**多线程作业**

1. **填空题**
2. 处于运行状态的线程在某些情况下，如执行了sleep（睡眠）方法，或等待I/O设备等资源，将让出CPU并暂时停止自己的运行，进入\_\_阻塞\_\_\_\_\_\_\_状态。
3. 处于新建状态的线程被启动后，将进入线程队列排队等待CPU，此时它已具备了运行条件，一旦轮到享用CPU资源就可以获得执行机会。上述线程是处于 就绪 状态。
4. 一个正在执行的线程可能被人为地中断，让出CPU的使用权，暂时中止自己的执行，进入 阻塞 状态。
5. 在Java中编写实现多线程应用有两种途径：一种是继承Thread类创建线程，另一种是实现 Runnable 接口创建线程。
6. 在线程控制中，可以调用\_\_\_\_\_join\_\_\_\_\_\_\_\_\_方法，阻塞当前正在执行的线程，等插队线程执行完后后再执行阻塞线程。
7. 多线程访问某个共享资源可能出现线程安全问题，此时可以使用\_\_\_\_Synchronized\_\_\_\_\_\_\_\_\_关键字来实现线程同步，从而避免安全问题出现，但会影响性能，甚至出现死锁。
8. 在线程通信中，调用wait( )可以是当前线程处于等待状态，而为了唤醒一个等待的线程，需要调用的方法是\_\_\_\_\_\_\_notity()\_\_\_\_\_\_\_。
9. 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。
10. **选择题**

|  |  |  |
| --- | --- | --- |
| **1.** | **下列关于Java线程的说法正确的是（ B ）。（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 |

|  |  |  |
| --- | --- | --- |
| **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（ A ）。（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(); |

|  |  |  |
| --- | --- | --- |
| **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( ) {  }  } |

|  |  |  |
| --- | --- | --- |
| **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 |

|  |  |  |
| --- | --- | --- |
| **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 |

|  |  |  |
| --- | --- | --- |
| **6.** | **下列关于线程的优先级说法中，正确的是（ C ）。（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1-100之间 |

|  |  |  |
| --- | --- | --- |
| **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 |

|  |  |  |
| --- | --- | --- |
| **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 |

|  |  |  |
| --- | --- | --- |
| **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（ B ）?。（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 |

1. **判断题**
2. 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。( f )
3. 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。（ f ）
4. 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。（ t ）
5. 线程可以用yield使低优先级的线程运行。（ t ）
6. 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。（ f ）
7. 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。（ f ）
8. **简答题**
9. 简述进程和线程的联系和区别。

进程作为资源分配的单位，系统在运行时会为每个进程分配不同的内存区域。

线程作为调度和执行的单位，每个线程拥有独立的运行栈和程序计数器(pc)，线程切换的开销小。

1. 创建线程的两种方式分别是什么？各有什么优缺点。

继承Thread: 线程代码存放Thread子类run方法中。

实现Runnable：线程代码存在接口的子类的run方法。

实现方法的好处：

1）避免了单继承的局限性；

2）多个线程可以共享同一个接口子类的对象，非常适合多个相同线程来处理同一份资源；

1. 请你简述sleep( )和wait( )有什么区别？

Sleep()方法休眠时间到了，自动将阻塞转为就绪，wait()方法让当前线程等待，直到其他线程调用了该锁对象的notify()或者notifyAll()方法为止，否则将一直等待。

1. Java中实现线程通信的三个方法及其作用。

Wait()方法：释放锁，当前线程等待

Notity()方法：唤醒当前锁对象的等待的单个线程

Notityall()方法：唤醒当前锁对象的所有等待线程

1. **编码题**
2. 设计一个多线程的程序如下：设计一个火车售票模拟程序。假如火车站要有100张火车票要卖出，现在有5个售票点同时售票，用5个线程模拟这5个售票点的售票情况。

**package** com.uek.test;

**public** **class** **BuyTest** {

**public** **static** **void** **main**(**String**[] args) {

**Piao** piao = **new** Piao();

**Thread** tA = **new** Thread(piao);

**Thread** tB = **new** Thread(piao);

**Thread** tC = **new** Thread(piao);

**Thread** tD= **new** Thread(piao);

**Thread** tE = **new** Thread(piao);

tA.start();

tB.start();

tC.start();

tD.start();

tE.start();

}

}

**class** **Piao** **implements** Runnable {

**int** piao = 100;

**boolean** flag = **true**;

@Override

**public** **void** **run**() {

**while** (flag) {

//第一种方法

/\*

synchronized (this) {

if (piao <= 0) {

System.out.println("买完了");

break;

}

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName() + "," + piao);

piao--;

\*/

//第二种

buy();

}

}

**public** **synchronized** **void** **buy**() {

**if**(piao<=0) {

**System**.***out***.println("票买完了！");

flag = **false**;

**return**;

}

**System**.***out***.println(**Thread**.*currentThread*().getName()+","+piao);

piao--;

}

}

1. 编写两个线程,一个线程打印1-52的整数，另一个线程打印字母A-Z。打印顺序为12A34B56C….5152Z。即按照整数和字母的顺序从小到大打印，并且每打印两个整数后，打印一个字母，交替循环打印，直到打印到整数52和字母Z结束。

要求：

1. 编写打印类Printer，声明私有属性index，初始值为1，用来表示是第几次打印。
2. 在打印类Printer中编写打印数字的方法print(int i)，3的倍数就使用wait()方法等待，否则就输出i，使用notifyAll()进行唤醒其它线程。
3. 在打印类Printer中编写打印字母的方法print(char c)，不是3的倍数就等待，否则就打印输出字母c，使用notifyAll()进行唤醒其它线程。
4. 编写打印数字的线程NumberPrinter继承Thread类，声明私有属性private Printer p;在构造方法中进行赋值，实现父类的run方法，调用Printer类中的输出数字的方法。
5. 编写打印字母的线程LetterPrinter继承Thread类，声明私有属性private Printer p;在构造方法中进行赋值，实现父类的run方法，调用Printer类中的输出字母的方法。
6. 编写测试类Test，创建打印类对象，创建两个线程类对象，启动线程。

**package** com.uek.test;

**public** **class** **TwoTest** {

**public** **static** **void** **main**(**String**[] args) {

**Printer** p = **new** Printer();

**NumberPrinte** n = **new** NumberPrinte(p);

**LetterPrinter** l = **new** LetterPrinter(p);

n.start();

l.start();

}

}

// 打印类

**class** **Printer** {

**private** **int** index = 1;

// 打印数字

**public** **synchronized** **void** **print**(**int** i) {

**for** (**int** j = 1; j <= 52; j++) {

**System**.***out***.println(j);

**System**.***out***.println("第" + (index++) + "次打印");

**if** (j % 2 == 0) {

//唤醒其他线程

**this**.notify();

**try** {

//释放锁，阻塞

**this**.wait();

} **catch** (**InterruptedException** e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

// 打印字母

**public** **synchronized** **void** **print**(**char** c) {

**for** (**int** j = 0; j < c; j++) {

**System**.***out***.println((**char**) ('A' + j));

**System**.***out***.println("第" + (index++) + "打印");

//唤醒其他进程

**this**.notify();

**try** {

//阻塞自己进程

**this**.wait();

} **catch** (**InterruptedException** e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

**class** **NumberPrinte** **extends** **Thread** {

**private** **Printer** p;

**public** **NumberPrinte**() {

}

**public** **NumberPrinte**(**Printer** p) {

**super**();

**this**.p = p;

}

@Override

**public** **void** **run**() {

**synchronized** (**this**) {

p.print(52);

}

}

}

**class** **LetterPrinter** **extends** **Thread** {

**private** **Printer** p;

**public** **LetterPrinter**() {

}

**public** **LetterPrinter**(**Printer** p) {

**super**();

**this**.p = p;

}

@Override

**public** **void** **run**() {

p.print('Z');

}

}

1. **可选题**
2. 设计4个线程，其中两个线程每次对j增加1，另外两个线程对j每次减少1。

要求：使用内部类实现线程，对j增减的时候不考虑顺序问题。

1. 编写多线程程序，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，有10个人同时准备过此山洞，显示每次通过山洞人的姓名和顺序。