1. 作用

当一个线程需要等待其他线程完成任务之后，才能接着往下执行。

这就相当于大家要去郊游：

需要等待所有人都到齐才可以出发，如果有一个人没到，司机都要等着。

2. 构造方法

new CountDownLatch(3);//构造一个指定数量的对象

3. 常用方法

public void await() throws InterruptedException { }; //调用await()方法的线程会被挂起， 它会等待直到count值为0才继续执行

public boolean await(long timeout, TimeUnit unit) throws InterruptedException { }; //和 await()类似，只不过等待一定的时间后count值还没变为0的话就会继续执行

public void countDown() { }; //将count值减1

4. demo

**public static void** main(String[] args){  
 *//定义三个人准备斗地主* CountDownLatch countDownLatch = **new** CountDownLatch(3);  
 *//刘备* Thread liu = **new** Thread(()-> {  
 **try** {  
 Thread.*sleep*(3000);  
 }**catch** (InterruptedException e){};  
 System.***out***.println(**"刘备来了...在准备"**);  
 countDownLatch.countDown();  
 });  
 *//张飞* Thread fei = **new** Thread(()->{  
 **try** {  
 Thread.*sleep*(1000);  
 }**catch** (InterruptedException e){};  
 System.***out***.println(**"张飞来了...在准备"**);  
 countDownLatch.countDown();  
 });  
 *//关羽* Thread yu = **new** Thread(()->{  
 **try** {  
 Thread.*sleep*(2000);  
 }**catch** (InterruptedException e){};  
 System.***out***.println(**"关羽来了...在准备"**);  
 countDownLatch.countDown();  
 });  
 *//开始约会* liu.start();  
 fei.start();  
 yu.start();  
 **try** {  
 countDownLatch.await();  
 }**catch** (InterruptedException e){};  
 System.***out***.println(**"大家都到齐了，开始斗地主"**);  
}

结果：

