**多线程应用练习**

1. **写出饿汉式单例模式、懒汉式单例模式、多线程情况下懒汉式双重判断单例模式的示例代码。（7分）**

/\*\*

\* 饿汉式的单例模式，在类加载时创建单例实例，而不是等到第一次请求实例时创建

\* 1、私有 的无参数构造方法Singleton()，避免外部创建实例

\* 2、私有静态属性instance \* 3、公有静态方法getInstance()

\*/

**public** **class** Singleton {

**private** **static** Singleton *instance* = **new** Singleton();

**private** Singleton(){ }

**public** **static** Singleton getInstance(){

**return** *instance*;

}

}

/\*\*

\* 懒汉式的单例模式

\*在类加载的时候不创建单例实例，只有在第一次请求实例的时候的时候创建

\*/

**public** **class** Singleton {

**private** **static** Singleton *instance*;

**private** Singleton(){ }

**public** **static** Singleton getInstance(){

**if**(*instance*==**null**){ //只创建一次对象

*instance* = **new** Singleton();

}

**return** *instance*;

}

}

/\*\*

\*多线程情况下懒汉式的单例模式，避免创建多个对象

\* 需要使用**synchronized关键字和**进行两次null判断

\*/

**public** **class** Singleton {

**private** **static** Singleton *instance*;

**private** Singleton(){ }

**public** **static** Singleton getInstance(){

**if**(*instance* ==**null**){//避免每次加锁，只有第一次没有创建对象时才加锁

**synchronized**(Singleton.**class**){//加锁，只允许一个线程进入

**if**(*instance*==**null**){ //只创建一次对象

*instance* = **new** Singleton();

}

}

}

**return** *instance*;

}

}

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（8分）**

推荐步骤：

1. 编写打印类Printer，声明私有属性index，初始值1，表示第几次打印。（1分）
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。（1.5分）
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。（1.5分）
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。（0.5分）实现父类的run方法，调用Printer类中的输出数字的方法。（1分）
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。（0.5分）实现父类的run方法，调用Printer类中的输出字母的方法。（1分）
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。（1分）

**打印类（共享资源）**

|  |
| --- |
| **public** **class** Printer {  **private** **int** index=1;//设为1，方便计算3的倍数  //打印数字的方法，每打印两个数字，等待打印一个字母  **public** **synchronized** **void** print(**int** i){  **while**(index%3==0){  **try** {  wait();  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  System.*out*.print(""+i);  index++;  notifyAll();  }  //打印字母，每打印一个字母，等待打印两个数字  **public** **synchronized** **void** print(**char** c){  **while**(index%3!=0){  **try** {  wait();  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  System.*out*.print(""+c);  index++;  notifyAll();  }  } |

**打印字母线程**

|  |
| --- |
| **public** **class** LetterPrinter **extends** Thread {  **private** Printer p;  **public** LetterPrinter(Printer p){  **this**.p=p;  }  **public** **void** run(){  **for**(**char** c='A';c<='Z';c++){  p.print(c);  }  }  } |

**打印数字线程**

|  |
| --- |
| **public** **class** NumberPrinter **extends** Thread {  **private** Printer p;  **public** NumberPrinter(Printer p){  **this**.p=p;  }  **public** **void** run(){  **for**(**int** i=1;i<=52;i++){  p.print(i);  }  }  } |

**测试类**

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[] args) {  Printer p=**new** Printer(); //创建打印机对象  Thread t1=**new** NumberPrinter(p); //创建线程对象  Thread t2=**new** LetterPrinter(p); //创建线程对象  t1.start(); //启动线程  t2.start(); //启动线程  }  } |

1. **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。**

**编写实现，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。（6分）**

|  |
| --- |
| **public** **class** ThreadTest { //创建类  **private** **int** j; //声明成员变量j  **public** **static** **void** main(String[] args) { //程序主入口  ThreadTest tt=**new** ThreadTest(); //创建ThreadTest的对象  **//创建内部类对象正确各0.5分，共1分**  Inc inc=tt.**new** Inc(); //创建内部线程类对象  Dec dec=tt.**new** Dec(); //创建内部线程类对象  **//如不使用循环，而是手动创建4个线程，可适当给分**  **for**(**int** i=0;i<2;i++){  **// 创建线程对象正确，启动线程各0.5分共1分**  Thread t=**new** Thread(inc);  t.start();  **// 创建线程对象正确，启动线程各0.5分共1分**  t=**new** Thread(dec);  t.start();  }  }  **//实现同步对j的值加，1分**  **private** **synchronized** **void** inc(){  j++; //j++  System.*out*.println(Thread.*currentThread*().getName()+"-inc:"+j);  }  **//实现同步对j的值减，1分**  **private** **synchronized** **void** dec(){  j--;  System.*out*.println(Thread.*currentThread*().getName()+"-dec:"+j);  }  //内部类  **//内部类定义正确并实现run方法1分**  **class** Inc **implements** Runnable{  @Override  **public** **void** run() {  **for** (**int** i = 0; i <100; i++) {  inc(); //调用加的方法  }  }  }  **//内部类定义正确并实现run 方法1分**  **class** Dec **implements** Runnable{  @Override  **public** **void** run() {  **for** (**int** i = 0; i <100; i++) {  dec(); //调用减的方法  }  }  }  } |

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。（6分）**

**public** **class** MulThreadTest{

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

//创建一个山洞(0.5分)

Tunnel tul = **new** Tunnel();

//创建十个过山洞线程（1.0分）

Thread p1 = **new** Thread(tul,"p1");

Thread p2 = **new** Thread(tul,"p2");

Thread p3 = **new** Thread(tul,"p3");

Thread p4 = **new** Thread(tul,"p4");

Thread p5 = **new** Thread(tul,"p5");

Thread p6 = **new** Thread(tul,"p6");

Thread p7 = **new** Thread(tul,"p7");

Thread p8 = **new** Thread(tul,"p8");

Thread p9 = **new** Thread(tul,"p9");

Thread p10 = **new** Thread(tul,"p10");

//启动十个线程0.5分

p1.start(); p2.start();

p3.start(); p4.start();

p5.start(); p6.start();

p7.start(); p8.start();

p9.start(); p10.start();

}

}

**class** Tunnel **implements** Runnable{

**private** **int** crossedNum = 0;//初始人数0.5分

**public** **void** run(){//1分

cross();

}

**public** **synchronized** **void** cross(){

**try** { //每个人通过山洞的时间为5秒（1分）

Thread.*sleep*(5000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

//人数计数（0.5分）

crossedNum++;

//显示一下每次通过山洞人的姓名（1分）

System.*out*.println(Thread.*currentThread*().getName()+

"通过了山洞，这是第"+crossedNum+"个用户");

}

}