**01. 传统线程技术回顾**

线程实现的两种方式：Thread Runable

多线程会提高程序的运行效率么？不会，一般来说会更慢。cpu切换不同线程，会降低性能。多线程下载快？抢了服务器的带宽，每个线程都占一定量的带宽，下载加快了，只是服务器开了多个线程，占了多个带宽

**02. 传统定时器技术回顾**

Timer类 TimerTask

相隔的时间不一样?

quarts

**03. 传统线程互斥技术**

2个线程访问同一个对象，线程安全问题（多线程环境+共享数据+多条语句操作共享数据）

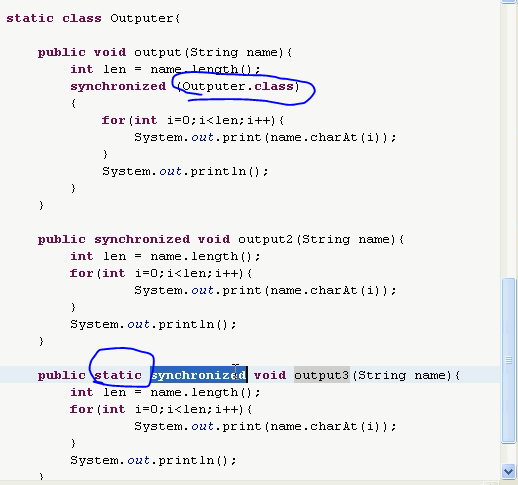
为了实现被执行的代码的原子性:

1.把这一段代码括起来加synchronized 必须用同一个对象 this

2.方法前面加synchronized

检查一个锁对象，几个线程想在某段代码上实现同步，必须用同一把锁

3.想让第一个和第三个用同一个所对象(普通方法和静态方法)



**04. 传统线程同步通信技术**

wait() ，notify()

面试题：子线程循环10次，主线程循环100次，又回到子线程循环10次再回到主线程循环100次

按这个规律循环50次写出程序：

|  |
| --- |
| public class TraditionalThreadCommunication {  public static void main(String[] args) {  final Business business = new Business();  new Thread(  new Runnable() {  @Override  public void run() {  for(int i=1;i<=50;i++){  business.sub(i);  }  }  }  ).start();  for(int i=1;i<=50;i++){  business.main(i);  }  }  }  class Business {  private boolean bShouldSub = true;  public synchronized void sub(int i){  while(!bShouldSub){  try {  this.wait();  } catch (InterruptedException e) {  e.printStackTrace();  }  }  for(int j=1;j<=10;j++){  System.out.println("sub thread sequence of " + j + ",loop of " + i);  }  bShouldSub = false;  this.notify();  }  public synchronized void main(int i){  while(bShouldSub){  try {  this.wait();  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  for(int j=1;j<=100;j++){  System.out.println("main thread sequence of " + j + ",loop of " + i);  }  bShouldSub = true;  this.notify();  }  } |

**05. 线程范围内共享变量的概念与作用**

**06. ThreadLocal类及应用技巧**

**07. 多个线程之间共享数据的方式探讨**

08. java5原子性操作类的应用

09. java5线程并发库的应用

10. Callable与Future的应用

11. java5的线程锁技术

12. java5读写锁技术的妙用

13. java5条件阻塞Condition的应用

14. java5的Semaphere同步工具

15. java5的CyclicBarrier同步工具

16. java5的CountDownLatch同步工具

17. java5的Exchanger同步工具

18. java5阻塞队列的应用

19. java5同步集合类的应用

20. 空中网挑选实习生的面试题1

21. 空中网挑选实习生的面试题2

22. 空中网挑选实习生的面试题3

23. 源代码与资料