|  |  |  |
| --- | --- | --- |
| **1.** | **下列关于Java线程的说法正确的是（   A ）。（选择一项）** | |
|  |  |  |
|  | **A** | 每一个Java线程可以看成由代码、一个真实的CPU以及数据三部分组成 |
|  | **B.** | 创建线程的两种方法中，从Thread类中继承方式可以防止出现多父类的问题 |
|  | **C.** | Thread类属于java.util程序包 |
|  | **D.** | 使用new Thread(new X()).run();方法启动一个线程  解析：       A、语句正确。       B、创建线程两种方法：一是继承Thread类，一是实现Runnable接口          根据Java语言类的单继承特性，防止出现多继承，但是可以实现多          重继承，也可以出现多父类。  C、Thread类是存在于java.lang包下       D、启动一个线程是调用start()方法，之后JVM会默认的调用run()方法          run()方法是线程的主体，核心代码放到此方法当中。 |

|  |  |  |
| --- | --- | --- |
| **2.** | **以下选项中可以填写到横线处，让代码正确编译和运行的是（  AD  ）。（选择一项）** | |
|  | **public** **class** Test **implements** Runnable {  **public** **static** **void** main(String[] args) {                     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_                     t.start();                     System.*out*.println("main");           }  **public** **void** run() {                     System.*out*.println("thread1!");           }  } | |
|  |  |  |
|  | **A.** | Thread t = new Thread(new Test()); |
|  | **B.** | Test t = new Test(); |
|  | **C.** | Thread t = new Test(); |
|  | **D.** | Thread t = new Thread();  解析：    线程的创建有两种方式，其中之一就是实现Runnable接口。    此接口只有一个抽象方法---run()方法，启动线程又需要start()方法    此时还需要借助Thread类，根据Thread类的构造方法：  Public  Thread(Runnable  run)  需要传入Runnable接口对象   此题还有一个方法： 直接创建Thread类对象，调用start()方法  所以D选项也是正确的，但是此题重点还是考的子类实现Runnable接口  根据Thread类的构造方法启动线程。 |

|  |  |  |
| --- | --- | --- |
| **3.** | **如下代码创建一个新线程并启动线程，问:四个选项中可以保证正确代码创建target对象，并能编译正确的是（   C  ）？（选择一项）** | |
|  | **public** **static** **void** main(String[] args) {            Runnable target=**new** MyRunnable( );            Thread myThread=**new** Thread(target);  } | |
|  |  |  |
|  | **A** | **public** **class** MyRunnable **extends** Runnable {  **public** **void** run( ) {        }  } |
|  | **B.** | **public** **class** MyRunnable **extends** Runnable {  **void** run( ) {  }  } |
|  | **C.** | **public** **class** MyRunnable  **implements** Runnable  {  **public** **void** run( ) {        }  } |
|  | **D.** | **public** **class** MyRunnable  **implements** Runnable  {  **void** run( ) { }  解析：接口是用来实现的，不是继承，所以AB两个选项错误。        根据Java多态机制，子类可以为父类（接口）实例化对象        根据子类实现接口的规则，子类必须重写接口当中的抽象方法        此时又复习到了方法重写的规则：子类重写的方法它的权限必须        大于等于父类方法权限（除去private外）。根据以上得出C是对的 |

|  |  |  |
| --- | --- | --- |
| **4.** | **当线程调用start( )后，其所处状态为（ C   ）。（选择一项）** | |
|  |  |  |
|  | **A** | 阻塞状态 |
|  | **B.** | 运行状态 |
|  | **C.** | 就绪状态 |
|  | **D.** | 新建状态  解析：线程的五个状态也就是它的生命周期：  新建状态----通过new关键字来创建线程对象  就绪状态----调用start()方法  运行状态----调用run()方法  阻塞状态----调用sleep()、wait()、join() 、yield()、interrupt ()等方法  消亡状态----调用stop()方法，但是此方法已经过时。 |

|  |  |  |
| --- | --- | --- |
| **5.** | **下列关于Thread类提供的线程控制方法的说法中，错误的是（  C  ）。（选择一项）** | |
|  |  |  |
|  | **A** | 线程A中执行线程B的join()方法，则线程A等待直到B执行完成 |
|  | **B.** | 线程A通过调用interrupt()方法来中断其阻塞状态 |
|  | **C.** | 若线程A调用方法isAlive()返回值为false，则说明A正在执行中，也可能是可运行状态 |
|  | **D.** | currentThread()方法返回当前线程的引用  解析：  A： join()方法—强制加入，加入的线程执行完毕之后才能执行其他线程  B：interrupt()方法---线程中断，题中的说法有些牵强  C：isAlive()方法---判断一个线程是否在活动，如果在活动返回真，反之假  D：currentThread()方法返回正在执行的线程  从以上得知，C明显错误。 |

|  |  |  |
| --- | --- | --- |
| **6.** | **下列关于线程的优先级说法中，正确的是（  BD  ）。（选择两项）** | |
|  |  |  |
|  | **A** | 线程的优先级是不能改变的 |
|  | **B.** | 线程的优先级是在创建线程时设置的 |
|  | **C.** | 在创建线程后的任何时候都可以重新设置 |
|  | **D.** | 线程的优先级的范围在1-10之间 |

**解析：线程的优先级可以通过setPriority**(int newPriority)**的方法进行设置**

**线程一共有三个优先级，分别是：最低优先级（1） 中等优先级（5）**

**最高优先级（10），咱们经常写的main方法就是中等优先级线程。**

**A：线程的优先级是可以更改也可以获取，但是有一点，即使设置为最高**

**优先级也不一定先执行，只是它优先执行的几率比较高。**

**B：线程的优先级是在创建时进行设置，通过setPriority()方法设置**

**C：正在执行的线程是不允许重新设置线程优先级的。**

**D：线程的优先级范围是1—10 符合要求**

|  |  |  |
| --- | --- | --- |
| **7.** | **以下选项中关于Java中线程控制方法的说法正确的是（  AD   ）。（选择二项）** | |
|  |  |  |
|  | **A.** | join ( ) 的作用是阻塞指定线程等到另一个线程完成以后再继续执行 |
|  | **B.** | sleep ( ) 的作用是让当前正在执行线程暂停，线程将转入就绪状态 |
|  | **C.** | yield ( ) 的作用是使线程停止运行一段时间，将处于阻塞状态 |
|  | **D.** | setDaemon( )的作用是将指定的线程设置成后台线程  解析：  A：join()方法—线程的强制加入，加入的线程执行完毕之后再执行其他线程  B：sleep()方法—线程休眠，等时间过时，线程处于运行状态  C：yield()方法—线程礼让，让出CUP资源，其他线程先执行，有些与join()                  方法类似  D：setDaemon()方法--将指定的线程设置成后台线程，对的。  题的答案是：AD，但是个人感觉C选项也是对的…… |

|  |  |  |
| --- | --- | --- |
| **8.** | **在多个线程访问同一个资源时，可以使用（  A  ）关键字来实现线程同步，保证对资源安全访问。（选择一项）** | |
|  |  |  |
|  | **A.** | Synchronized |
|  | **B.** | Transient |
|  | **C.** | Static |
|  | **D.** | Yield |

**解析：关于同步，有两种实现方式，一种是同步方法，一种是同步代码块**

**无论怎样，都需要使用到**synchronized关键字。

                同步方法：

                     权限修饰符  synchronized  返回值类型  方法名称(参数列表){

                     N行代码;

}

            同步代码块：

                 Synchronized（对象）{

                     N行代码;

}

|  |  |  |
| --- | --- | --- |
| **9.** | **Java中线程安全问题是通过关键字（   C  ）解决的?。（选择一项）** | |
|  |  |  |
|  | **A.** | Finally |
|  | **B.** | wait( ) |
|  | **C.** | Synchronized |
|  | **D.** | notify( ) |
| **10.** | **以下说法中关于线程通信的说法错误的是（  D  ）?。（选择一项）** | |
|  |  |  |
|  | **A.** | 可以调用wait()、notify()、notifyAll()三个方法实现线程通信 |
|  | **B.** | wait()、notify()、notifyAll()必须在synchronized方法或者代码块中使用 |
|  | **C.** | wait()有多个重载的方法，可以指定等待的时间 |
|  | **D.** | wait()、notify()、notifyAll()是Object类提供的方法，子类可以重写  解析：  A选项：  在线程通信中，可以调用wait()、notify()、notifyAll()三个方法实现线程通信，这三个方法都是Object类提供的public方法，所以任何类都具有这三个方法。  B选项：在编程题的第二题当中，wait()方法、notify()方法、notifyAll()方法都是写在同步方法当中，具体可以查看此类的源码。  C选项：Object类当中的wait方法的重载如下：  public final void wait()  public final void wait(long timeout)  public final void wait(long timeout, int nanos)  三个方法都抛出异常-- InterruptedException     D选项 ： 看A选项的解释      以上三个方法都使用final修饰，所以子类是不能重写的 |

**三、****判断题**

1. 进程是线程Thread内部的一个执行单元，它是程序中一个单一顺序控制流程。( ×  )

解析：线程是进程的一个执行单位，进程包含线程。

1. Thread类实现了Runnable接口。(   √  )

解析：Thread类在JDK当中的定义如下：

         Public class Thread extends Object  implements Runnable

        从定义可知，它是Runnable接口的子类。

1. 一个进程可以包括多个线程。两者的一个主要区别是：线程是资源分配的单位，而进程CPU调度和执行的单位。（  ×  ）

解析：不管是进程还是线程，都是通过循环获得自己执行的时间片，获得CUP资源。

1. 用new关键字建立一个线程对象后，该线程对象就处于新生状态。处于新生状态的线程有自己的内存空间，通过调用start进入就绪状态。（  √  ）

解析：具体解析看选择题第四题的解释。

1. A线程的优先级是10，B线程的优先级是1，那么当进行调度时一定会先调用A（ ×  ）

解析：线程的执行，不是谁的优先级高就先执行，只是它的概率会高。

1. 线程可以用yield使低优先级的线程运行。（  ×  ）

解析：yield()方法是线程礼让，可以让出自己执行其他线程，由于线程的执行存在严重

     的随机性，不能确定使低优先级的线程执行。

1. Thread.sleep( )方法调用后，当等待时间未到，该线程所处状态为阻塞状态。当等待时间已到，该线程所处状态为运行状态。（ √   ）

解析：sleep()方法是线程休眠，处于阻塞状态，当时间一到，就会执行处于运行状态。

      如果真要追究用词的严谨性，应该不是”等待时间”，而是”休眠时间”。

1. 当一个线程进入一个对象的一个synchronized方法后，其它线程不可以再进入该对象同步的其它方法执行。（  √  ）

解析：对的，可以查看一下编程题第二题的源码。

1. wait方法被调用时，所在线程是会释放所持有的锁资源， sleep方法不会释放。（ √  ）

解析：sleep 是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复，所以调用sleep 不会释放对象锁。  
wait 是Object 类的方法，对此对象调用wait 方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出notify 方法（或notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

1. wait、notify、notifyAll是在Object类中定义的方法。（ √  ）

解析：具体可以查看API帮助文档，查看。

1. notify是唤醒所在对象wait pool中的第一个线程。（ ×  ）

解析：唤醒的不一定是第一个线程。

      JDK已经明确说明唤醒哪个thread是随意决定的，没有特定顺序  
The choice is arbitrary and occurs at the discretion of the implementation