Java 核心技术(进阶)

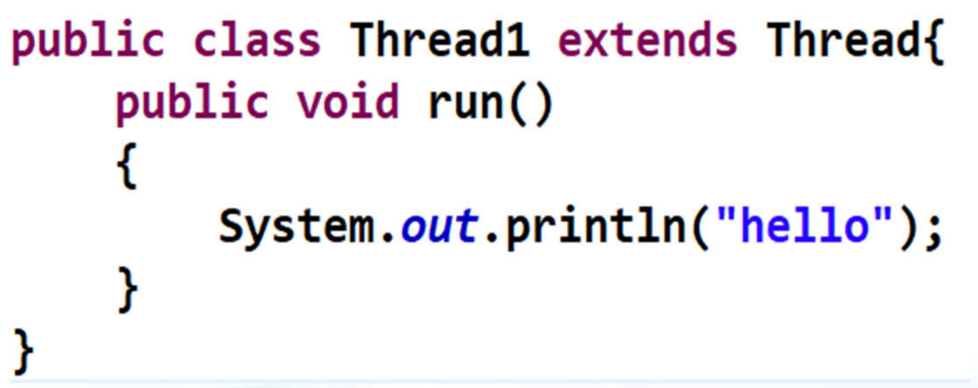
第五章 Java多线程和并发编程

第二节 Java多线程实现

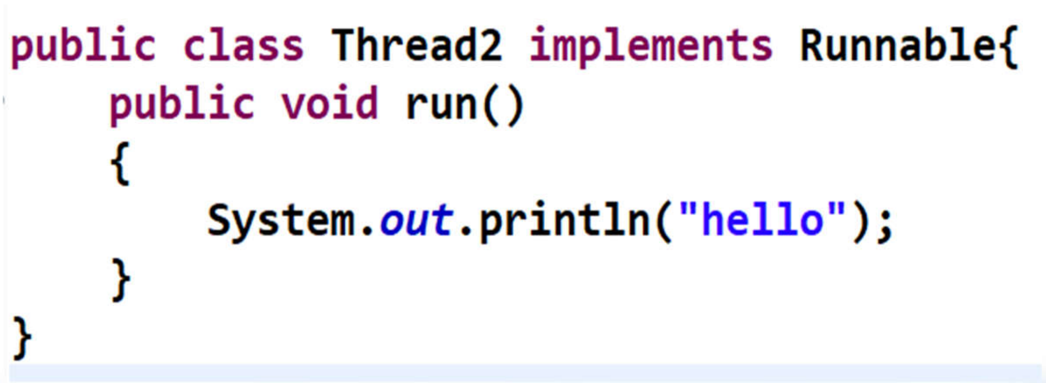
华东师范大学 陈良育

Java 多线程创建

- java.lang.Thread
  - 线程继承Thread类，实现run方法
- java.lang.Runnable接口
  - 线程实现Runnable接口，实现run方法



```java
public class Thread1 extends Thread{
    public void run()
    {
        System.out.println("hello");
    }
}
```



```java
public class Thread2 implements Runnable{
    public void run()
    {
        System.out.println("hello");
    }
}
```

Java多线程启动

- 启动
  - start方法，会自动以新进程调用run方法
  - 直接调用run方法，将变成串行执行
  - 同一个线程，多次start会报错，只执行第一次start方法
  - 多个线程启动，其启动的先后顺序是随机的
  - 线程无需关闭，只要其run方法执行结束后，自动关闭
  - main函数(线程)可能早于新线程结束，整个程序并不终止
  - 整个程序终止是等所有的线程都终止(包括main函数线程)

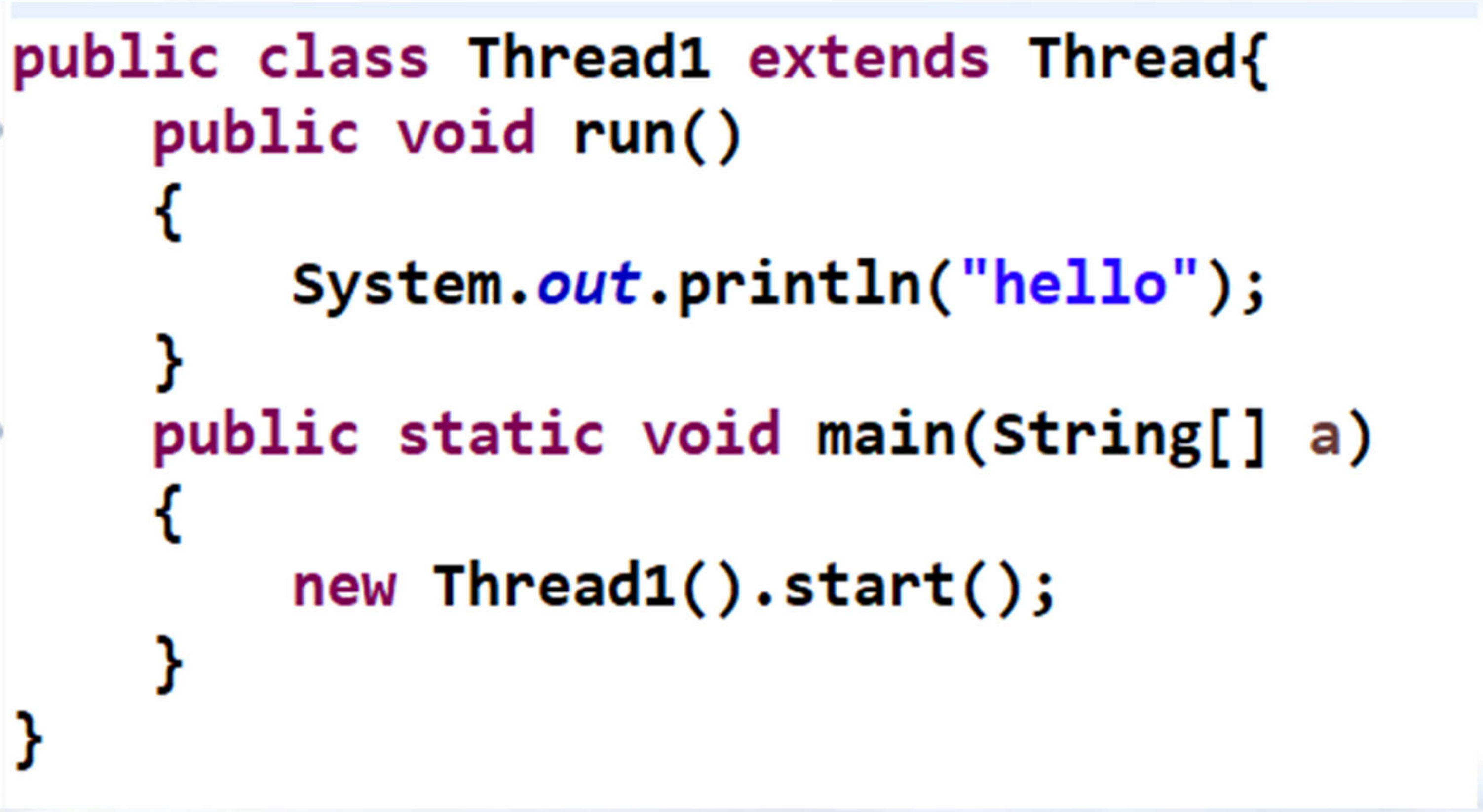
Java 多线程实现对比

- Thread vs Runnable
  - Thread占据了父类的名额，不如Runnable方便
  - Thread 类实现Runnable
  - Runnable启动时需要Thread类的支持
  - Runnable 更容易实现多线程中资源共享
- 结论：建议实现Runnable接口来完成多线程

总结

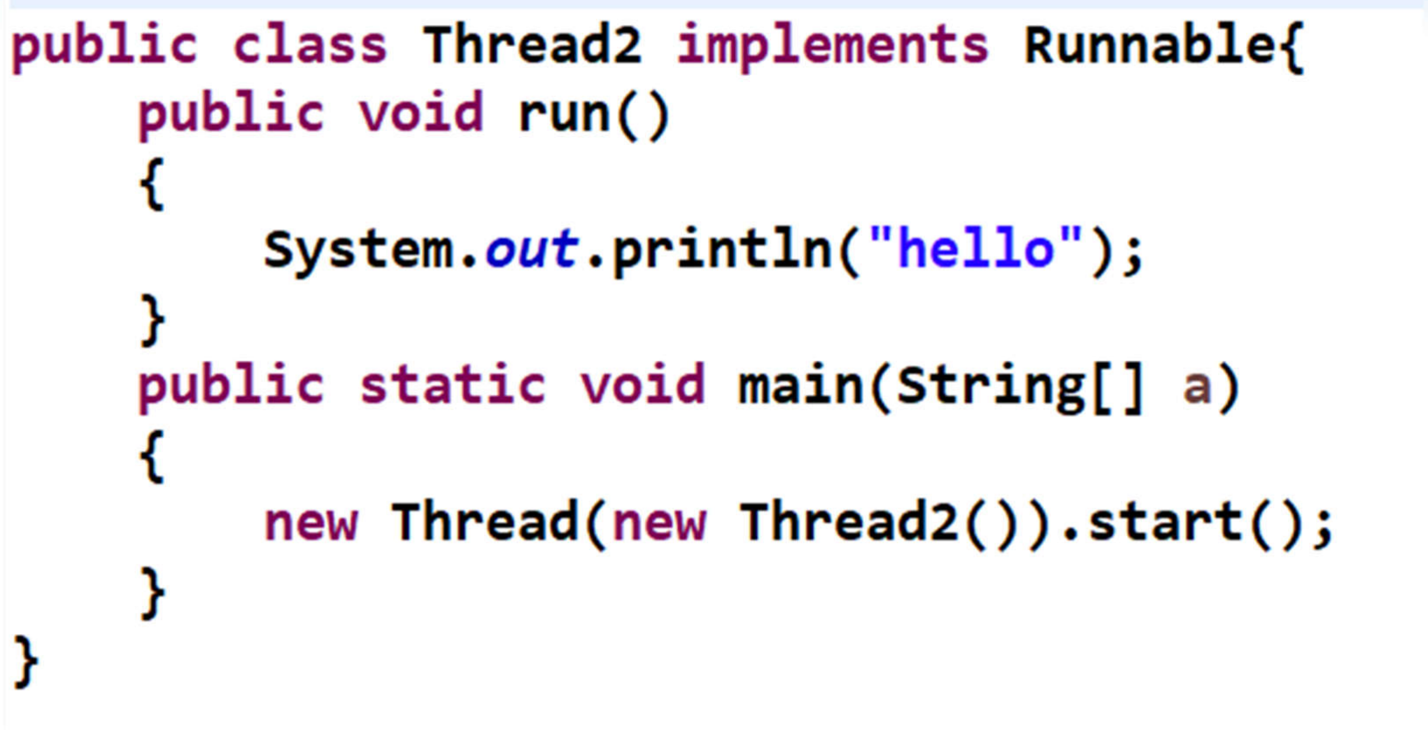
- 总结
  - 了解Java多线程两种实现方式
  - 了解Java多线程运行基本规则

代码(1) Thread1.java



```
public class Thread1 extends Thread{
    public void run()
    {
        System.out.println("hello");
    }
    public static void main(String[] a)
    {
        new Thread1().start();
    }
}
```

代码(2) Thread2.java

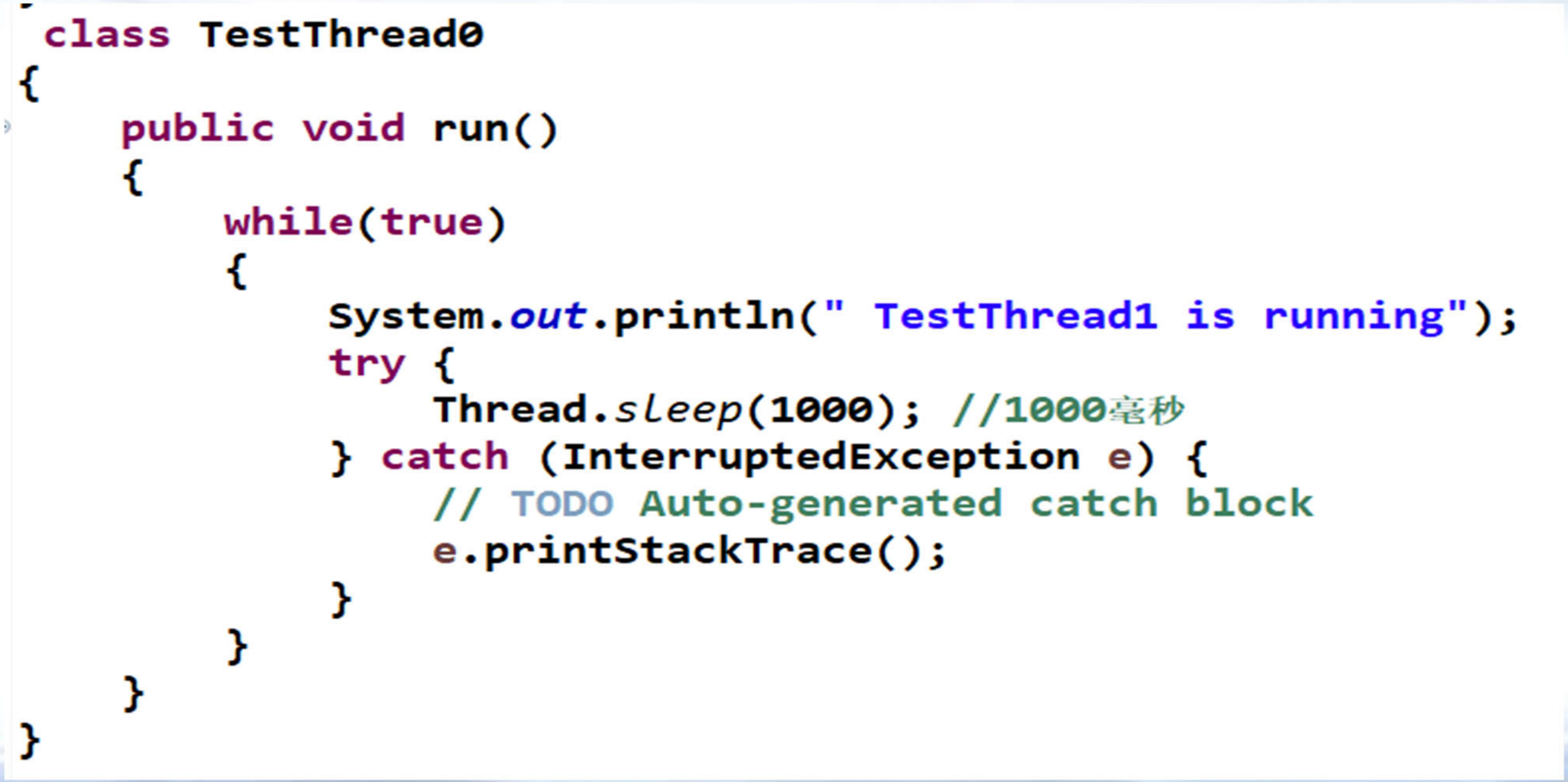


```
public class Thread2 implements Runnable{
    public void run()
    {
        System.out.println("hello");
    }
    public static void main(String[] a)
    {
        new Thread(new Thread2()).start();
    }
}
```

代码(3) ThreadDemo0.java

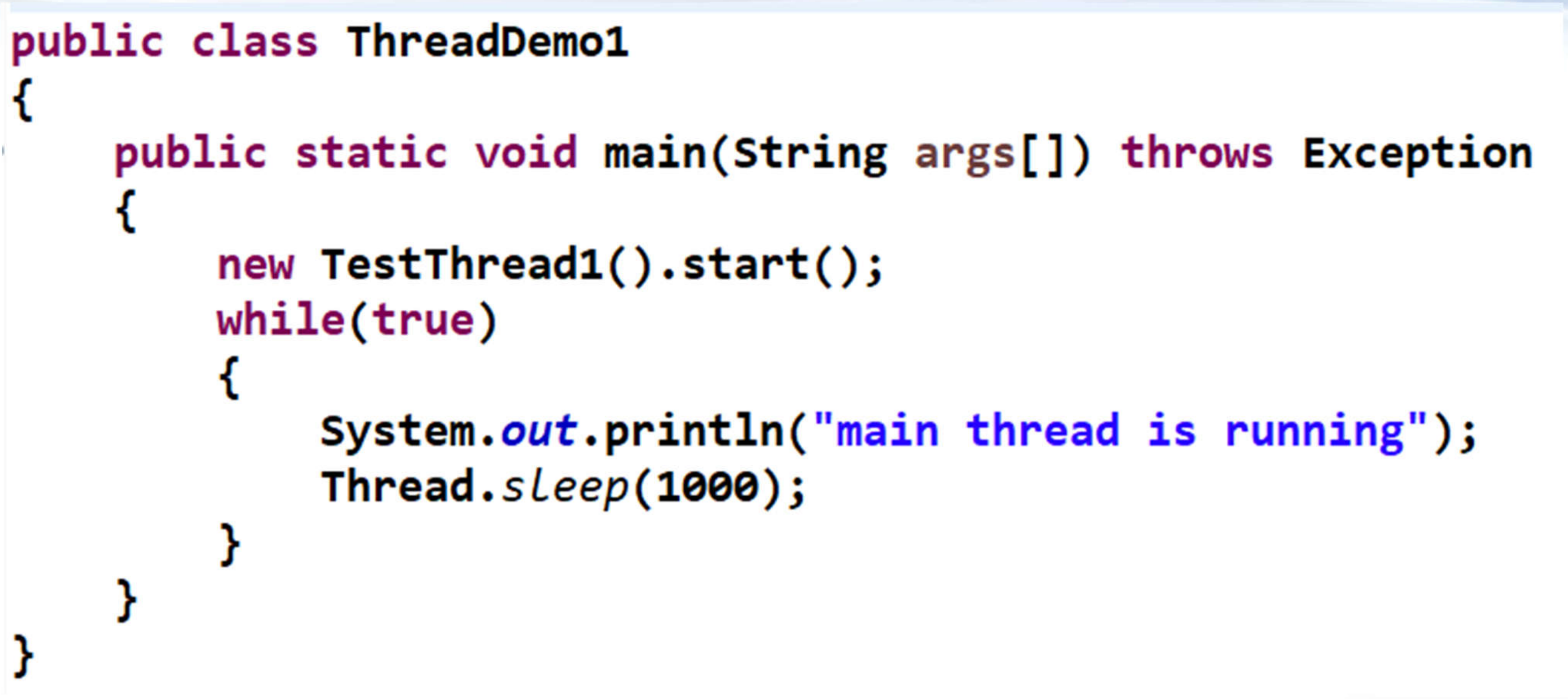
```
public class ThreadDemo0
{
    public static void main(String args[]) throws Exception
    {
        new TestThread0().run();
        while(true)
        {
            System.out.println("main thread is running");
            Thread.sleep(10);
        }
    }
}
```

代码(4) TestThread0.java



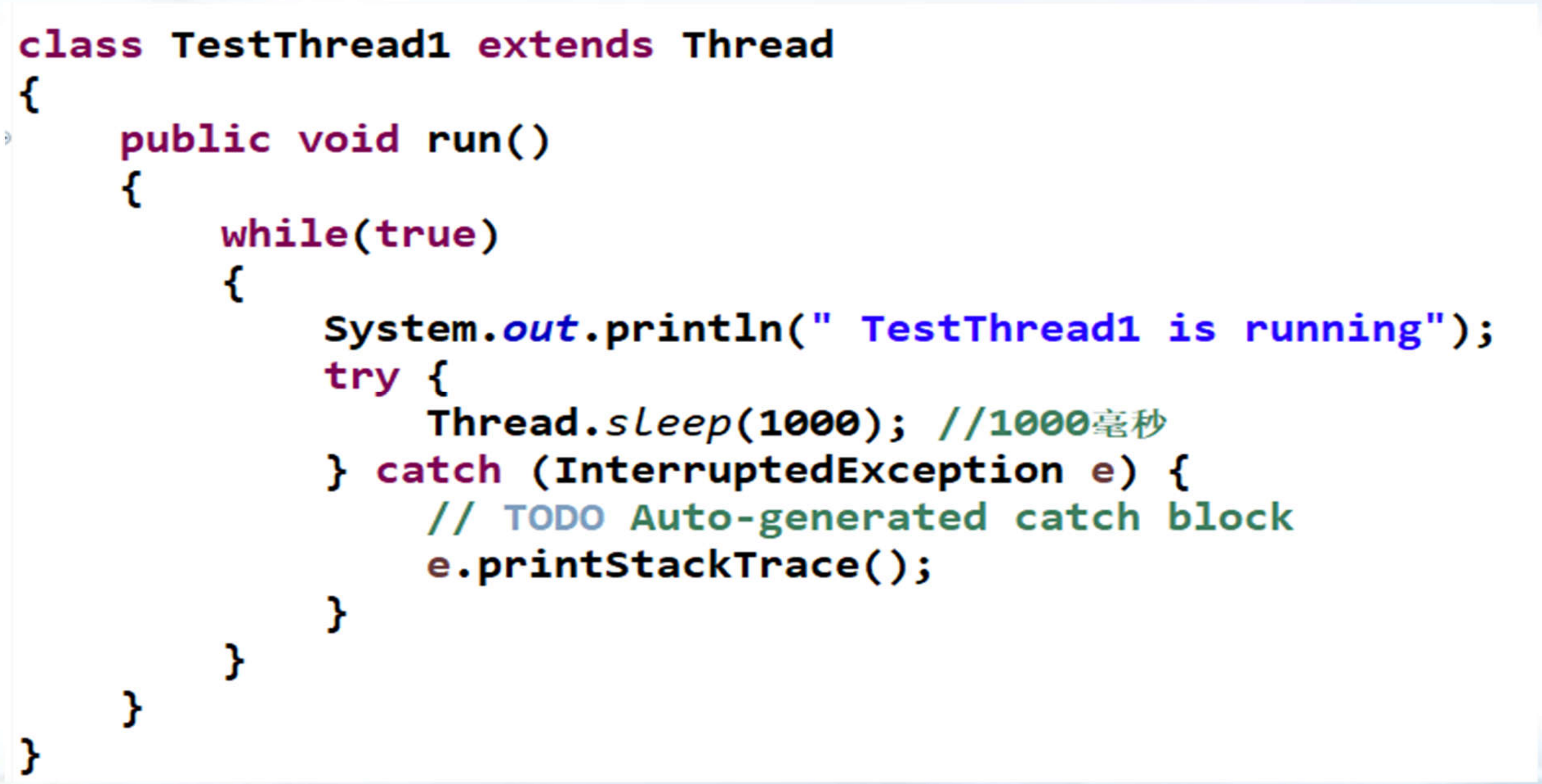
```java
class TestThread0
{
    public void run()
    {
        while(true)
        {
            System.out.println(" TestThread1 is running");
            try {
                Thread.sleep(1000); //1000毫秒
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

代码(5) ThreadDemo1.java



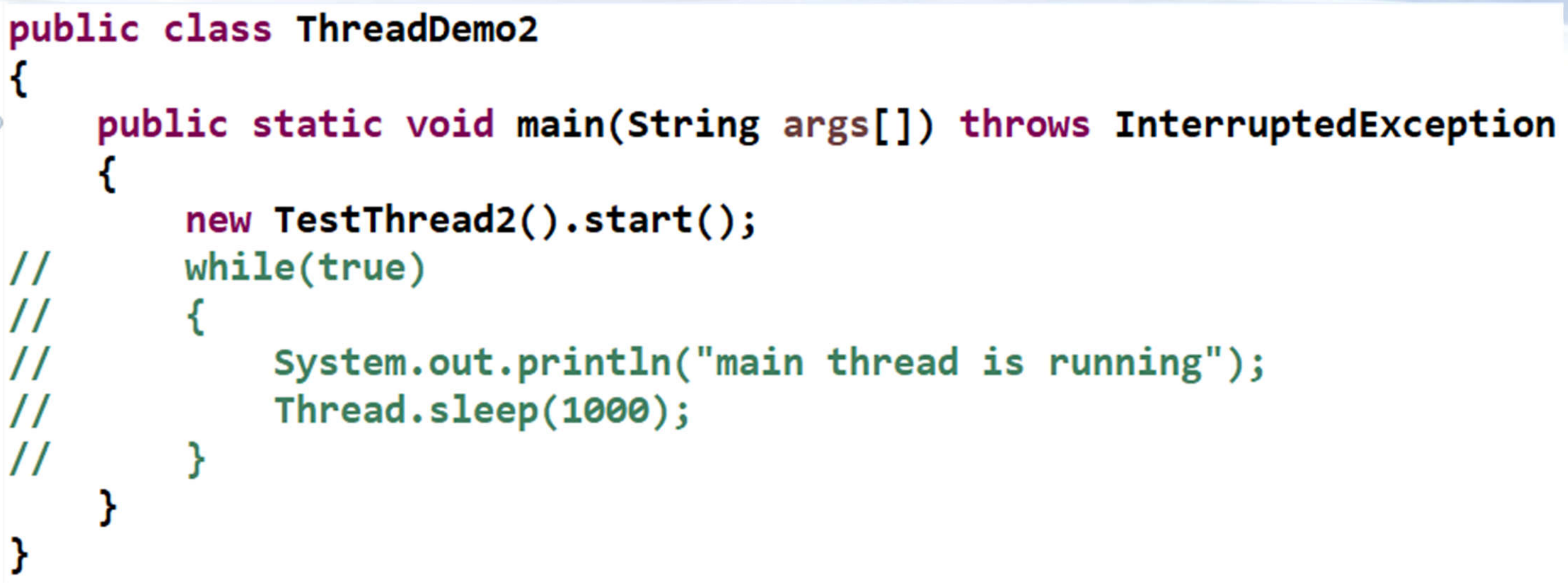
```java
public class ThreadDemo1
{
    public static void main(String args[]) throws Exception
    {
        new TestThread1().start();
        while(true)
        {
            System.out.println("main thread is running");
            Thread.sleep(1000);
        }
    }
}
```

代码(6) TestThread1.java



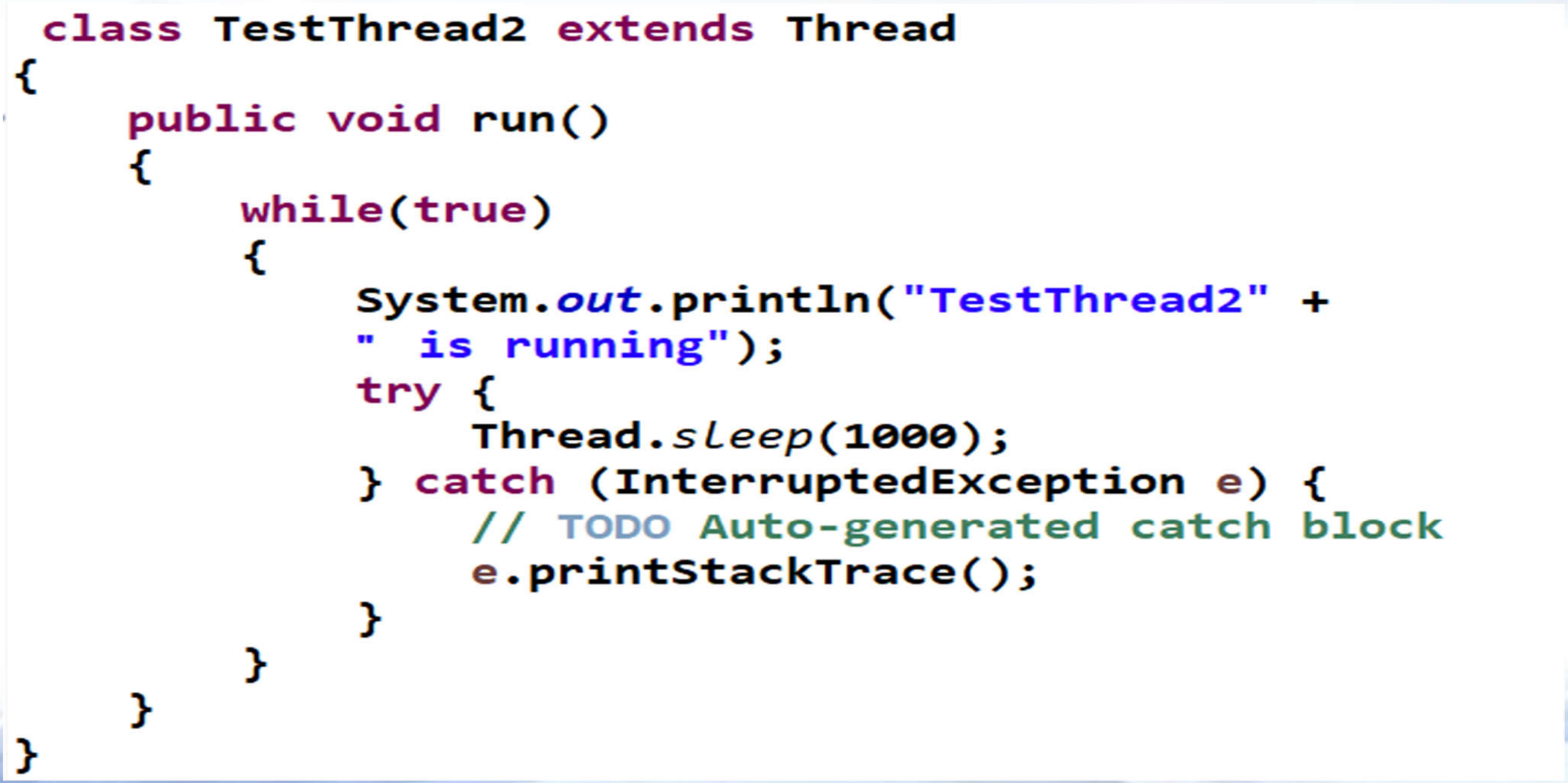
```java
class TestThread1 extends Thread
{
    public void run()
    {
        while(true)
        {
            System.out.println(" TestThread1 is running");
            try {
                Thread.sleep(1000); //1000毫秒
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

代码(7) ThreadDemo2.java



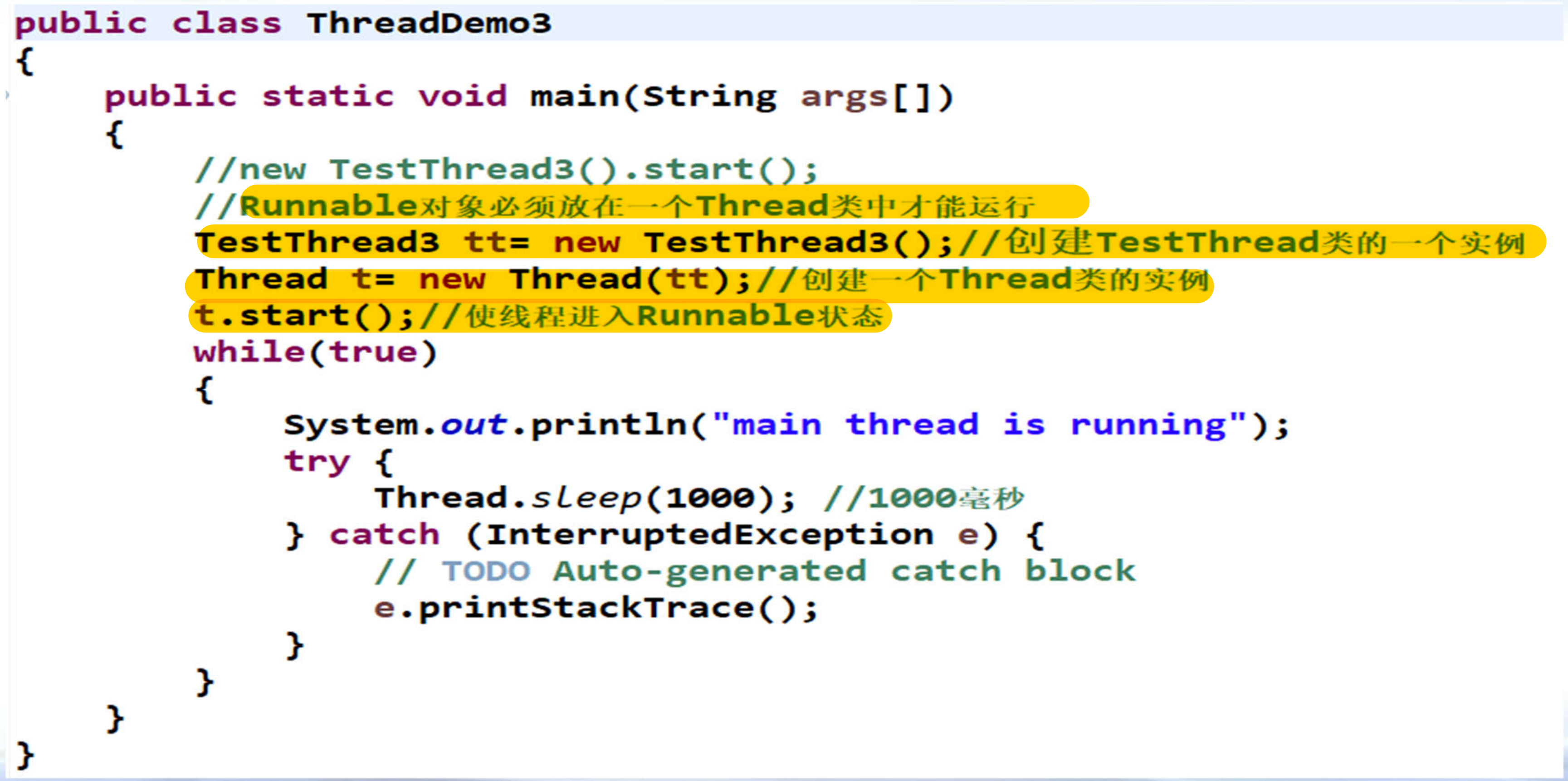
```
public class ThreadDemo2
{
    public static void main(String args[]) throws InterruptedException
    {
        new TestThread2().start();
//      while(true)
//      {
//          System.out.println("main thread is running");
//          Thread.sleep(1000);
//      }
    }
}
```

代码(8) TestThread2.java



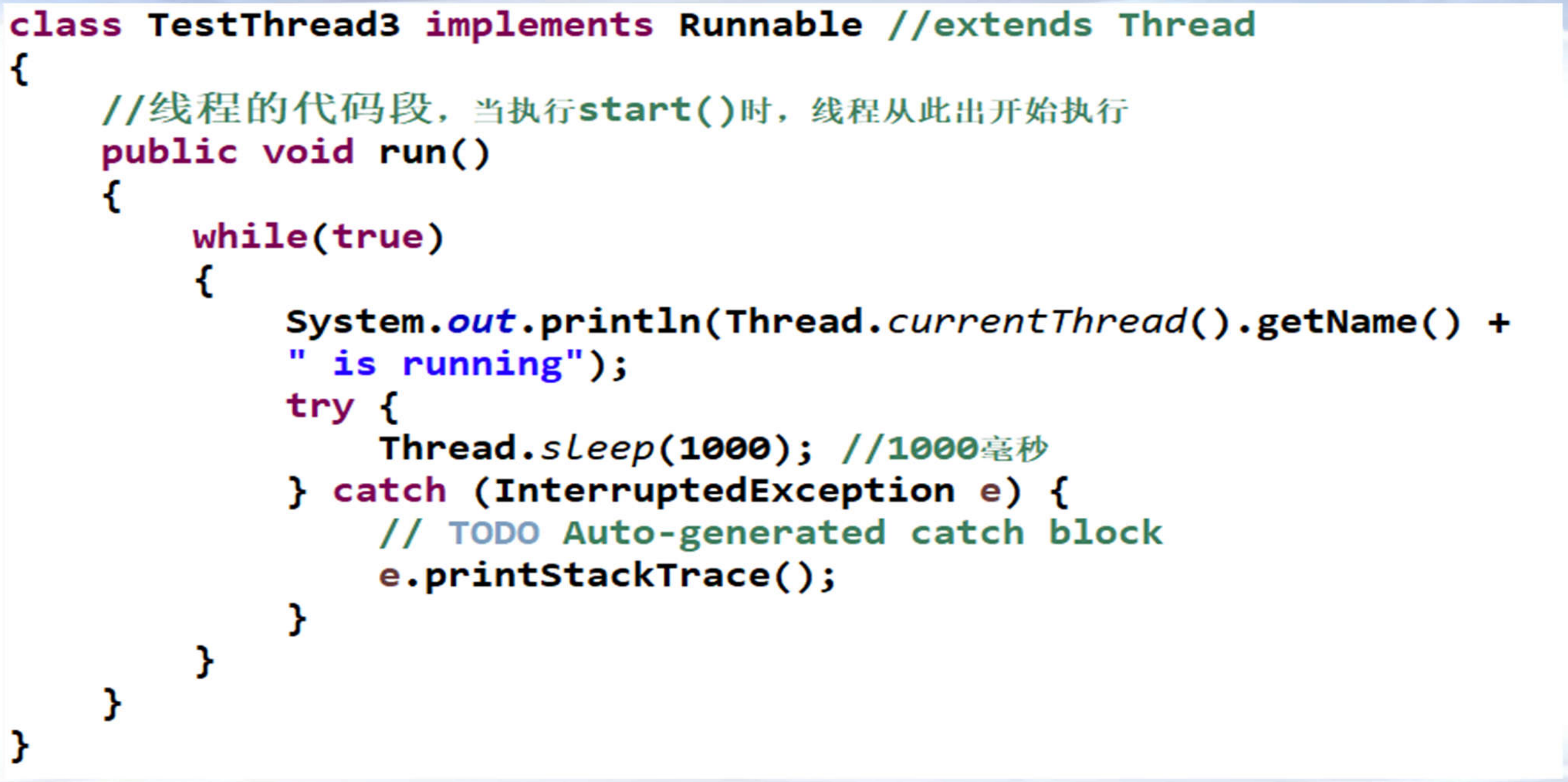
```
 class TestThread2 extends Thread
{
    public void run()
    {
        while(true)
        {
            System.out.println("TestThread2" +
            " is running");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

代码(9) ThreadDemo2.java



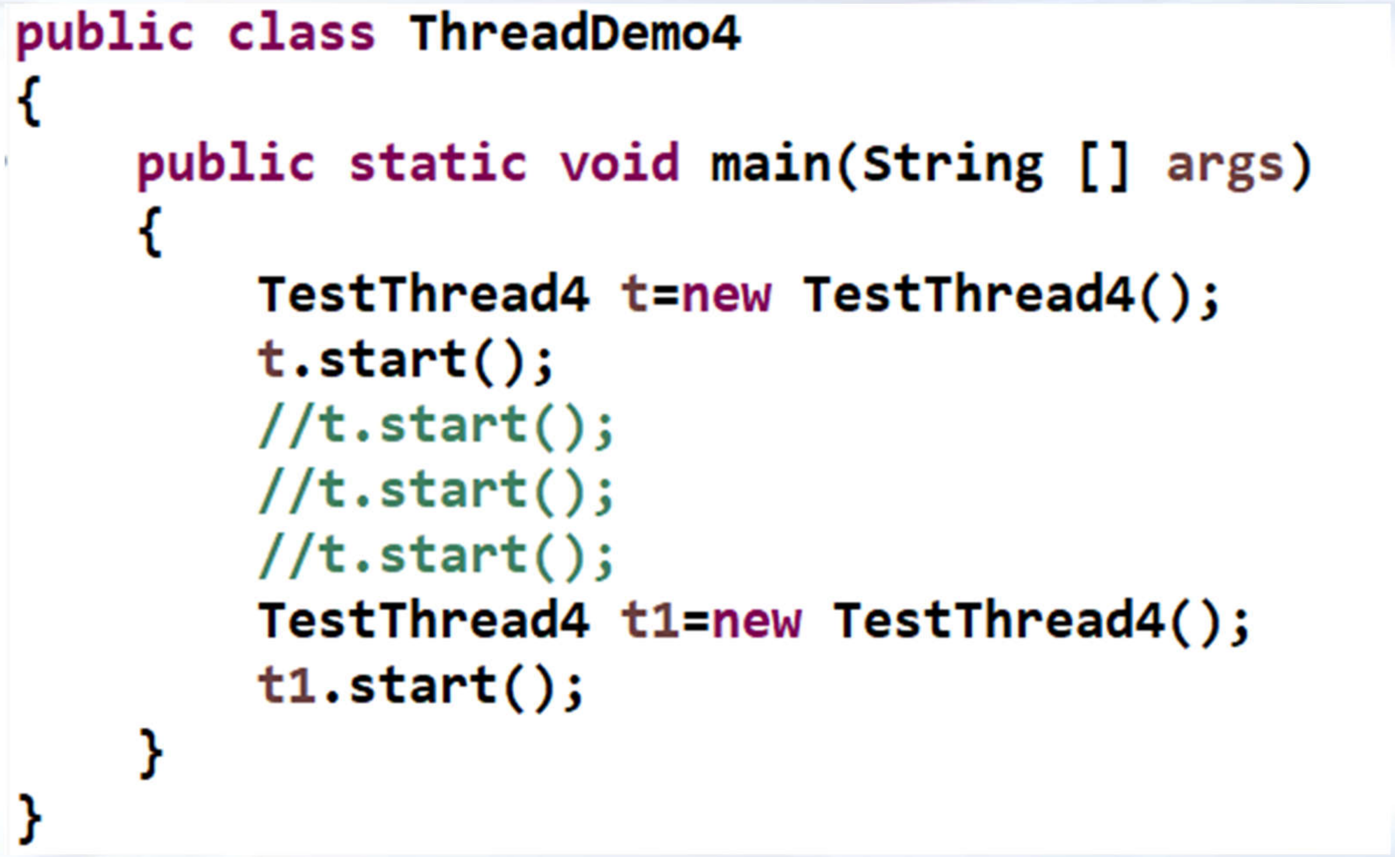
```
public class ThreadDemo3
{
    public static void main(String args[])
    {
        //new TestThread3().start();
        //Runnable对象必须放在一个Thread类中才能运行
        TestThread3 tt= new TestThread3();//创建TestThread类的一个实例
        Thread t= new Thread(tt);//创建一个Thread类的实例
        t.start();//使线程进入Runnable状态
        while(true)
        {
            System.out.println("main thread is running");
            try {
                Thread.sleep(1000); //1000毫秒
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

代码(10) TestThread3.java



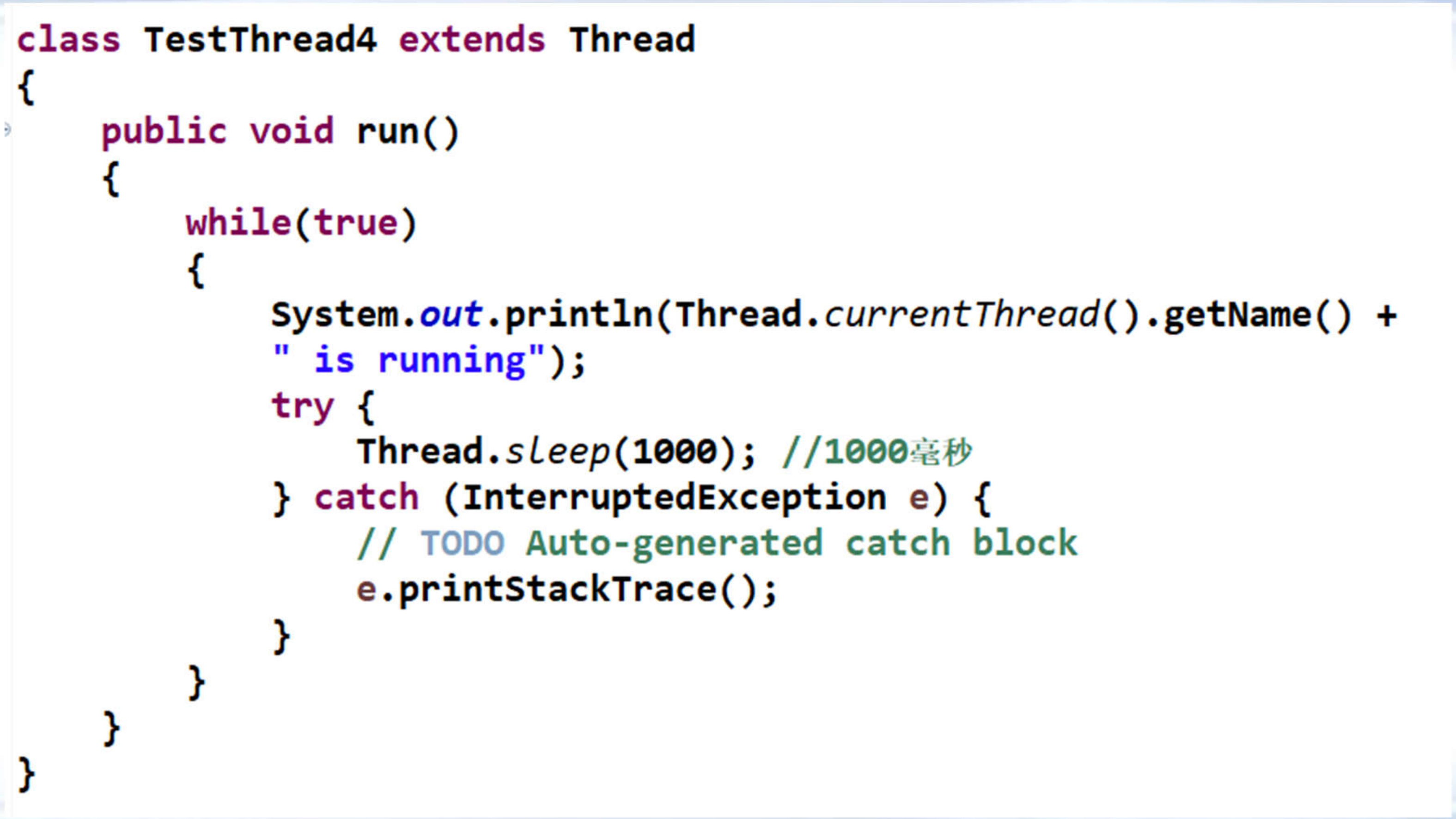
```java
class TestThread3 implements Runnable //extends Thread
{
    //线程的代码段，当执行start()时，线程从此出开始执行
    public void run()
    {
        while(true)
        {
            System.out.println(Thread.currentThread().getName() +
            " is running");
            try {
                Thread.sleep(1000); //1000毫秒
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

代码(11) ThreadDemo4.java



```
public class ThreadDemo4
{
    public static void main(String [] args)
    {
        TestThread4 t=new TestThread4();
        t.start();
        //t.start();
        //t.start();
        //t.start();
        TestThread4 t1=new TestThread4();
        t1.start();
    }
}
```

代码(12) TestThread4.java



```java
class TestThread4 extends Thread
{
    public void run()
    {
        while(true)
        {
            System.out.println(Thread.currentThread().getName() +
            " is running");
            try {
                Thread.sleep(1000); //1000毫秒
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

谢 谢！