单元 9.1 跑步比赛（一）

单元教学进度设计

教学环节	教学内容	教师活动	学生活动	时间
十分钟测试	**1．编写线程类，要继承的父类是：** A：Object B：Runnable C：Serializable D：Thread E: Exception 答案：D	提问解析	抢答	10
	2．编写线程类，可以通过实现那个接口来实现？ A：Runnable B：Throwable C：Serializable D：Comparable E：Cloneable 答案：A	提问解析	抢答	
	3．如果要一个线程等待一段时间后再恢复执行此线程，需要调用什么方法？ A：wait B：yield C：join D：sleep E：stop F：notify 答：D	提问解析	抢答	
	4.Runnable 接口定义了如下哪些方法？ A: start() B: stop() C: resume() D: run() E: suspend() 答案：D	提问解析	抢答	
	5. 如下代码创建一个新线程并启动线程：	提问	抢答	

	Runnable target=new MyRunnable(); Thread myThread=new Thread(target); 问:如下哪些类可以创建 target 对象，并能编译正确？ A: public class MyRunnable extends Runnable { public void run(){} } B: public class MyRunnable extends Object { public void run() {} } C: public class MyRunnable implements Runnable {public void run() {}} D: public class MyRunnable extends Runnable {void run() {}} E: public class MyRunnable implements Runnable {void run() {}} 答：C	**解析**		
	6.下列方法中可以用来创建一个新线程的是（　　）。 A.实现 java.lang.Runnable 接口并重写 start()方法 B.实现 java.lang.Runnable 接口并重写 run()方法 C.实现 java.lang.Thread 类并重写 run()方法 D.实现 java.lang.Thread 类并重写 start()方法 答案：C	**提问 解析**	**抢答**	
	7. 启动一个线程是用 run()还是 start()？这两个方法有什么关系？ 答：启动一个线程是调用 start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 JVM 调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。	**提问 解析**	**抢答**	
小结	进程是指在系统中正在运行的一个应用程序；线程是系统分配处理器时间资源的基本单元，或者说进程之内独立执行的一个单元。对于操作系统而言其调度单元是线程。一个进程至少包括一个线程，通常将该线程称为主线程。一个进程从主线程的执行开始进而创建一个或多个附加线程，就是所谓基于多线程的多任务 创建线程： （1）线程类继承 Thread，重写 run()方法，在 run()方法中完成此线程所要完成的工作，直接创建线程类的对象，然后调用 start()方法启动线程，默认调用 run()方法 （2）线程类实现 Runnable 接口，实现 run()方法，在 run()方法中完成此线程所要完成的工作；创建线程的时候要注意： 首先创建线程类的对象 t，然后通过 Thread tt = new Thread(t);来创建线程 tt。也是通过调用 start()方法来启动线程。 要想创建一个线程类，实质上是必须实现 Runnable 接口，因为 Thread 本身也是实现了 Runnable 接口的。但是如果要调用 start()方法来启动一个线程，就必须构建 Thread 类的对象：如果是用第①中方式创建的线程类，	**总结**	**记录**	

<table>
<tr><td></td><td></td><td>那么他所创建的对象就可以隐式转换成 Thread 对象，所以可以直接调用 start()方法；如果使用第②种方式创建的线程类，那么使用这个线程类创建对象之后，还必须通过这个对象创建一个 Thread 类的对象。</td><td></td><td></td><td></td></tr>
<tr><td colspan="2">新课导入</td><td>我们一定听过龟兔赛跑的故事，说的是在一个原始森林里动物乌龟和兔子举行了一次赛跑。结果兔子因小瞧了乌龟，在半道上睡觉，叫乌龟拿了奖杯。结果乌龟家族受动物们的羡慕，兔子家族却臭名远扬。
这个经典的寓言故事，经过了网络大神们的加工，衍生出了 n 多前传、后传和传奇，今天我们一起来做一个下游戏，游戏的主人公不再是兔子和乌龟，而是大名鼎鼎的小破孩和小丫。
小破孩和小丫参加了学校组织的运动会，让我们来看一看他们俩谁跑的更快呢？
请同学们先看一下程序的运行。
这个程序和我们以前所做的所有程序都是不一样的，以前我们做的程序只有一个窗口，而这个程序不但有窗口，还有两个演员，他们分别是 Thread 公司的代言人，和 Runnable 公司的代言人，让我们编程看一看，那个公司的代言人跑的更快吧！</td><td>提出问题
分析</td><td>讨论
思考</td><td>10</td></tr>
<tr><td rowspan="2">任务一</td><td>任务引入</td><td>在赛跑之前呢，我们先来构建游戏的场景，小屁孩和小丫在一个长 800，宽 300 的的跑道上赛跑。
请同学们将操场定义出来。</td><td>布置任务</td><td>思考</td><td rowspan="2">25</td></tr>
<tr><td>任务实施 1</td><td>5 分钟时间，定义一个 800*300 的窗口。
展示部分同学带背景的操场。</td><td>辅导</td><td>编程</td></tr>
<tr><td rowspan="3">任务二</td><td>任务引入</td><td>首先出场的是 Thread 公司的代言人，小屁孩。</td><td>引入</td><td>思考</td><td rowspan="3"></td></tr>
<tr><td>任务部署</td><td>通过继承 Thread 类，并覆盖 run()方法，这时就可以用该类的实例作为线程的目标对象。</td><td>讲解
总结</td><td>思考</td></tr>
<tr><td>任务执行</td><td>1.将小屁孩显示在操场上。
为了方便的站到操场上，我们将小屁孩定义为操场的内部类。
 
</td><td>编程</td><td>思考
计算</td><td>20</td></tr>
</table>

实例化小屁孩对象，默认会调用构造方法将小屁孩站在起跑线上。

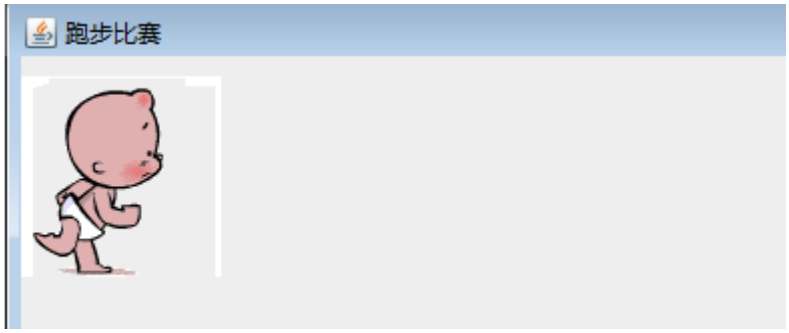
（1）成员变量 xph `Thread xph;` ，用父类的引用指向子类的对象。

（2）实例化小屁孩对象，注意 Containerc 和小屁孩的先后顺序：

```
c=this.getContentPane();
c.setLayout(null);
xph=new Boy();
```

注意，实例化c后，才让小屁孩站在起跑线上。

程序运行如下：



2. 重写 run 方法，让小屁孩跑动起来。

```
int i=0;
public void run(){
    i++;
    while(true){
        i++;
        me.setBounds(i*20, 0, 100, 100);
        c.add(me);
    }
}
```

用来记录跑了多少步

每跑一步，前进20个px

重写显示小屁孩位置

程序运行后，发现操场上空空如也，没有了小屁孩的踪影，这是怎么回事呢？

为了监控程序的运行，我们每跑一步，都输出一句话。

我们发现，虽然操场上没有了小屁孩的踪影，但是控制台上不停的打印出“小屁孩，加油！”的语句，说明，小屁孩依然在跑。

```
while(true){
    i++;
    me.setBounds(i*20, 0, 100, 100);
    c.add(me);
    System.out.println("小屁孩，加油！");
}
```

```
小屁孩，加油！
小屁孩，加油！
小屁孩，加油！
小屁孩，加油！
小屁孩，加油！
```

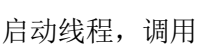
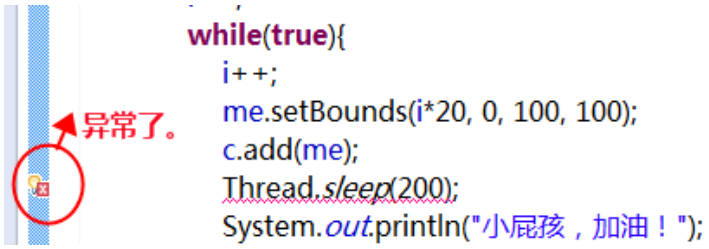
这是为什么呢？

从打印的频率可以看出，小屁孩迈步子的速度是非常快的，“嗖”的一下就不见了，跑出了操场，继续跑。。。。

为什么一直跑呢？因为我们定义了死循环。

演示 讲解

学习 修改

<table>
<tr><td></td><td></td><td>那，如何让小屁孩的速度慢一点，让我们能看到他跑步的过程呢？</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3. 让小屁孩跑起来
启动线程，调用 start（）方法。
 
</td><td>演示
讲解</td><td>学习
记录</td><td></td></tr>
<tr><td></td><td></td><td>4. 让小屁孩跑慢一点
我们在小屁孩每迈出一步的时候，sleep 一下，通过课前的学习，我们知道，sleep 是让线程等待的意思。
 

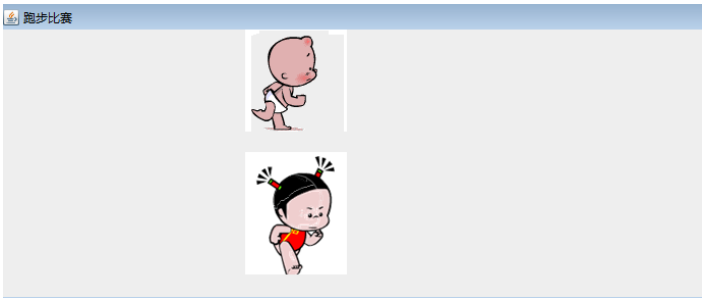
显然这个操作会抛出 InterruptedException，我们需要对这个异常进行处理。
 

经过修改后，小屁孩在操场上跑了起来。</td><td>演示
讲解</td><td>学习
记录</td><td></td></tr>
<tr><td rowspan="3">任务三</td><td>任务引入</td><td>然后出场的是为 Runnable 代言的运动员小丫。
我们依然使用内部类。</td><td>引入</td><td>思考</td><td rowspan="3">20</td></tr>
<tr><td>任务部署</td><td>可以定义一个类实现 Runnable 接口，然后将该类对象作为线程的目标对象。实现 Runnable 接口就是实现 run()方法。</td><td>引导</td><td>思考
讨论</td></tr>
<tr><td>任务实施</td><td>1. 定义 Girl 类，实现 Runnable 接口，必须实现 run 方法。</td><td>演示
讲解</td><td>学习
记录</td></tr>
</table>

		public class Girl implements Runnable{
		JLabel me=new JLabel(new ImageIcon("images/girl.gif"));
		public Girl(){
		me.setBounds(0, 150, 100, 120);
		c.add(me);
		}
		public void run() {
		}
		}
		构造方法，小丫站在起跑线上。
		实现Runnable的run方法
		注意：
		1.方法是public类型
		2.只有一个虚拟方法要实现。			

		2. 实现 run（）方法
		public void run() {
		int i=0;
		while(true){
		i++;
		me.setBounds(i*20, 0, 100, 100);
		c.add(me);
		try {
		Thread.sleep(200);
		} catch (InterruptedException e) {
		e.printStackTrace();
		}
		System.out.println("小丫，加油！");
		}
		}
		记录跑了多少步
		设定速度
		可见，与小屁孩的速度相同。
			演示
			讲解
				学习
				记录	

		3. 实例化小丫对象，让小丫站在起跑线上。 （1）声明线程小丫。Thread xy; （2）实例化小丫：xy=new Thread(new Girl()); 注意，与实例化小屁孩不同，小丫归根结底也是一个 Thread 对象，一个实现了 Runnable 接口的 Thread 对象。	演示 讲解	学习 记录	
		4. 让小丫跑起来。 启动线程，调用 start（）方法。 xy.start();	演示 讲解	学习 记录	
任务	任务导入	运行后，发现小丫和小屁孩的跑步速度差不多，但是从我们在工作台上打印出的结果还是可以看出来小屁孩线程和小丫线程交叉运行的结果。	演示 讲解	学习 记录	10

<table>
<tr><td rowspan="3">四</td><td></td><td> 

如何让小屁孩和小丫的显示出赛跑的效果呢？
根据实际跑步，匀速前进肯定是不合理的，因此我们可以给他们以随机数的形式，设置每一步迈出的速度。速度设置为（100-300）之间
请同学们自己将程序修改一下。</td><td></td><td></td><td rowspan="3"></td></tr>
<tr><td rowspan="2">任务实施</td><td>

```
int speed=(int)(Math.random()*200)+100;
try {
    Thread.sleep(speed);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

生成 100-300 之间是随机数，作为当前速度，小丫和小屁孩的比赛就演示出来了。</td><td>辅导</td><td>编程</td></tr>
<tr><td>思考：
如果在线程开始的时候，2 次调用 star（）方法，会不会产生两个线程呢？
结果产生了异常：

```
<terminated> Playground3 [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2016年5月:
Exception in thread "main" java.lang.IllegalThreadStateException
        at java.lang.Thread.start(Unknown Source)
        at game.Playground3.<init>(Playground3.java:18)
        at game.Playground3.main(Playground3.java:28)
```

所以说，一个线程只能调用一次 start（）方法。</td><td>演示讲解</td><td>学习记录</td></tr>
<tr><td>教学小结</td><td>1. 什么是线程？
2. 线程的创建有两种方法
（1）继承 Thread 类的线程。
（2）实现 Runnable 接口的线程。
3. 所有的多线程代码执行顺序都是不确定的，每次执行的结果都是随机的。</td><td>总结</td><td>听讲记录</td><td>5</td></tr>
<tr><td>布置作业</td><td colspan="4">同步课外项目：模拟时钟。
本节复习：课堂录像、项目说明文档
作业要求：程序上传至云盘，放在 *组-姓名 文件夹下。</td></tr>
</table>

课后学习资源	下节预告 项目 2.5.2　跑步比赛（二） 课前储备： 微课《线程的常用方法》