龟兔赛跑

### 【案例介绍】

**1.任务描述**

众所周知的“龟兔赛跑”故事，兔子因为太过自信，比赛中途休息而导致乌龟赢得了比赛.本案例要求编写一个程序模拟龟兔赛跑，乌龟的速度为1米/1500毫秒，兔子的速度为5米/500毫秒，等兔子跑到第700米时选择休息10000毫秒，结果乌龟赢得了比赛。

**2.运行结果**

运行结果如图8-1所示。

手机屏幕截图

描述已自动生成

图2-1 运行结果

### 【案例目标】

* 学会分析”龟兔赛跑”任务实现的逻辑思路。
* 能够独立完成”龟兔赛跑”程序的源代码编写、编译以及运行。
* 能够在程序中使用多线程完成逻辑思路。

### 【案例思路】

1. 查看运行结果分析后，首先创建一个Torist（）方法作为乌龟线程的内部类，在Torist（）方法中使用sleep模拟乌龟跑步。
2. 查看运行结果分析后，创建一个Rabbit（）方法作为兔子线程的内部类，在Torist（）方法中使用sleep模拟乌龟跑步。
3. 最后在main方法中调用Torist（）与Rabbit（）方法实现龟兔赛跑。

### 【案例代码】

龟兔赛跑小程序的代码如文件8-1所示。

文件8-1 race.java

package chapter0401;

public class race {

private int toristDistance;//乌龟跑过的距离

private int rabbitDistance;//兔子跑过的距离

/\*\*

\* 乌龟线程内部类

\*/

class Torist extends Thread{

@Override

public void run() {

//分析编程代码

for(int i=1;i<=800;i++){

//判断兔子是否到达终点

if(rabbitDistance==800){

//当兔子先800的时候 兔子就已经赢了

System.out.println("兔子赢得了比赛，此时乌龟才跑了"+toristDistance+"米");

break;

}else{

//乌龟开始跑

toristDistance+=1;

//判断距离是否是100的倍数

if(toristDistance%100==0){

try {

if(rabbitDistance==700){

System.out.println("乌龟跑了"+toristDistance+"米，此时兔子在睡觉");

}else{

System.out.println("乌龟跑了"+toristDistance+"米，此时兔子跑过段距离

是"+rabbitDistance);

}

Thread.sleep(1500);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

/\*\*

\* 兔子线程内部类

\*/

class Rabbit extends Thread{

@Override

public void run() {

//分析编程代码

for(int i=1;i<=800/5;i++){

//判断兔子是否到达终点

if(toristDistance==800){

//当兔子先1000的时候 兔子就已经赢了

System.out.println("乌龟赢得了比赛，此时兔子跑了"+rabbitDistance+"米");

break;

}else{

//乌龟开始跑

rabbitDistance+=5;

//判断距离是否是100的倍数

if(rabbitDistance%100==0){

try {

System.out.println("兔子跑了"+rabbitDistance+"米，乌龟跑过了"

+toristDistance);

if (rabbitDistance==700) {

System.out.println("兔子觉得自己怎么能可以赢得比赛，所以选择睡一会");

Thread.sleep(10000);

}

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

}

//测试

public static void main(String[] args) {

//1 外部类实例构建

race outer=new race();

//2兔子 乌龟线程实例构建

Rabbit rabbit=outer.new Rabbit();

Torist torist=outer.new Torist();

//3 依次启动

//在现实中 也不可能两个同时跑 这样也是很公平的

rabbit.start();

torist.start();

}

}

代码第9~38行代码是乌龟线程的内部类，使用for循环嵌套Thread的sleep方法模拟乌龟跑步，代码第42~72行代码是兔子线程的内部类，与乌龟线程的内部类类似，第76~84行代码是启动线程，模拟乌龟兔子赛跑。