

多线程

经验

```
class MyCallable implements Runnable {
    private static void main() {
        for (int i = 0; i < 100; i++) {
            // ...
        }
    }
}
```

这里面sleep函数放在循环里面 一般好像run方法里面都是循环

运行效果

运行效果

join位置



补充

```
class Demo {
    public static void main(String[] args) throws Exception {
        MyCallable mc = new MyCallable();
        Thread t1 = new Thread(mc);
        Thread t2 = new Thread(mc);
        t1.setName("线程1");
        t2.setName("线程2");
        t1.start();
        t2.start();
        // ...
    }
}
```

避免执行顺序在上面 只要最后的一个线程执行完 这个代码就可以不同 效果一样

Thread.join

放在t1和t2上面 中间 下面 结果都不同 以及t1和t2只有一个运行和两个都运行结果也都不一样 (!!! join必须要紧接着放在当前执行的线程, 以起到隔断的作用。!!!) 才能达到图片的效果

lock死锁例子

```
class TestThreadLock implements Runnable {
    private static void main() {
        // ...
    }
}
```

方法

lock死锁解锁---重要易错点



thread睡眠

推荐使用TimeUnit.MILLISECONDS.sleep(10);代替thread.sleep

finally知识点+lock锁

如果产生死锁的话finally就不会执行到 卡死进程

callable和runable的区别

callable(要有返回值 callable后要有泛型 比如String 进行线程实现需要调用futuretask函数 实现的是call方法 以及如果不多次调用futuretask函数就会单条线程执行到底 例如三条线程需要new三次futuretask 可在futuretask后加上泛型 例如String 但多次实验发现即使多次new该futuretask 交替执行并没有那么明显 现在推荐使用runable() runnable没有返回值 实现的是run方法 常用

建议使用runable创建多线程

- 1. 什么场景用? 在多线程时, 有多个任务或多个CPU上(交替)运行
 - 2. 什么场景用? 在多线程时, 有多个任务或多个CPU上(同时)运行
- 扩展 并发与并行的区别

分支主题

分支主题

分支主题

分支主题

分支主题