****

****

**信息学院软件工程系**

**《JAVA程序设计》实验报告**

实验12

**完成时间：2024.05.22**

**一、实验目的及要求**

* 熟悉文件处理

**二、实验题目及实现过程**

题目一

实现过程:

方法1：继承Thread类

创建MyThread类：继承Thread类并重写run()方法。

在run()方法中：循环从1到10输出线程名和数字。

在main方法中：

创建两个MyThread实例，分别命名为"Thread1"和"Thread2"。

调用start()方法启动两个线程。

方法2：实现Runnable接口

创建MyRunnable类：实现Runnable接口并重写run()方法。

在run()方法中：循环从1到10输出线程名和数字。

在main方法中：创建两个Thread实例，将MyRunnable实例传入构造函数，分别命名为"Thread1"和"Thread2"。调用start()方法启动两个线程。

方法3：使用线程池管理

创建MyRunnable类：实现Runnable接口并重写run()方法。

在run()方法中：循环从1到10输出线程名和数字。

在main方法中：

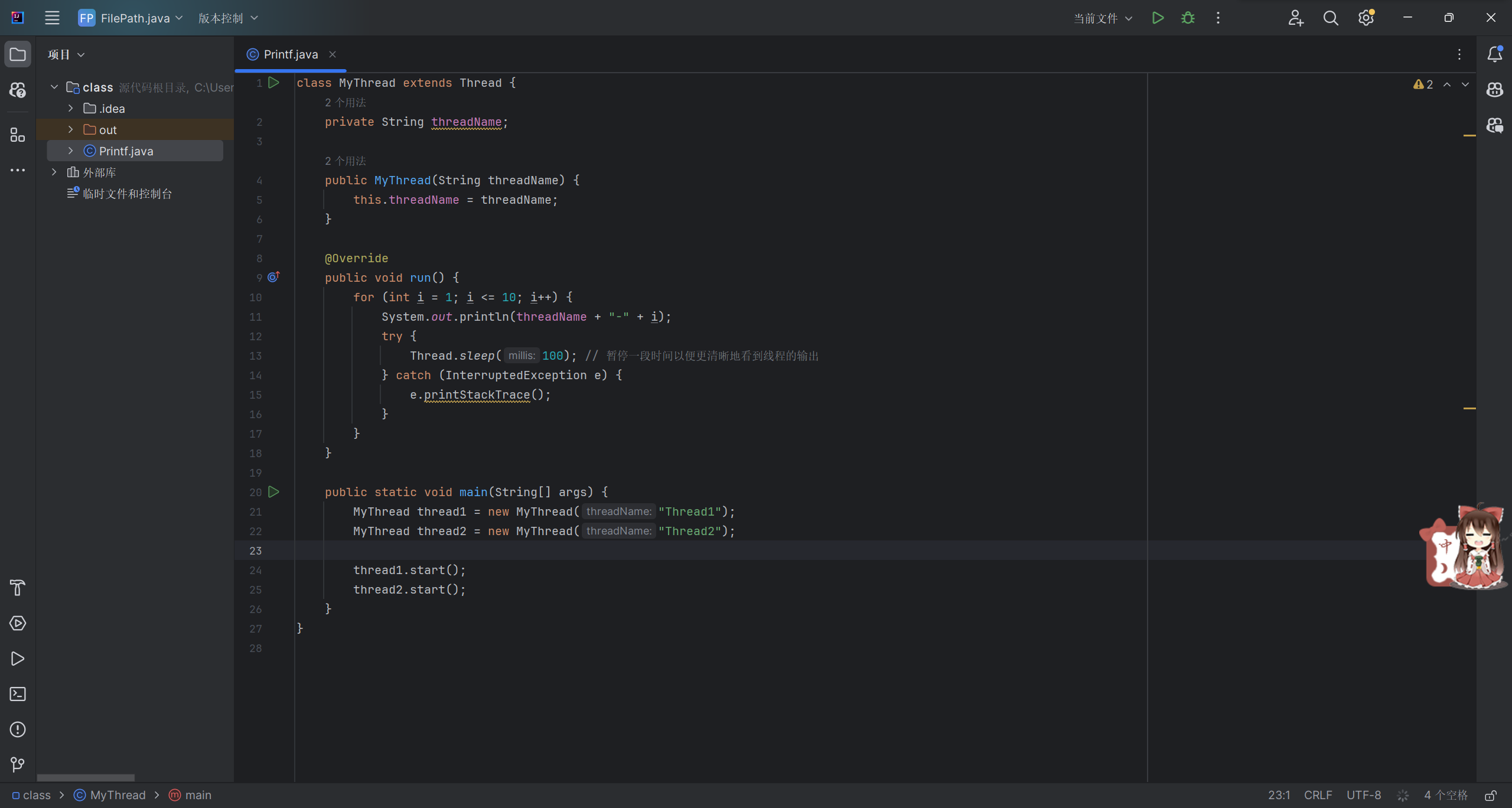
创建一个固定大小为2的线程池。

使用execute()方法提交两个MyRunnable任务，分别命名为"Thread1"和"Thread2"。

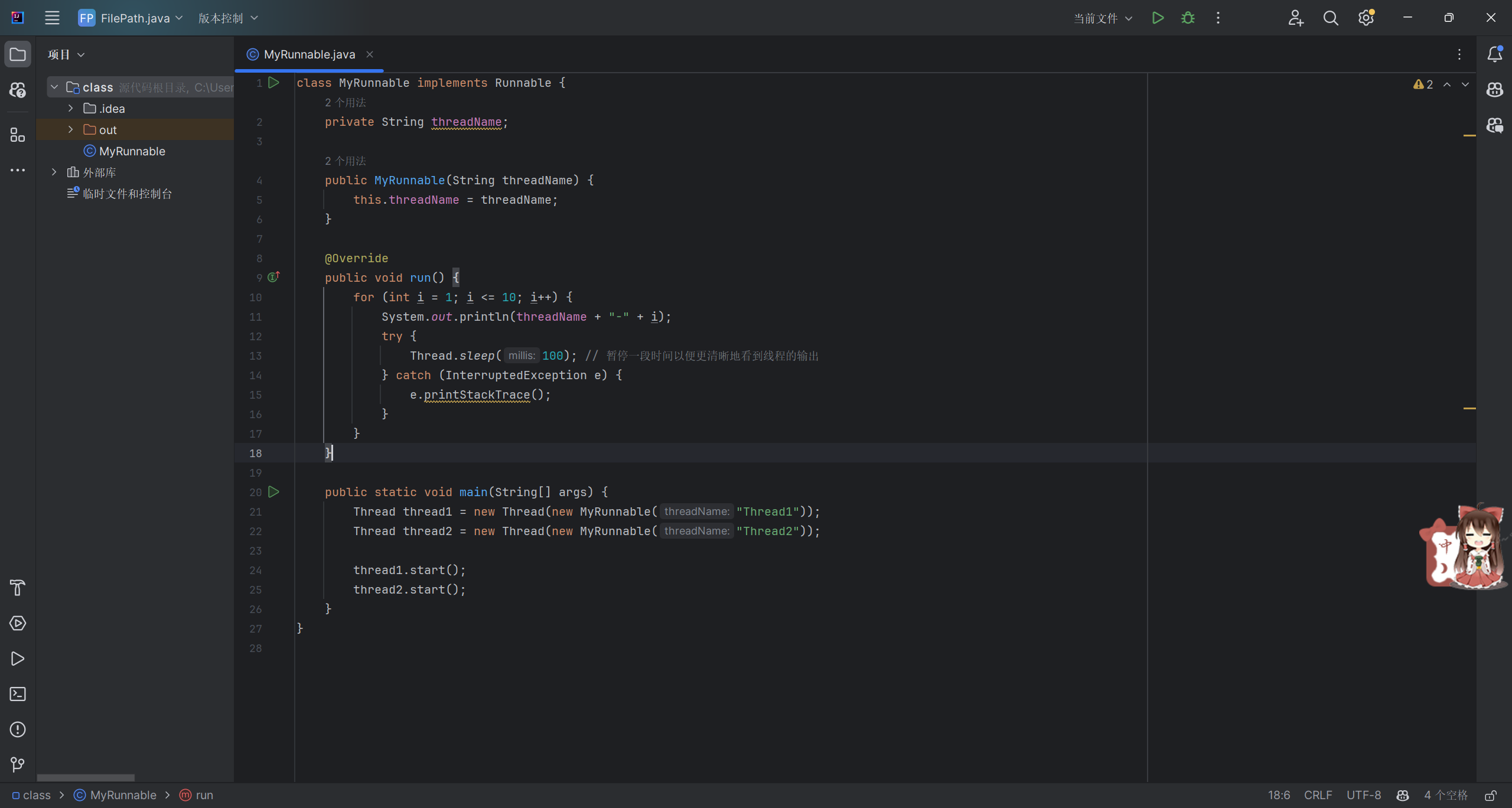
调用shutdown()方法关闭线程池。

全屏截图：

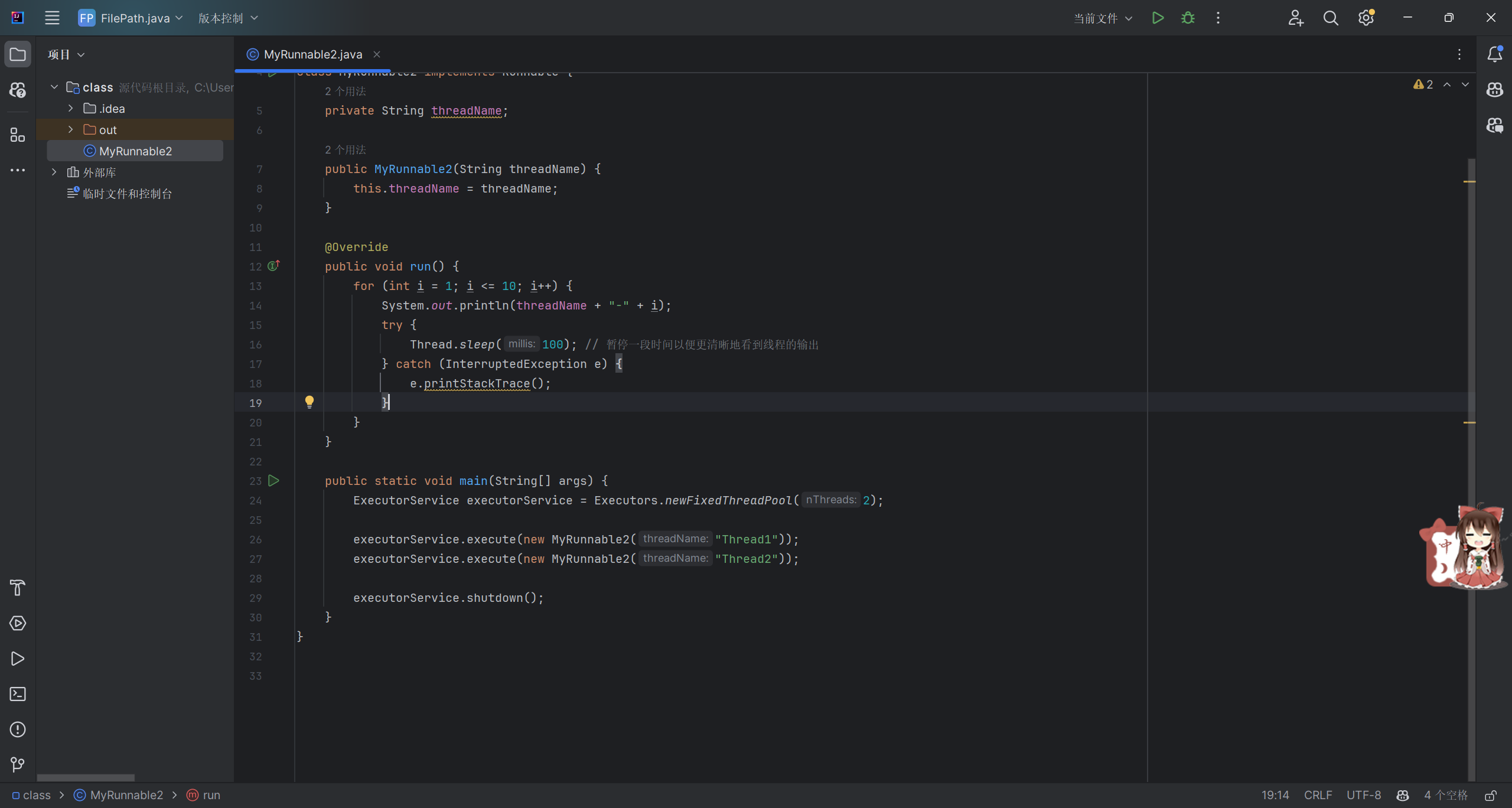
方法一：



方法二：

、

方法三：

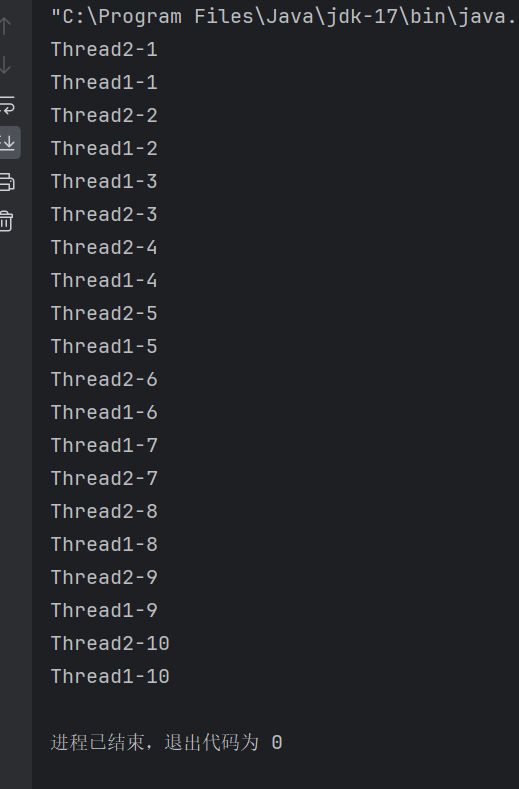


输出结果：

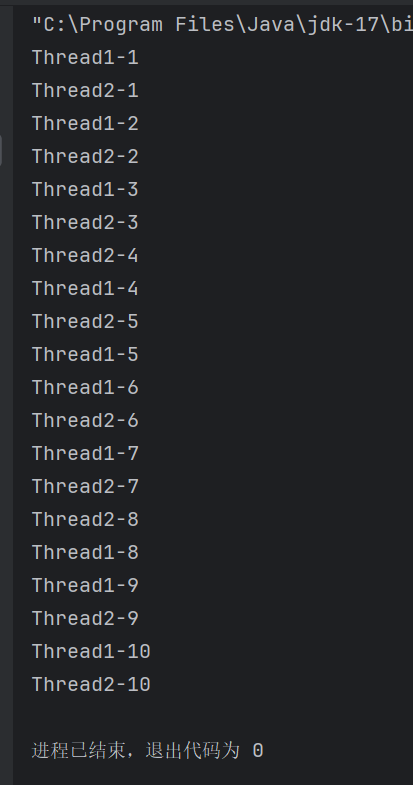
方法一：



方法二：



方法三：

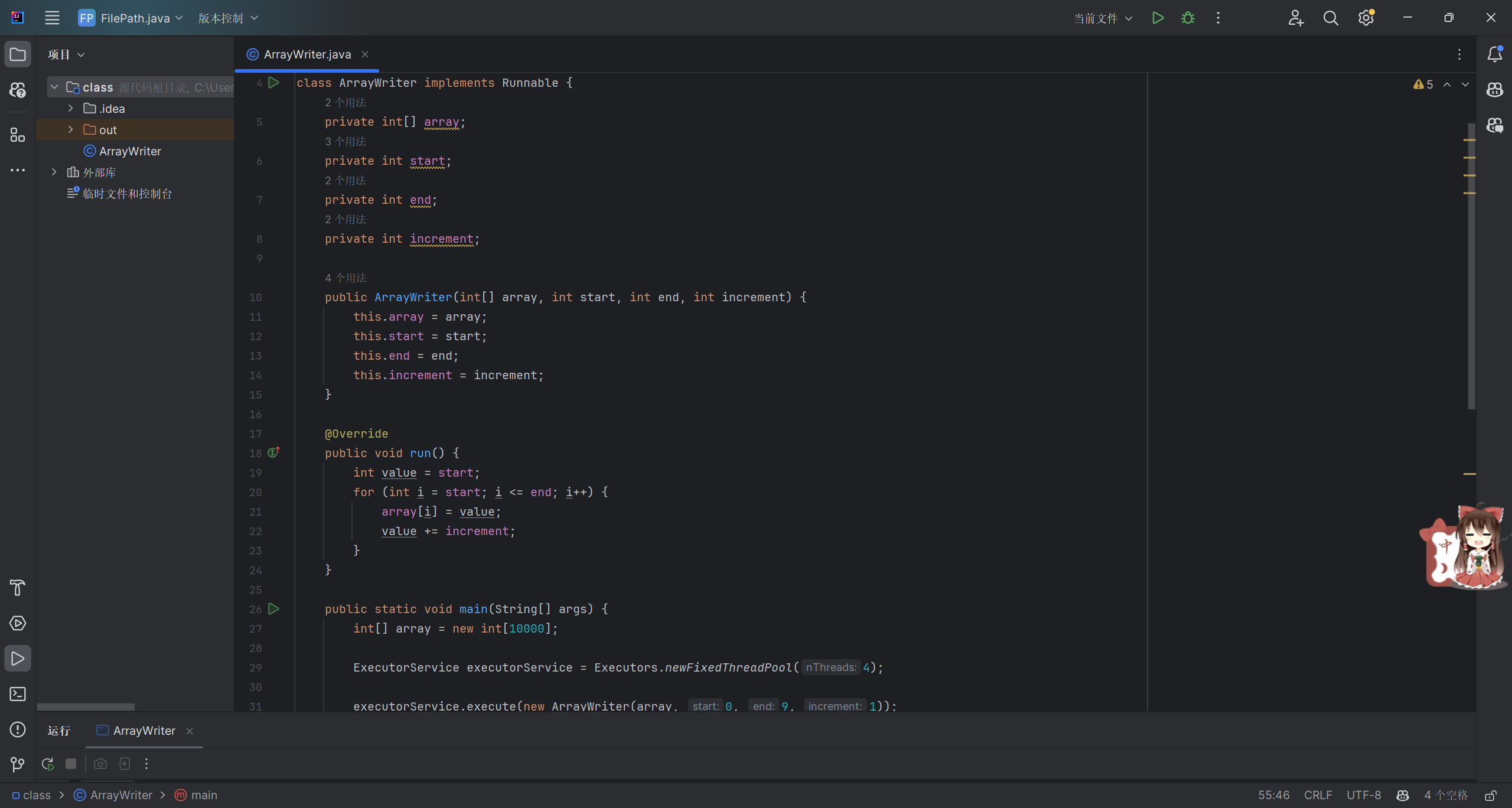
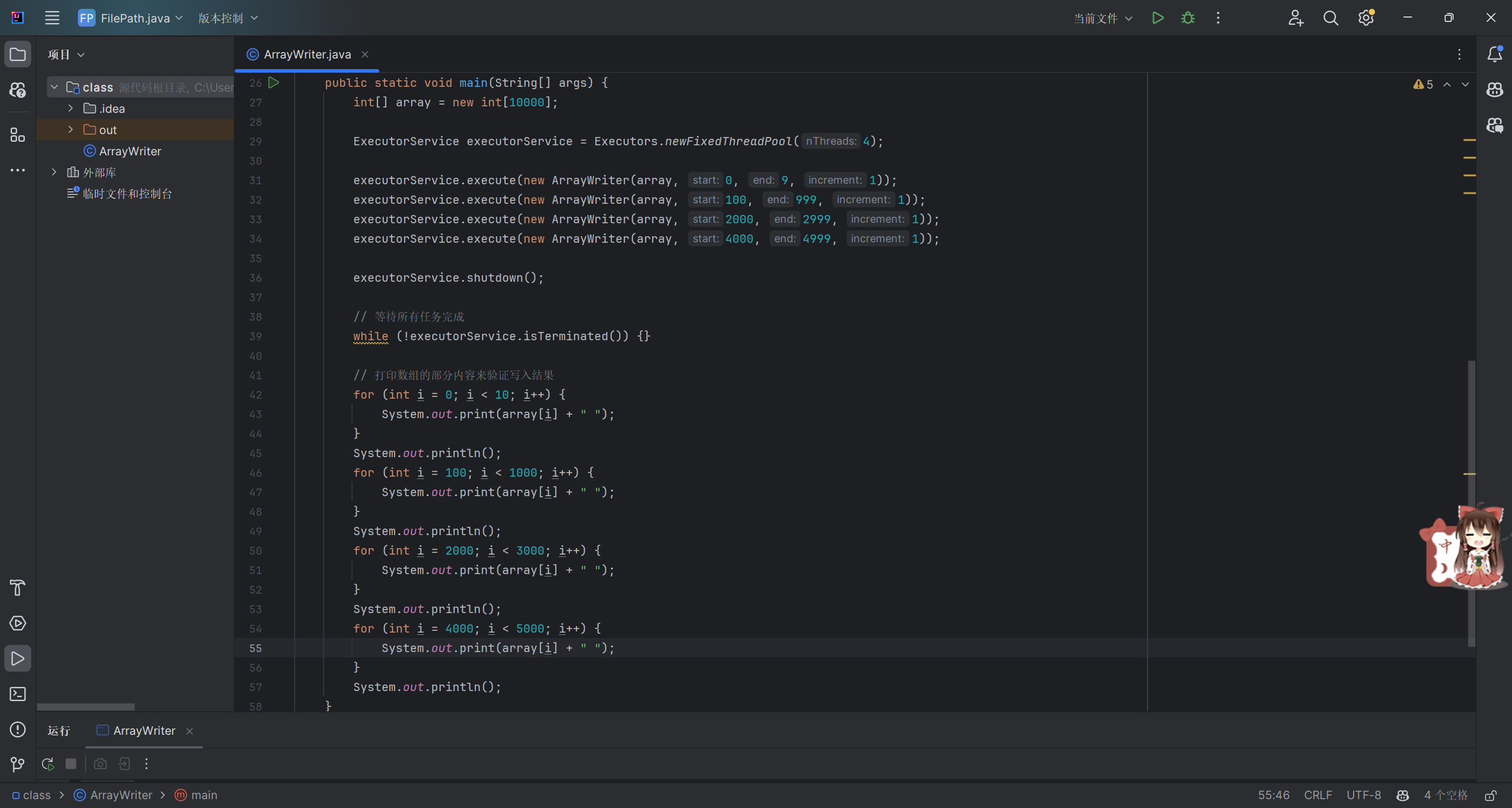


题目二

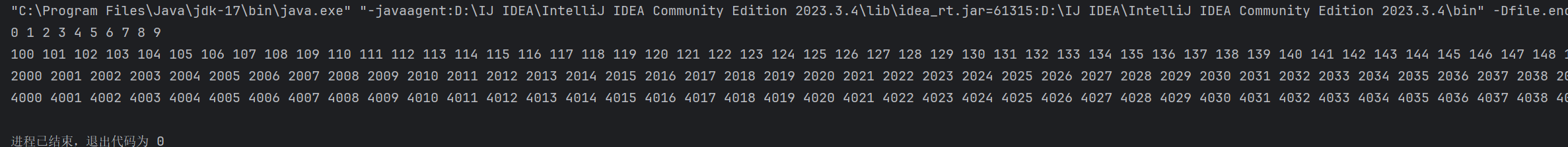
实现过程：

1. 创建ArrayWriter类：实现Runnable接口，包含构造方法和run()方法。
   * 构造方法：接受数组、开始索引、结束索引、增量作为参数。
   * run()方法：在指定范围内按增量值写入数据。
2. 在main方法中：
   * 创建一个长度为10000的数组。
   * 创建一个具有4个线程的固定线程池。
3. 提交任务到线程池：
   * 创建并提交4个ArrayWriter任务，分别写入以下范围：
     + 第一个线程：写入1到10（索引0到9）。
     + 第二个线程：写入100到1000（索引100到999）。
     + 第三个线程：写入2000到3000（索引2000到2999）。
     + 第四个线程：写入4000到5000（索引4000到4999）。
4. 关闭线程池：
   * 调用shutdown()方法关闭线程池。
   * 等待所有任务完成。
5. 验证结果：
   * 打印数组的内容以验证写入结果。

全屏截图：

结果截图：



题目三

实现过程：

方法1：使用 synchronized 关键字实现线程同步

1. 创建主类 CircularBufferSync
   * 定义一个大小为4的缓冲区数组。
   * 定义 count 表示当前缓冲区中的元素数量。
   * 定义 in 和 out 分别表示写入位置和读取位置。
2. 创建 put 方法
   * 使用 synchronized 关键字同步方法。
   * 如果缓冲区已满，调用 wait 方法等待。
   * 将值写入缓冲区，并更新写入位置。
   * 更新元素数量，调用 notifyAll 唤醒其他线程。
3. 创建 get 方法
   * 使用 synchronized 关键字同步方法。
   * 如果缓冲区为空，调用 wait 方法等待。
   * 从缓冲区读取值，并更新读取位置。
   * 更新元素数量，调用 notifyAll 唤醒其他线程。
4. 创建 Producer 类实现 Runnable 接口
   * 在 run 方法中生成 1 到 10 的随机数，并调用 put 方法写入缓冲区。
5. 创建 Consumer 类实现 Runnable 接口
   * 在 run 方法中调用 get 方法从缓冲区读取值，并打印出来。
6. 在 main 方法中
   * 创建 Producer 和 Consumer 线程并启动。

方法2：使用 java.util.concurrent.ArrayBlockingQueue

创建主类 CircularBufferQueue

使用 ArrayBlockingQueue 创建一个大小为4的缓冲区。

创建 Producer 类实现 Runnable 接口

在 run 方法中生成 1 到 10 的随机数，并调用 buffer.put 方法写入缓冲区。

创建 Consumer 类实现 Runnable 接口

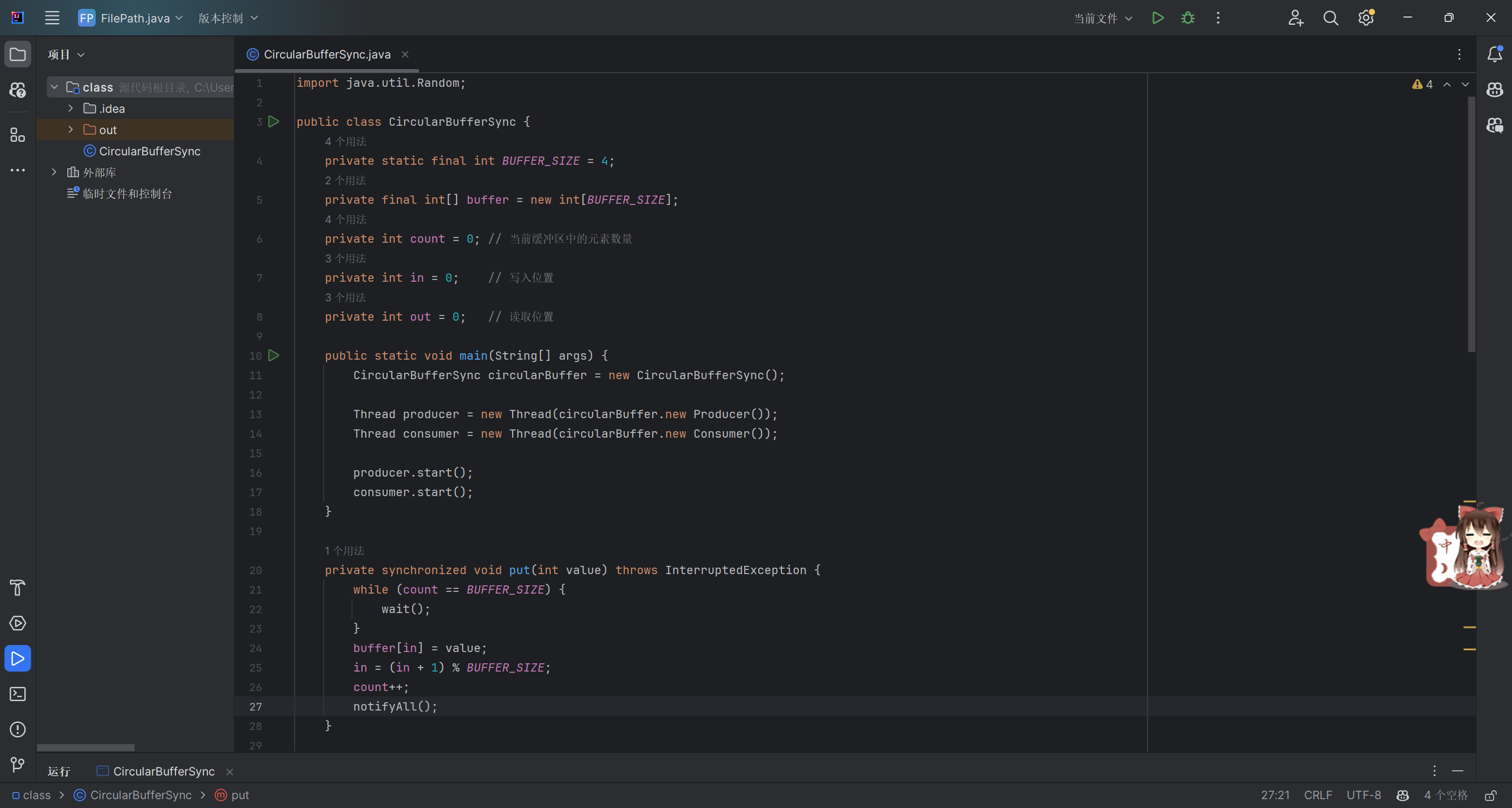
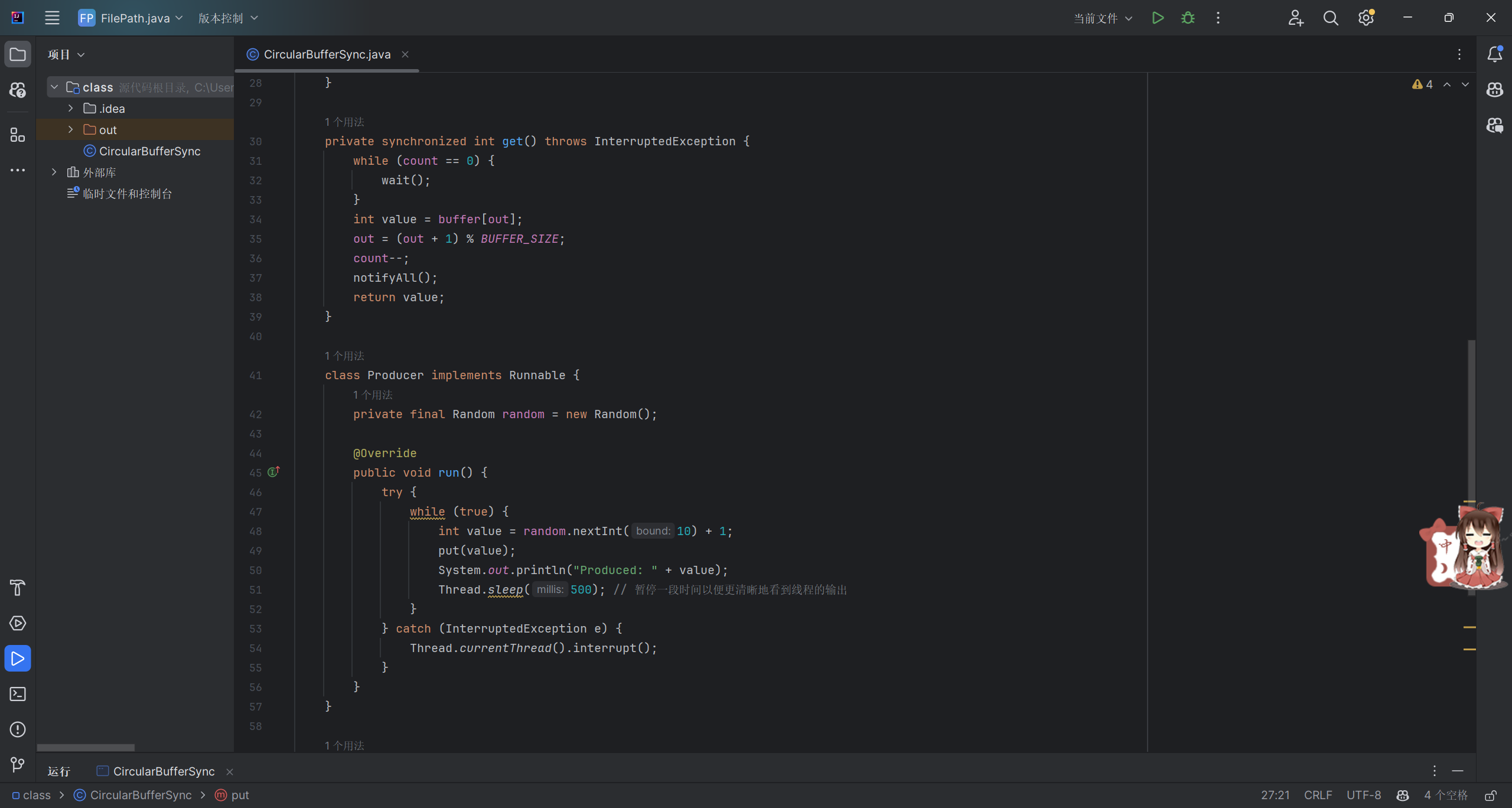
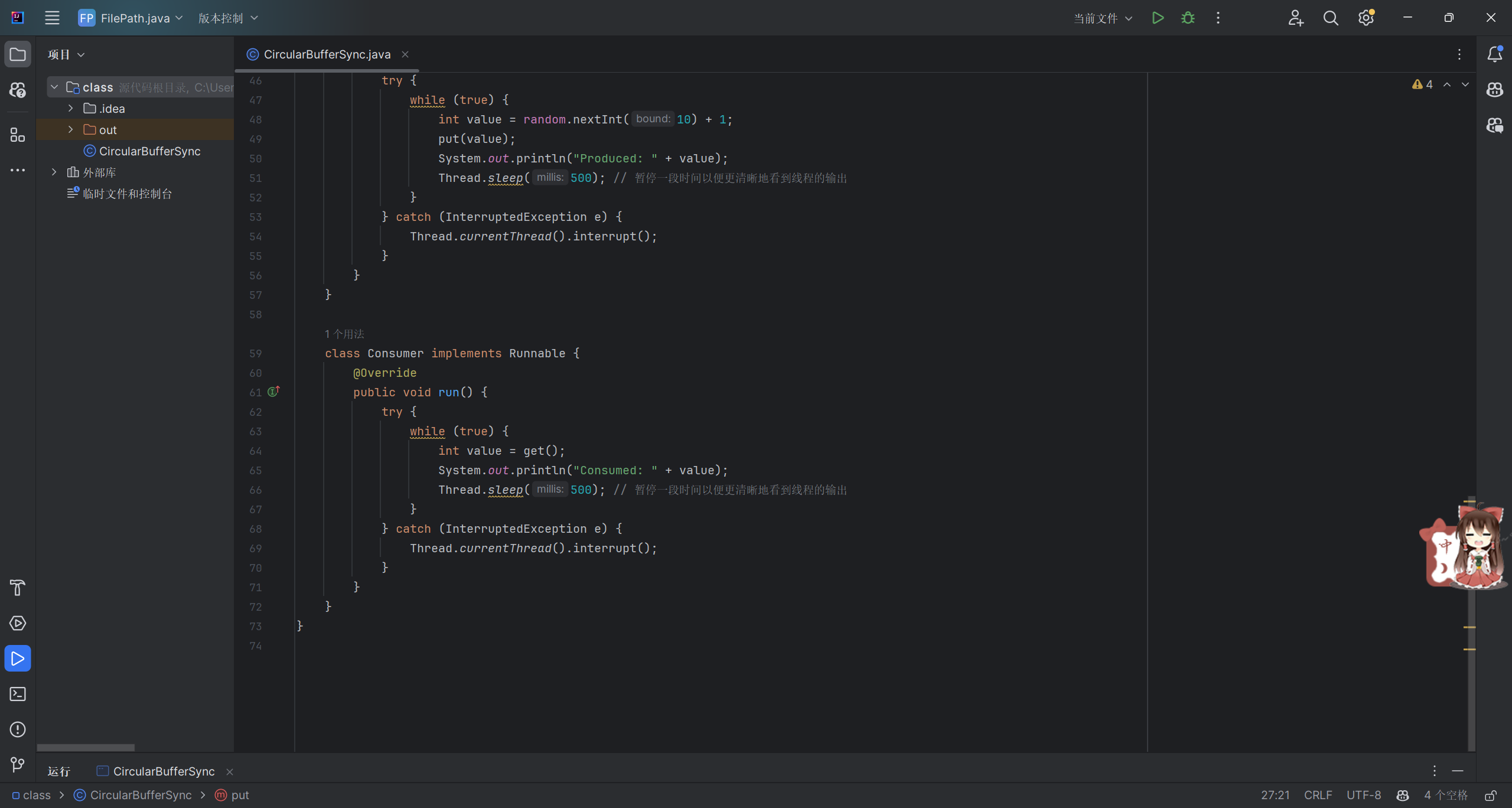
在 run 方法中调用 buffer.take 方法从缓冲区读取值，并打印出来。

在 main 方法中

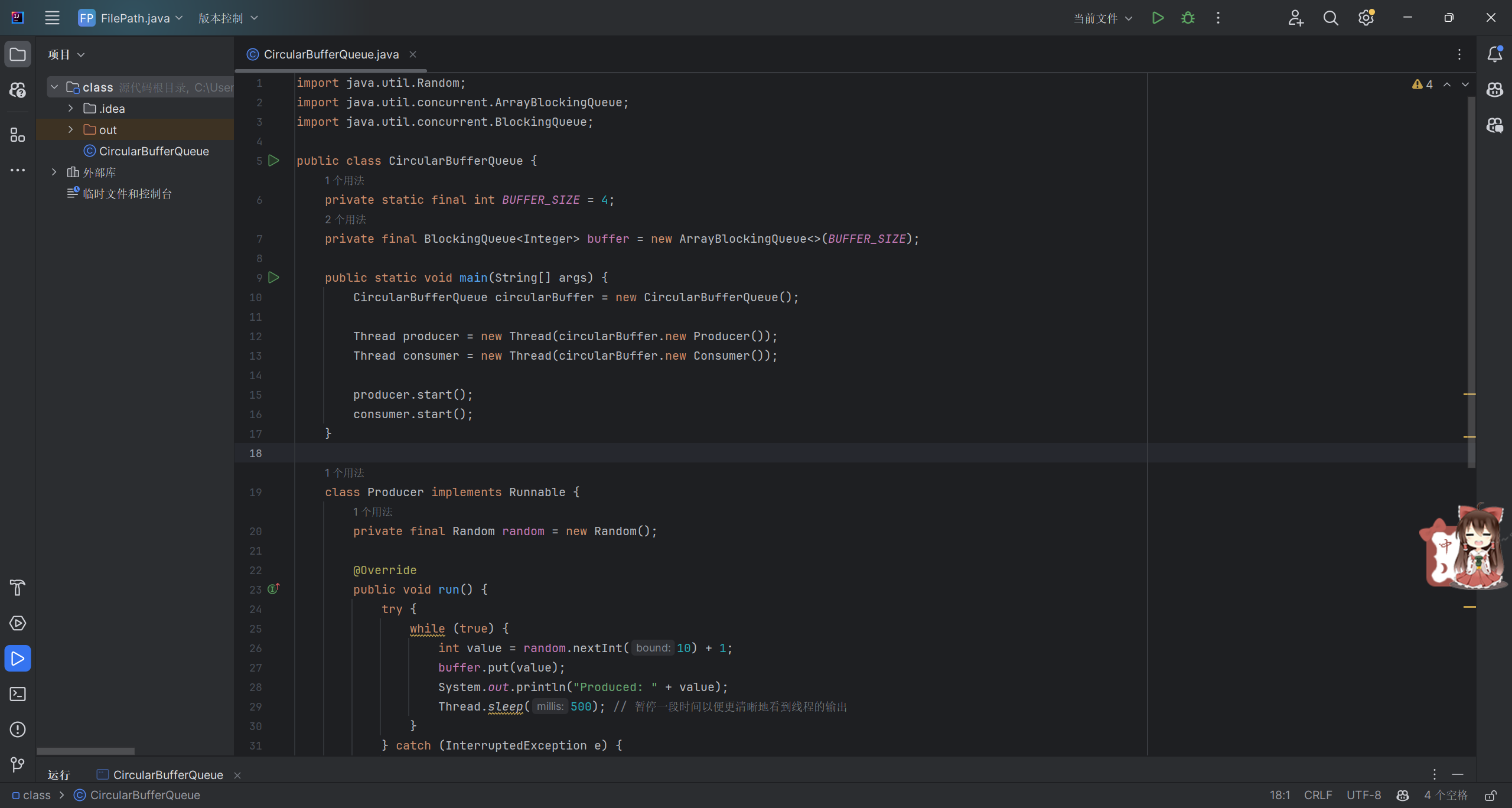
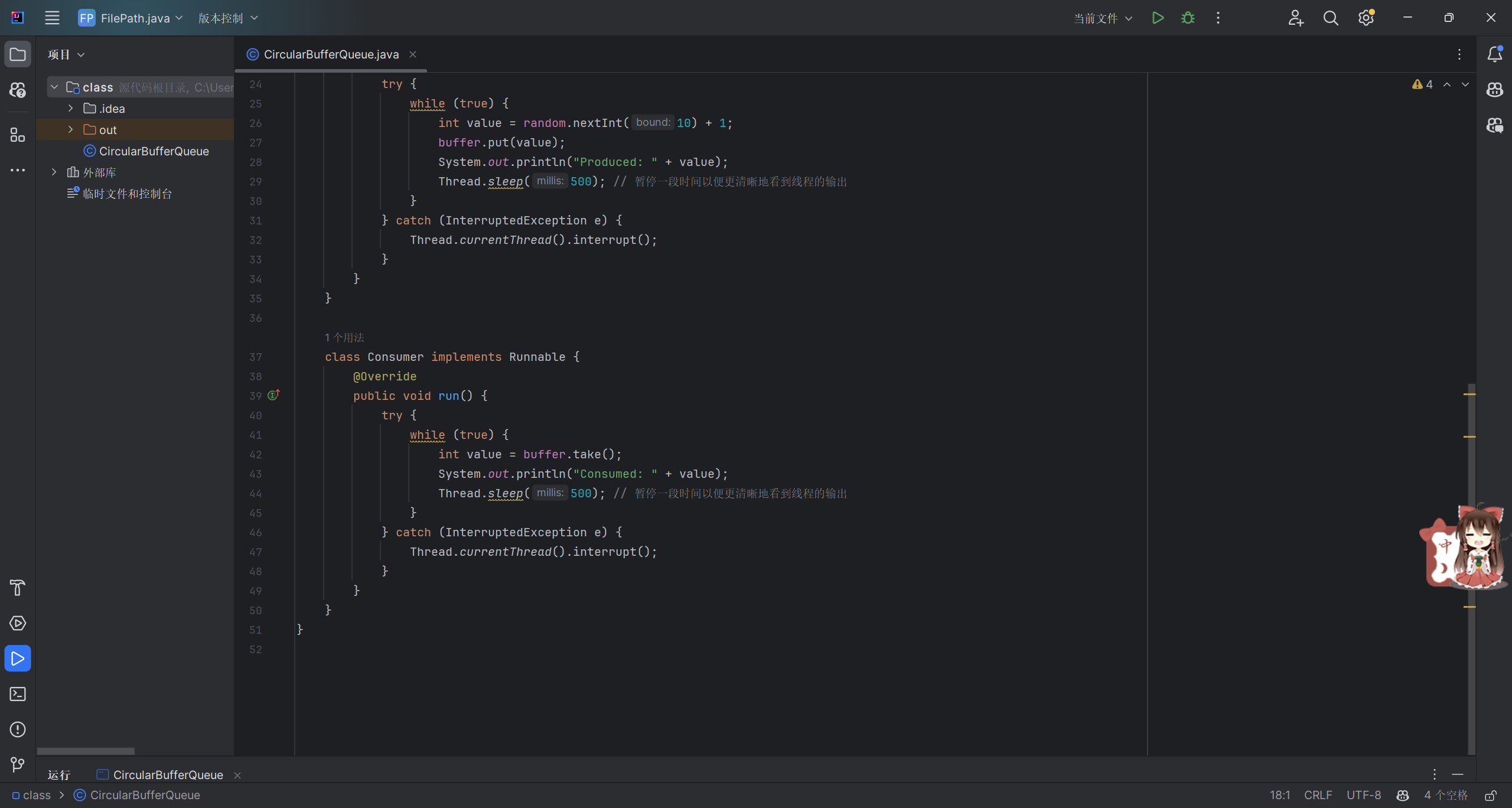
创建 Producer 和 Consumer 线程并启动。

全屏截图：

方法一：

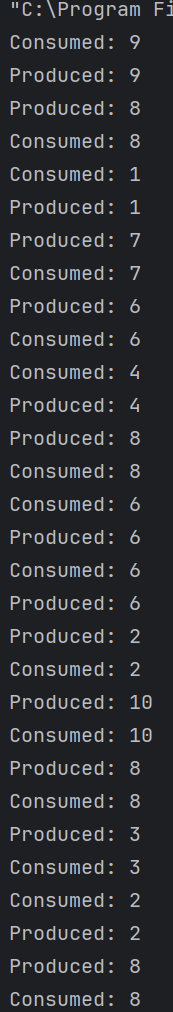
  

方法二：

结果截图：

方法一：



方法二：



**三、实验总结与心得记录**

在书写java代码时，发现java对异常处理的要求很严格。当函数可能存在异常时，代码段中必须有捕捉这一异常的相关代码。