1. 作用

循环栅栏，作用于CountDownLatch类似，但是CyclicBarrier技术到达指定值可循环使用。



2. 构造函数

public CyclicBarrier(int parties, Runnable barrierAction) {}

public CyclicBarrier(int parties) {}

参数parties指让多少个线程或者任务等待至barrier状态；

参数barrierAction为当这些线程都达到barrier状态时会执行的内容。

3. 常用方法

public int await() throws InterruptedException, BrokenBarrierException { };//挂起当前线程，直至所有线程都到达barrier状态再同时执行后续任务；

public int await(long timeout, TimeUnit unit) throws InterruptedException,BrokenBarrierException,TimeoutException { };//让这些线程等待至一定的 时间，如果还有线程没有到达barrier状态就直接让到达barrier的线程执行后续任务

4. demo

**public class** CountDownLatchDemo {  
 **public static class** Dou **implements** Runnable{  
 **private int sum** = 0;  
 @Override  
 **public void** run() {  
 **sum**++;  
 System.***out***.println(**"大家都到齐了，开始第"**+**sum**+**"局斗地主"**);  
 }  
 }  
 **public static void** main(String[] args){  
 *//定义三个人准备斗地主* CyclicBarrier barrier = **new** CyclicBarrier(3,**new** Dou());  
 Runnable runnable = () -> {  
 **while**(**true**){  
 **try** {  
 Thread.*sleep*(**new** Random().nextInt(4000));  
 }**catch** (InterruptedException e){};  
 System.***out***.println(Thread.*currentThread*().getName()+**"来了...在准备"**);  
 **try** {  
 barrier.await();  
 }**catch** (BrokenBarrierException|InterruptedException e){};  
 }  
 };  
 *//刘备* Thread liu = **new** Thread(runnable,**"刘备"**);  
 *//张飞* Thread fei = **new** Thread(runnable,**"张飞"**);  
 *//关羽* Thread yu = **new** Thread(runnable,**"关羽"**);  
 *//开始约会* liu.start();  
 fei.start();  
 yu.start();  
 }  
}

结果：

