait() 和 sleep() 方法的不同。

答案：

sleep方法是Thread的静态方法，wait方法是Object类的普通方法

sleep方法不释放同步锁，wait方法释放同步锁(执行notify方法唤醒wait的线程时是不释放同步锁的)

wait方法用于线程间通信，而sleep方法用于短暂的暂停线程

sleep针对当前线程，而wait针对被同步代码块加锁的对象

sleep方法是当前线程暂停指定时间，将执行机会让给其它线程，时间结束后进入就绪状态等待

调用wait方法会暂停线程，当前线程释放对象的同步锁，进入等待池(wait pool)，只有调用对象的notify或者notifyAll方法唤醒时，线程进入等锁池(lock pool)，直到线程再次获得对象的锁才会进入就绪状态

wait方法(notify，notifyAll)只能在同步方法或者同步块中使用(如果在non-synchronized函数或non-synchronizedblock中进行调用，虽然能编译通过，但在运行时会发生illegalMonitorStateException的异常)；

sleep方法可以在任意位置使用

14、线程和进程有什么区别？

（1）线程是CPU独立运行和独立调度的基本单位；  
（2）进程是资源分配的基本单位；  
两者的联系：进程和线程都是操作系统所运行的程序运行的基本单元。  
区别：  
（1）进程具有独立的空间地址，一个进程崩溃后，在保护模式下不会对其它进程产生影响。  
（2）线程只是一个进程的不同执行路径，线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉。

15、如何在Java中实现线程？

1）定义一个类继承自Thread类，重写run方法，然后创建这个类的对象，然后通过对象调用start方法启动线程。

2）定义一个类实现Runnable接口，重写run方法，然后创建一个这个类的子类对象，然后建Thread类的对象，将子类对象作为参数进行传递，然后通过start方法启动线程

17、什么是线程安全？Vector是一个线程安全类吗？

1. 线程安全的代码是多个线程同时执行也能工作的代码
2. 如果一段代码可以保证多个线程访问的时候正确操作共享数据，那么它是线程安全的
3. 当一个类被多个线程进行访问并且正确运行，它就是线程安全的

Vector是一个线程安全的类，其在add()等操作上添加了synchronized关键字实现同步，但是并非是绝对的线程安全类． 当进行迭代遍历时，如果在另一个线程执行add(),remove()操作，仍然会有机率抛出异常ConcurrentModificationException．

18、Java中notify 和 notifyAll有什么区别？

当有线程调用了对象的 notifyAll()方法（唤醒所有 wait 线程）或 notify()方法（只随机唤醒一个 wait 线程），被唤醒的的线程便会进入该对象的锁池中，锁池中的线程会去竞争该对象锁。也就是说，调用了notify后只要一个线程会由等待池进入锁池，而notifyAll会将该对象等待池内的所有线程移动到锁池中，等待锁竞争

20、有三个线程T1，T2，T3，怎么确保它们按顺序执行

第一种方式：顺序在线程中创建实例（最容易想到的办法）。

第二种方式：看到有人说运用单个线程池（SingleThreadExecutor）来实现，确切的说这里不太符合，其实我是在一个线程里，执行了三个任务。

第三种方式：运用线程的  join   方法来实现, 来保证上一个线程执行完后，再执行现在的线程执行

26. 在Java中Lock接口比synchronized块的优势是什么？

* 同步块被完全包含在方法中-我们可以有Lock API的lock() 和unlock() 在单独的方法操作
* synchronized块不支持公平性，任何线程一旦释放就可以获得锁定，不能指定任何偏好。我们可以通过指定公平属性来实现Lock API中的公平性。它确保最长的等待线程可以访问锁
* 如果线程无法访问synchronized 块，则会阻塞该线程。所述锁 API提供的tryLock（）方法。线程只有在可用且不被任何其他线程保持时才获取锁定。这减少了线程等待锁定的阻塞时间
* 处于 “waiting” 状态以获取对同步块的访问的线程不能被中断。所述锁 API提供了一种方法的lockInterruptibly（） ，处于"waiting"时其可以用来中断线程

28.volatile关键字是否能保证线程安全？

volatile保证不了线程安全。想要线程安全必须保证原子性，可见性，有序性。而volatile只能保证可见性和有序性

六、程序实现题

1、信号量问题：有时被称为信号灯，是在多线程环境下使用的一种设施, 它负责协调各个线程, 以保证它们能够正确、合理的使用公共资源，假设初始化公共资源的数量为n，当需要获取公共资源时需要申请，当剩余可以使用的公共资源时，则分发，如当前无可使用的公共资源则等待。若某一个线程使用公共资源后可以释放，并交由其他线程使用。

（10分）

|  |
| --- |
| package test;  import java.util.Random;  import java.util.Vector;  public class Semaphore {  private Vector<Object> locks = new Vector<Object>();  private int permitNum = 1;  private int nowPermitNum = 1;  private boolean permitNumGrow = false;  private boolean fair = false;  Random random = new Random();  public Semaphore(int permitNum, boolean permitNumGrow, boolean fair) {  this.permitNum = permitNum;  this.nowPermitNum = permitNum;  this.permitNumGrow = permitNumGrow;  this.fair = fair;  }  public Semaphore(int permitNum) {  this(permitNum, true, false);  }  public Semaphore() {  this(1);  }  public void acquire() {  Object lock = new Object();  synchronized (lock) {  if (nowPermitNum > 0) {  nowPermitNum--;  } else {  locks.add(lock);  try {  lock.wait();  } catch (Exception ex) {  ex.printStackTrace();  }  }  }  }  public void release() {  if (locks.size() > 0) {  int index = 0;  if (!fair) {  index = Math.abs(random.nextInt()) % locks.size();  }  Object lock = locks.get(index);  locks.remove(lock);  synchronized (lock) {  lock.notify();  }  } else if (nowPermitNum < permitNum || permitNumGrow) {  nowPermitNum++;  }  }  public int getAvailablePermitNum() {  return nowPermitNum;  }  } |

2、有一个抽奖池,该抽奖池中存放了奖励的金额,该抽奖池用一个数组int[] arr = {10,5,20,50,100,200,500,800,2,80,300};

创建两个抽奖箱(线程)设置线程名称分别为“抽奖箱1”，“抽奖箱2”，随机从arr数组中获取奖项元素并打印在控制台上,格式如下:

抽奖箱1 又产生了一个 10 元大奖

抽奖箱2 又产生了一个 100 元大奖

（10分）

|  |
| --- |
| public class ChoujiangDemo {  public static void main(String[] args) {  Chou c = new Chou();  Thread t1 = new Thread(c,"抽奖箱1");  Thread t2 = new Thread(c,"抽奖箱2");  t1.start();  t2.start();  }  }  public class Chou implements Runnable {  int[] arr = { 10, 5, 20, 50, 100, 200, 500, 800, 2, 80, 300 };  int num = arr.length;  boolean[] flag = new boolean[arr.length];  public void run() {  while (true) {  synchronized (this) {  if (num > 0) {  int index = (int) (Math.random() \* arr.length);  int get = arr[index];  // 代表这张抽奖券抽过了  if (flag[index] != true) {  flag[index] = true;  System.out.println(Thread.currentThread().getName()  + " 又产生了一个" + get + "元大奖");  num--;  }  }  }  }  }  } |

3、多线程实现多个窗口同时卖票

4、[编写多线程应用程序，模拟多个人通过一个山洞的模拟。这个山洞每次只能通过一个人，每个人通过山洞的时间为5秒，随机生成10个人，同时准备过此山洞，显示一下每次通过山洞人的姓名。](http://www.cnblogs.com/wenwen123/p/5740579.html)

（10分）

|  |
| --- |
| public class ShanDong extends Thread{  @Override  public void run() {  synchronized (this) {  System.out.println(Thread.currentThread().getName()+"正在过山洞,请等待5秒");  try {  Thread.sleep(5000);  } catch (InterruptedException e) {  e.printStackTrace();  }  }    }    public static void main(String[] args) {  String[] peoples = {"赵","钱","孙","李","周","吴","郑","王","冯","陈"};  ShanDong sd = new ShanDong();  for(String people:peoples){  Thread t = new Thread(sd,people);  t.start();  }    }  } |

5、请编写多线程的程序模拟三个售票窗口同时出售20张票。

**Station.java**

|  |
| --- |
| **public** **class** Station **extends** Thread {  // 通过构造方法给线程名字赋值  **public** Station(String name) {  **super**(name);// 给线程名字赋值  }  // 为了保持票数的一致，票数要静态  **static** **int** *ticketCount* = 20;  // 创建一个静态钥匙  **static** Object *objLock* = "ticketLock";// 值是任意的  // 重写run方法，实现买票操作  @Override  **public** **void** run() {  **while** (*ticketCount* > 0) {  **synchronized** (*objLock*) {// 这个很重要，必须使用一个锁，  // 进去的人会把钥匙拿在手上，出来后才把钥匙拿让出来  **if** (*ticketCount* > 0) {  System.***out***.println(getName() + "卖出了第" + *ticketCount* + "张票");  *ticketCount*--;  } **else** {  System.***out***.println("票卖完了");  }  }  **try** {  *sleep*(1000);// 休息一秒  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }  } |

**MainApp.java**

|  |
| --- |
| **public** **class** MainApp {  /\*\*  \* java多线程同步锁的使用 示例：三个售票窗口同时出售20张票  \*/  **public** **static** **void** main(String[] args) {  // 实例化站台对象，并为每一个站台取名字  Station station1 = **new** Station("窗口1");  Station station2 = **new** Station("窗口2");  Station station3 = **new** Station("窗口3");  // 让每一个站台对象各自开始工作  station1.start();  station2.start();  station3.start();  }  } |

**参考运行结果：**

**6、编写线程打印5次，每次睡眠1秒**

（10分）

|  |
| --- |
|  |

7、写两个线程，一个线程打印1~52，另一个线程打印A~Z，打印顺序是12A34B...5152Z；

（10分）

|  |
| --- |
| public class TwoThread {    public static void main(String args[]){  MyObject1 my = new MyObject1();  new Thread(new Runnable(){  @Override  public void run() {  // TODO Auto-generated method stub  for(int i = 0; i < 26; i++){  my.printNum();  }  }    }).start();  new Thread(new Runnable(){  @Override  public void run() {  // TODO Auto-generated method stub  for(int i = 0; i < 26; i++){  my.printA();  }  }    }).start();  }  }class MyObject1{  private static boolean flag = true ;  public int count = 1;    public synchronized void printNum(){  while(flag == false){  try {  this.wait();  } catch (InterruptedException e) {  // TODO Auto-generated catch block e.printStackTrace();  }  }  System.out.print((2\*count-1));  System.out.print(2\*count);    flag = false;  this.notify();  }  public synchronized void printA(){  while(flag == true){  try {  this.wait();  } catch (InterruptedException e) {  // TODO Auto-generated catch block e.printStackTrace();  }  }  System.out.print((char)(count+'A'-1));  count++;  flag = true;  this.notify();  }  }  [IMG_256](https://www.cnblogs.com/zhihuayun/p/javascript:void(0);) |

8、子线程循环10次，主线程循环100次，然后子线程循环10次，主线程循环100次，这样循环50次；

|  |
| --- |
|  |

9、需求说明：定义一个线程，输出1 ～ 200之间的奇数，定义另一个线程，逆序输出1 ～ 200之间的奇数。要求：采用Thread类和Runnable接口两种方式实现，要求线程A和线程B交替，使用sleep()方法阻塞当前线程，每打印一次后休息1秒。

|  |
| --- |
|  |