****

****

**信息学院软件工程系**

**《JAVA程序设计》实验报告**

实验十二

**姓名：任宇**

**学号：33920212204567**

**学院：信息学院**

**专业：软件工程**

**完成时间：2024-05-24**

1. **实验目的及要求**

* 熟悉多线程编程

1. **实验题目及实现过程**
2. 题目一：

用三种方法完成：2个线程同时循环输出“线程名-数字”，其中数字从1变化到10。

1. 实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK版本：JDK17

1. 实现过程

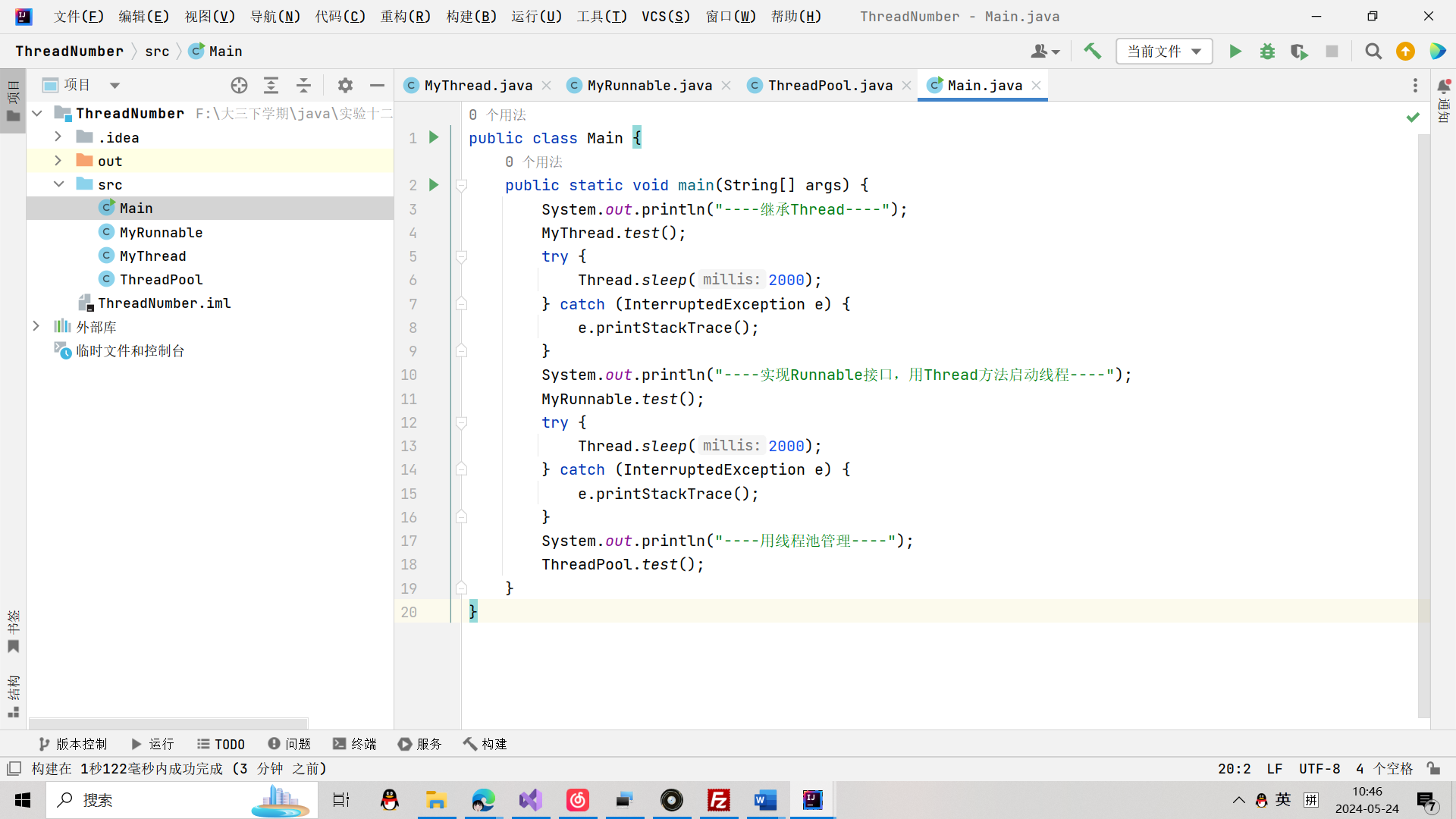
总共设计四个类：

* MyThread：继承自Thread类，实现第一个方法。
* MyRunnable：实现Runnable接口，实现第二个方法。
* ThreadPool：实现Runnable接口，配合线程池，实现第三个方法。
* Main：主类，依次调用上述三种方法。

对于每个类都重写run方法，实现循环输出线程名和数字。

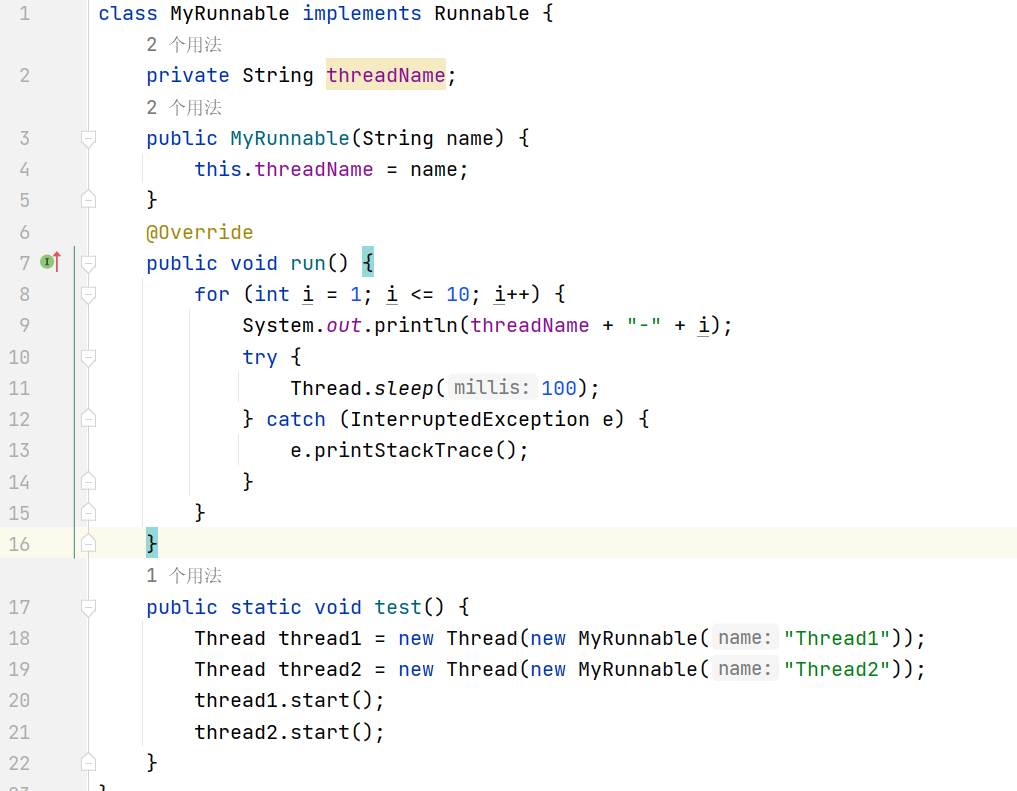
程序的调用关系为：Main类作为主程序入口，依次调用三个不同的方法，每次调用之间暂停一段时间。MyThread类继承自Thread类，直接调用start方法启动线程。MyRunnable类实现Runnable接口，通过Thread类的构造方法创建线程并启动。ThreadPool类实现Runnable接口，通过线程池管理线程的创建和执行。

1. 过程截图
2. 全屏截图：

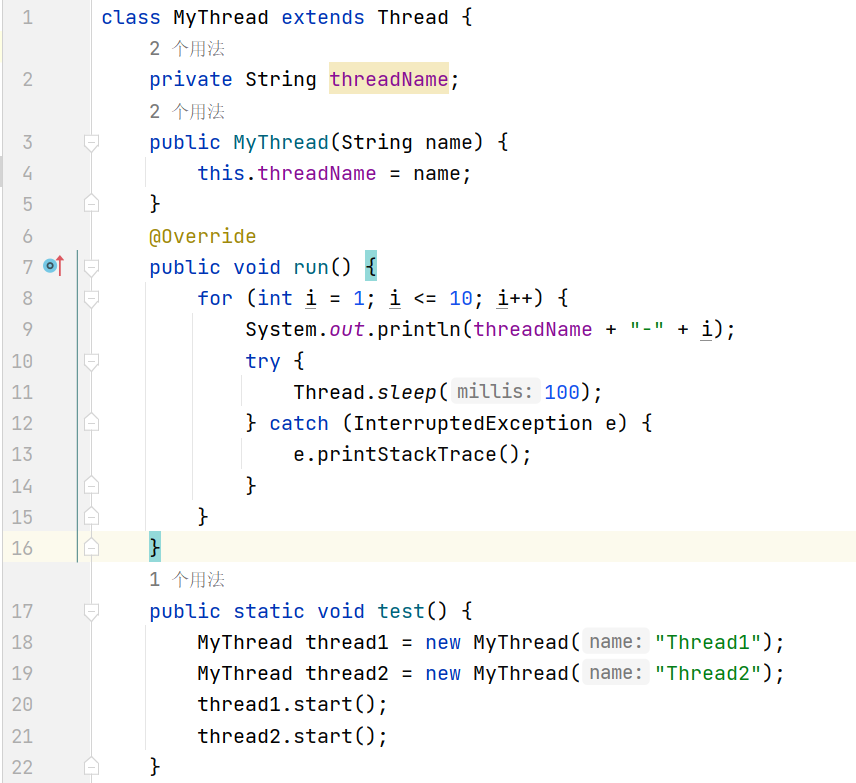


1. 主要代码：

MyRunnable类：



MyThread类：



ThreadPool类：

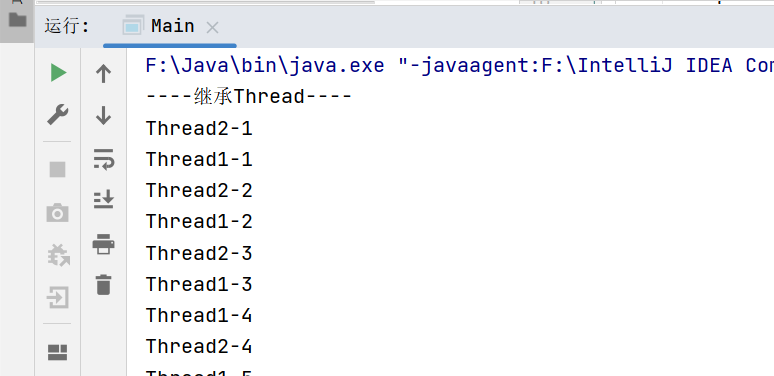


Main方法：

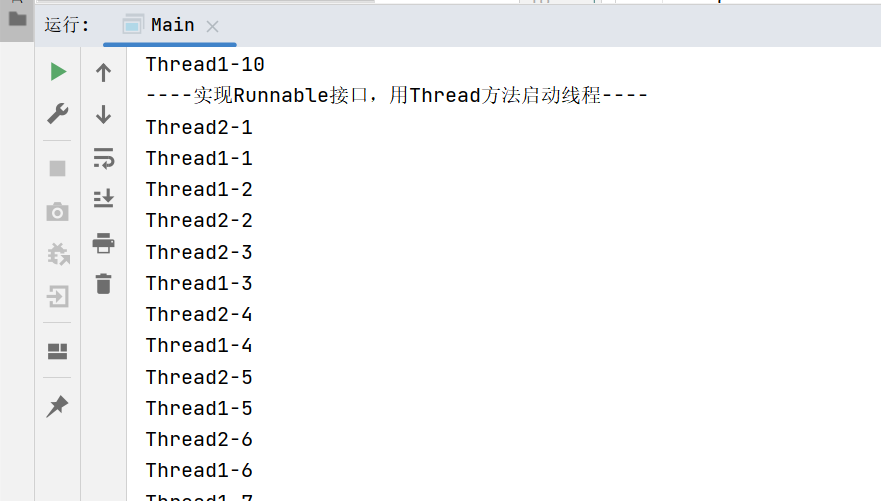


1. 运行结果：

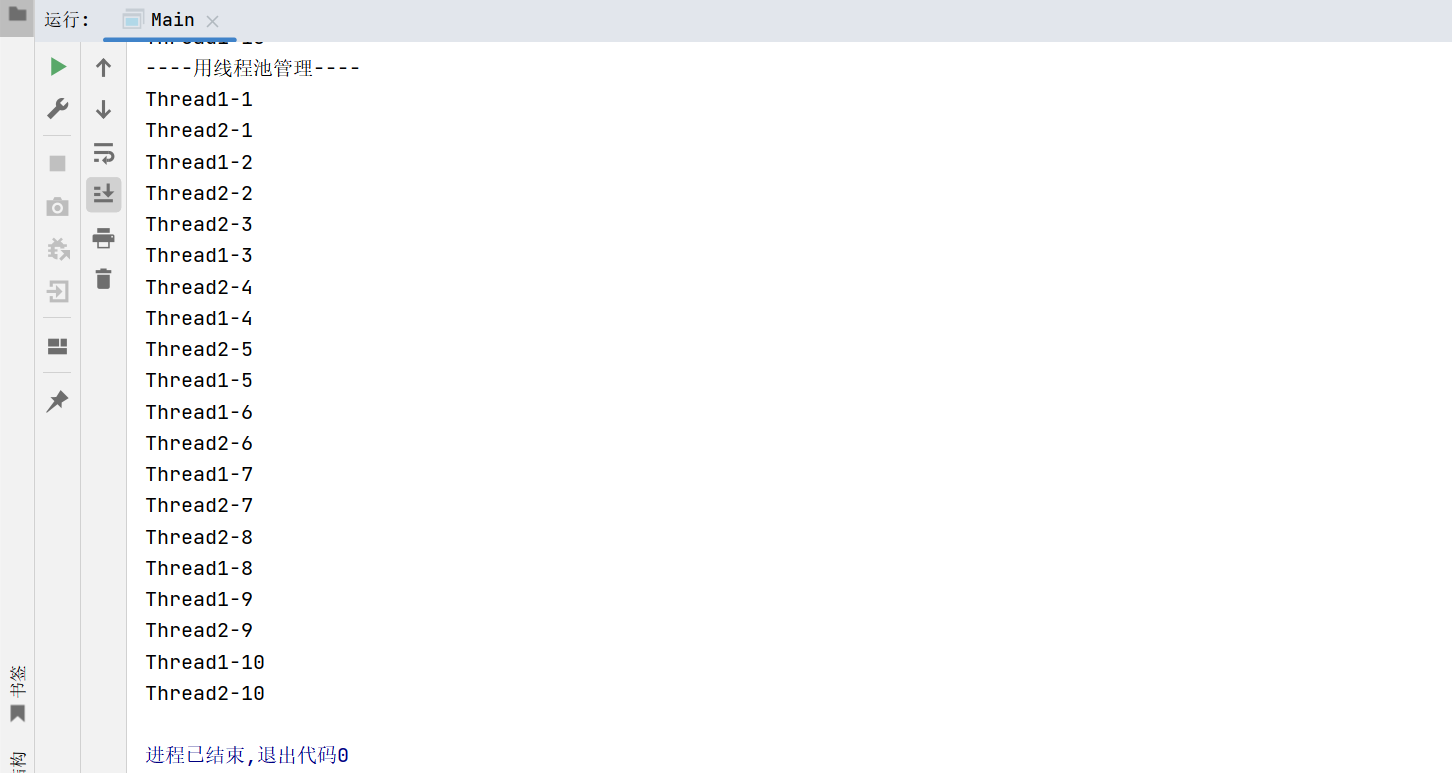
继承自Thread类：



实现Runnable接口，用Thread方法启动线程：



使用线程池管理：



1. 题目二：

编程一个程序，4个线程同时向一个10000位的数组中分别写入1-10,100-1000,2000-3000,4000-5000的数。

1. 实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK版本：JDK17

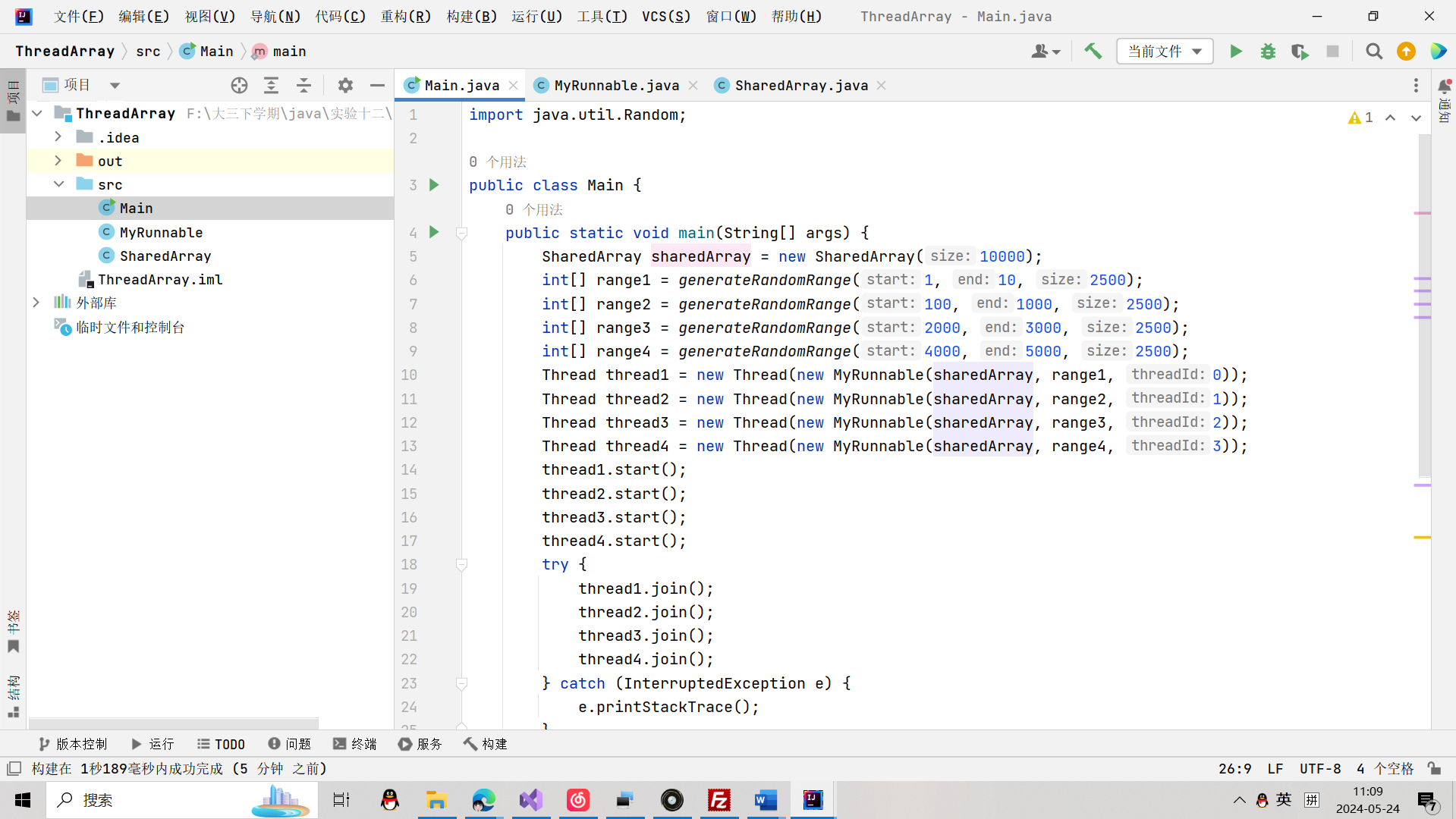
1. 实现过程

总共设计了三个类，分别为：SharedArray类、MyRunnable类和Main类。程序实现创建一个共享的数组和四个线程，每个线程负责向数组中写入特定范围的数字。类的具体说明如下：

* SharedArray类
* array：存储数据的数组。
* currentIndex：当前写入数组的位置。
* lock：用于控制线程访问共享资源的锁。
* conditions：用于控制线程执行顺序的条件变量数组。
* currentThread：当前执行的线程ID。
* writeNumber(int number, int threadId)：线程调用此方法写入数组。方法确保只有当前线程ID的线程可以写入数据，其它线程等待。
* printArray()：打印数组内容。
* MyRunnable类
* sharedArray：共享数组的引用。
* numbers：该线程负责写入的数字数组。
* threadId：线程ID。
* run()：循环调用sharedArray.writeNumber(number, threadId)方法将数字写入数组。
* Main类

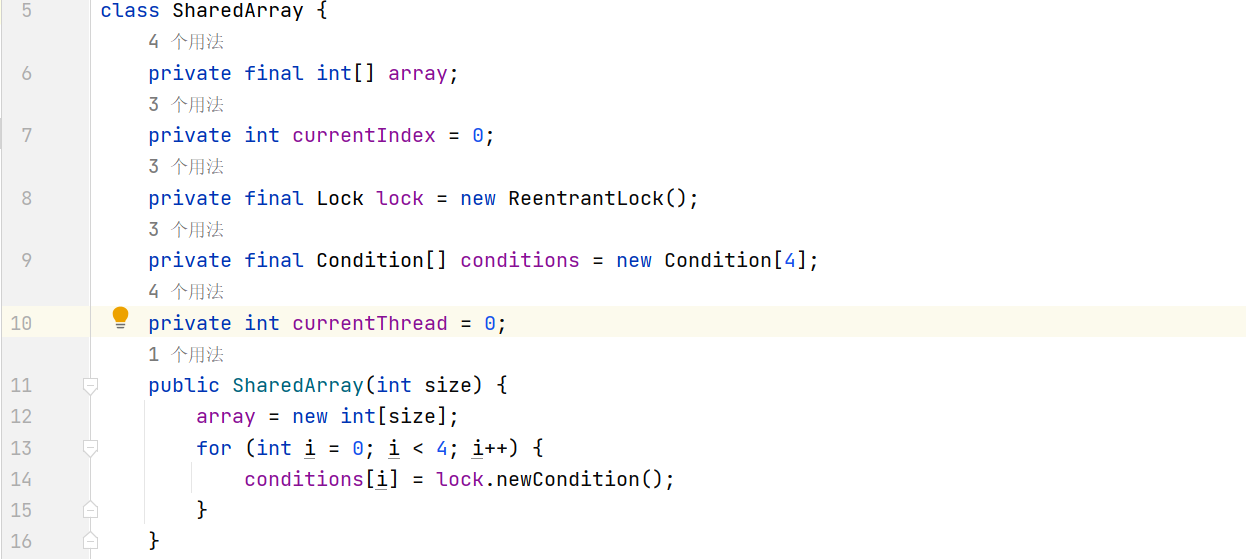
首先创建一个SharedArray实例。接着使用generateRandomRange方法生成四个不同范围内的随机数数组。然后创建并启动四个线程，每个线程负责写入一个范围的随机数。最后等待所有线程完成后打印数组内容。

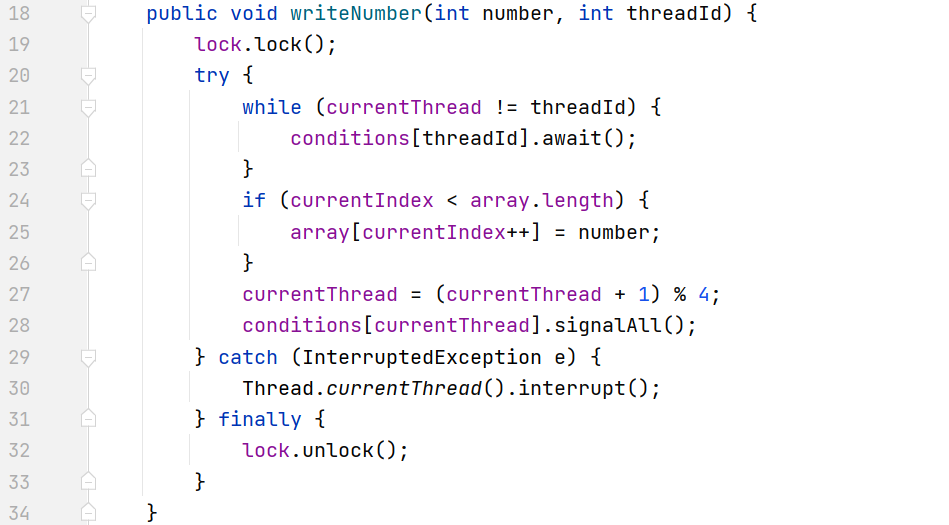
1. 过程截图
2. 全屏截图：



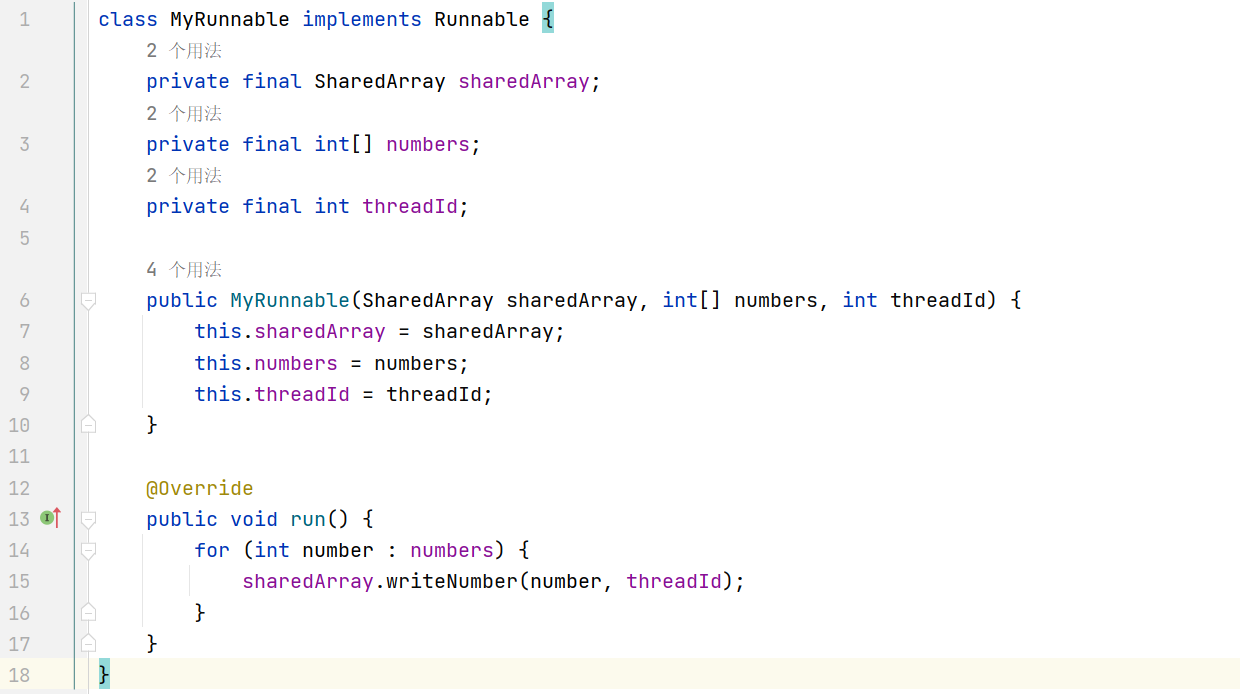
1. 主要代码：

SharedArray类：

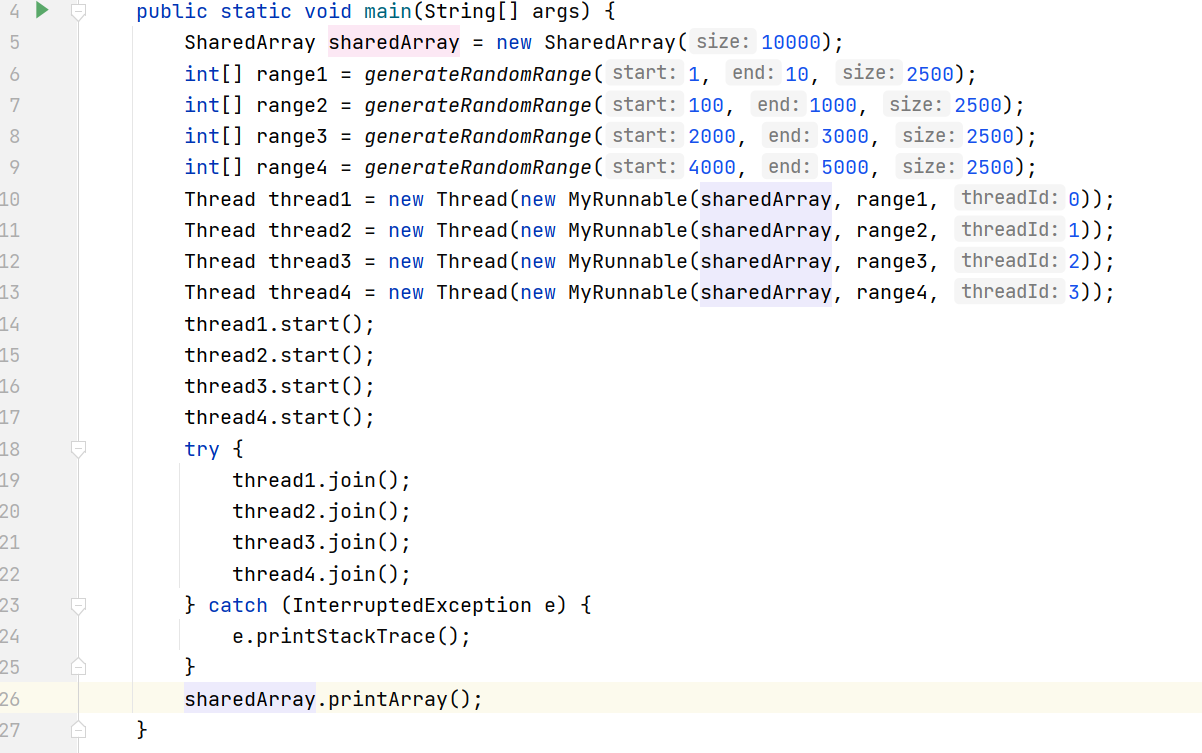




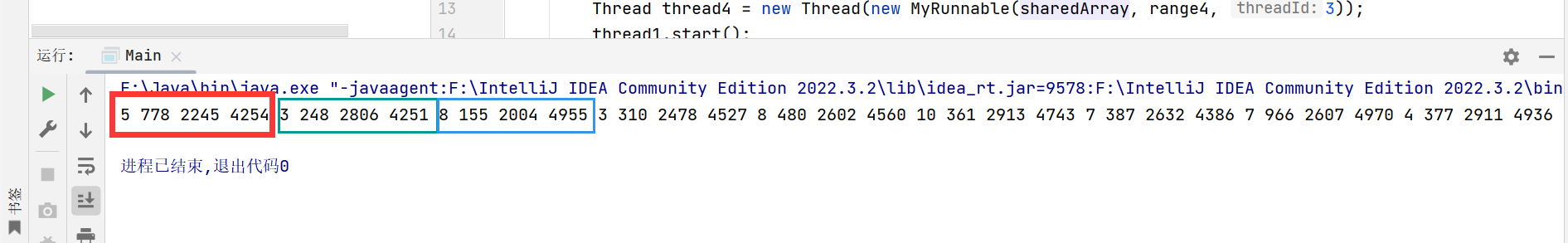
MyRunnable类：



Main类：



1. 运行结果，可以发现线程按顺序写入特定范围的数字：



1. 题目三：

用两种方法编程一个程序：一个线程向一个4位循环缓冲区中循环写入1-10的随机数，另一个线程将循环缓冲区中数据取出打印出来。

1. 实验环境

集成开发环境：IDEA Community Edition 2022.3.2

JDK版本：JDK17

1. 实现过程

两种方法分别为：使用wait()和notify()或者是使用java.util.concurrent包中的BlockingQueue。方法一使用wait()和notify()来实现线程间的通信和协调，确保一个线程在缓冲区满时等待，另一个线程在缓冲区空时等待。方法二使用BlockingQueue，BlockingQueue是一个线程安全的阻塞队列，支持在队列为空时阻塞读取操作，在队列满时阻塞写入操作。同时使用java.util.concurrent包中的ArrayBlockingQueue实现固定大小的阻塞队列，自动处理同步和线程间的协调。

总共设计了6个类，分别为：CircularBuffer类、Consumer类、ConsumerBQ类、Producer类、ProducerBQ类和Main类。具体说明如下：

* CircularBuffer类：

实现循环缓冲区，提供线程安全的put和get方法。

* int[] buffer：存储数据的数组。
* int count：当前缓冲区中的数据数量。
* int in：写入位置的索引。
* int out：读取位置的索引。
* synchronized void put(int value)：向缓冲区写入数据，若缓冲区已满，则等待。
* synchronized int get()：从缓冲区读取数据，若缓冲区为空，则等待。
* Consumer类：

消费者线程，从循环缓冲区读取数据并打印。

* CircularBuffer buffer：共享的循环缓冲区实例。
* void run()：从缓冲区读取数据并打印。
* ConsumerBQ类：

消费者线程，从BlockingQueue读取数据并打印。

* BlockingQueue<Integer> queue：共享的阻塞队列实例。
* void run()：从队列读取数据并打印。
* Producer类：

生产者线程，向循环缓冲区写入1到10的随机数。

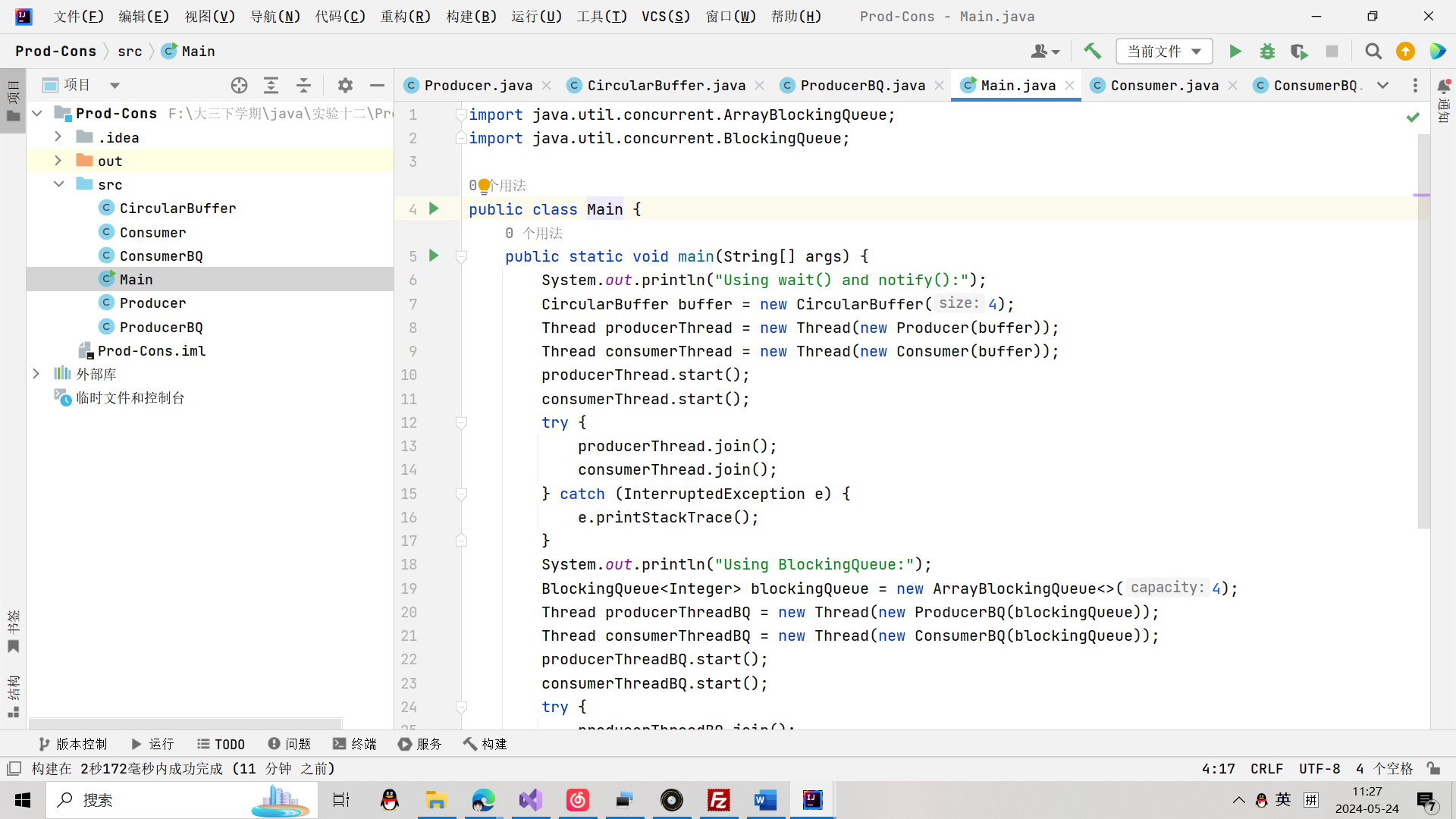
* CircularBuffer buffer：共享的循环缓冲区实例。
* Random random：用于生成随机数。
* void run()：生成随机数并调用buffer.put(value)将数据写入缓冲区。
* ProducerBQ类：

生产者线程，向BlockingQueue写入1到10的随机数。

* BlockingQueue<Integer> queue：共享的阻塞队列实例。
* Random random：用于生成随机数。
* void run()：生成随机数并调用queue.put(value)将数据写入队列。
* Main类：

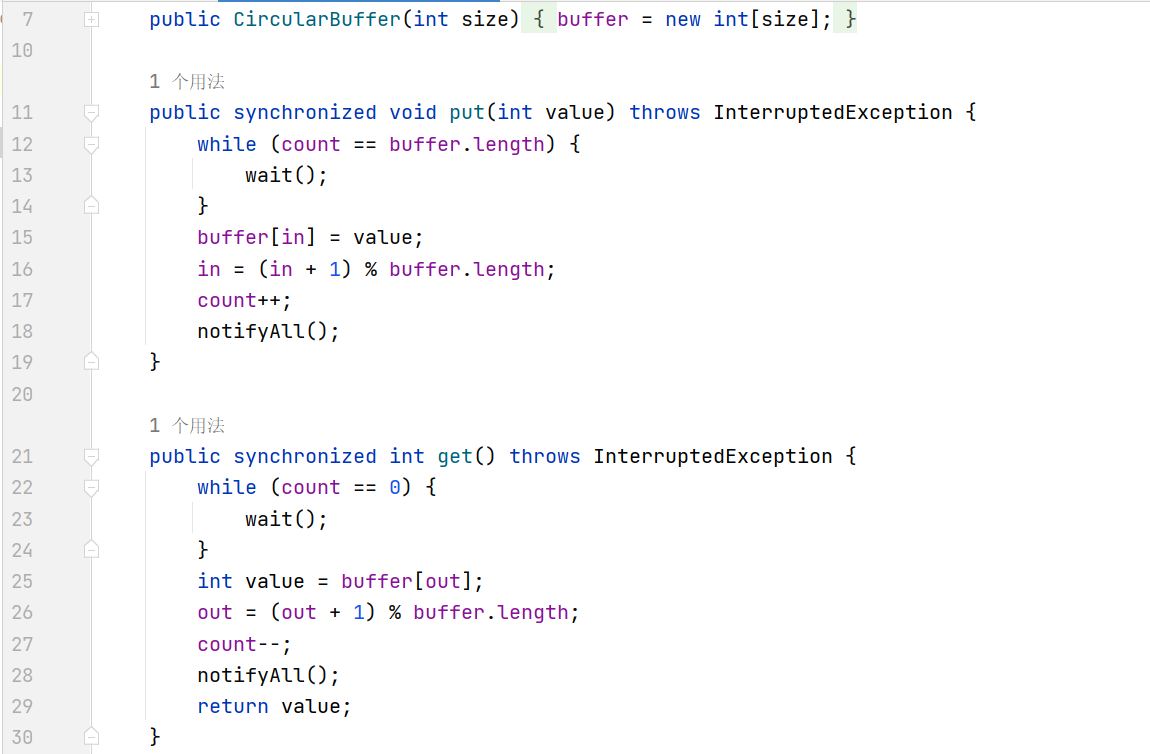
Main类中的main方法首先创建CircularBuffer实例。然后创建并启动Producer和Consumer线程。Producer线程向缓冲区写入数据，Consumer线程从缓冲区读取数据。完成后创建BlockingQueue实例。然后创建并启动ProducerBQ和ConsumerBQ线程。ProducerBQ线程向队列写入数据，ConsumerBQ线程从队列读取数据。

1. 过程截图
2. 全屏截图：

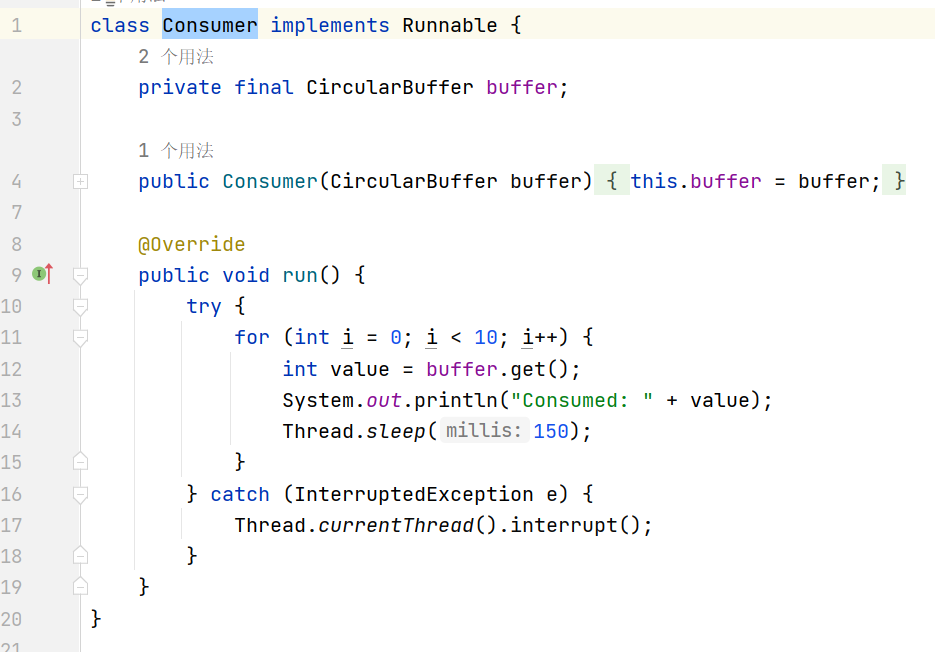


1. 主要代码：

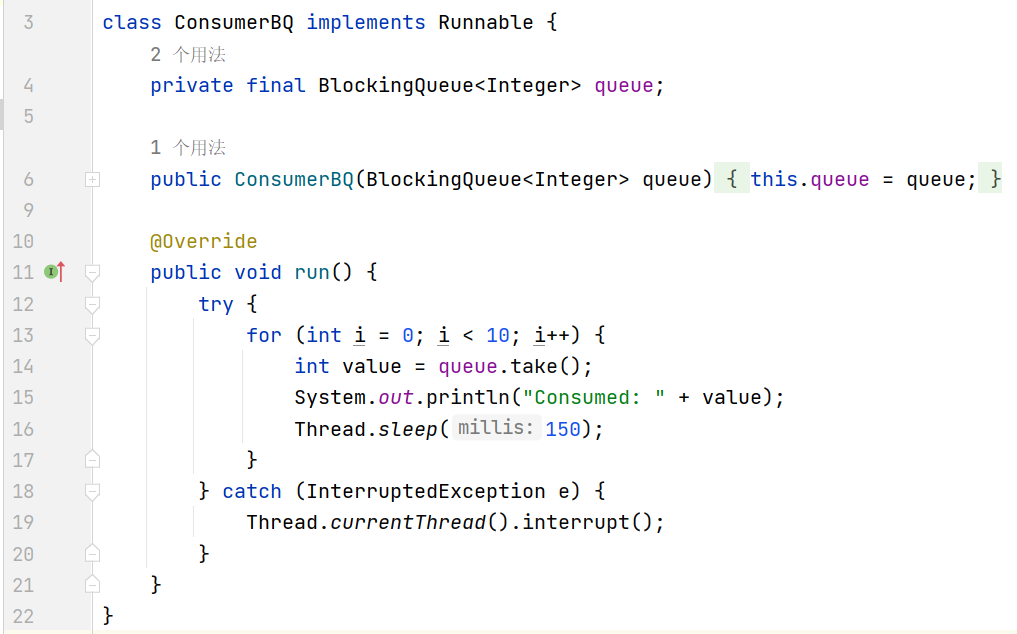
CircularBuffer类：



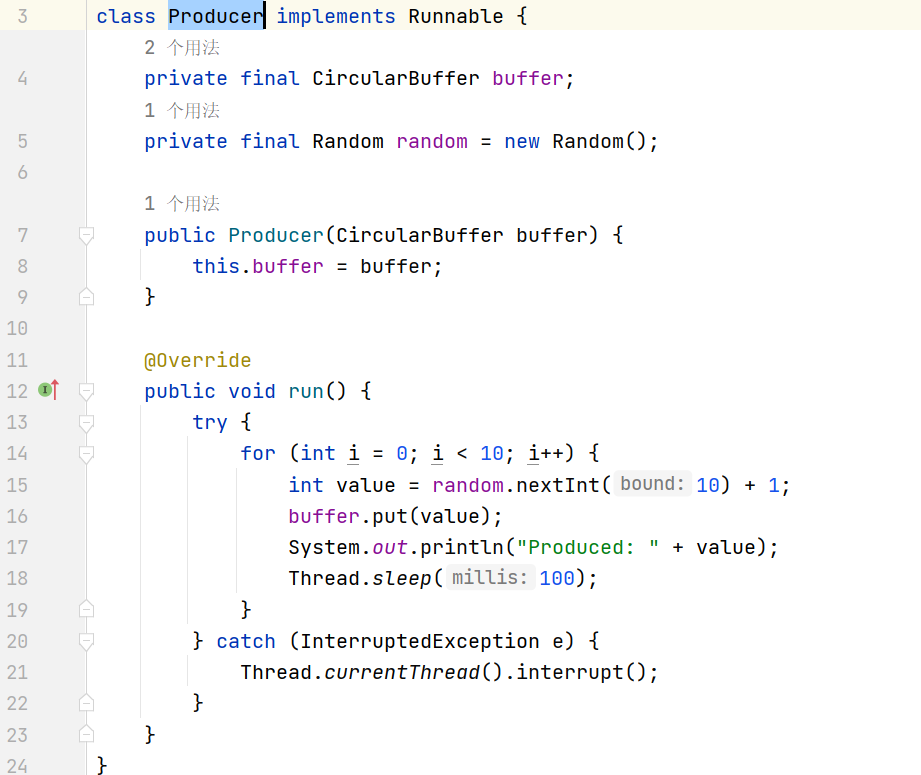
Consumer类：



ConsumerBQ类：



Producer类：



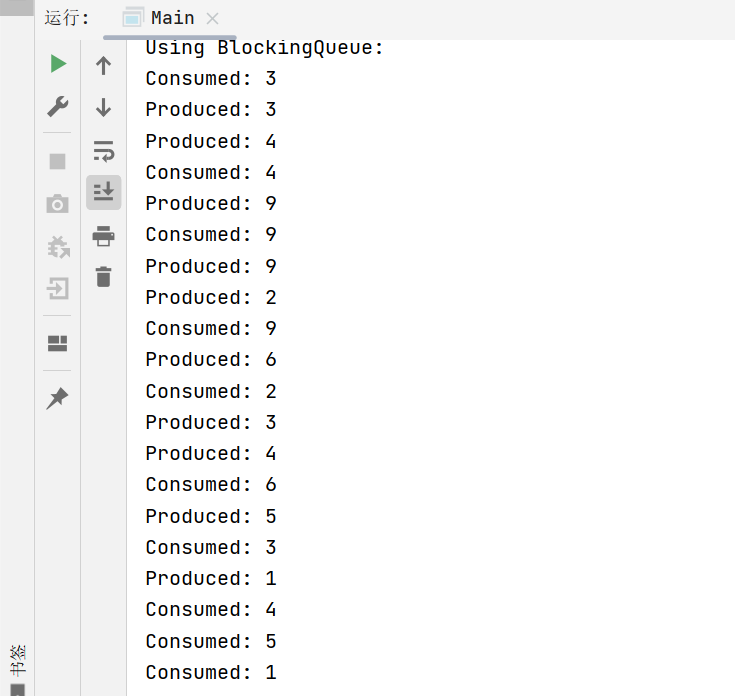
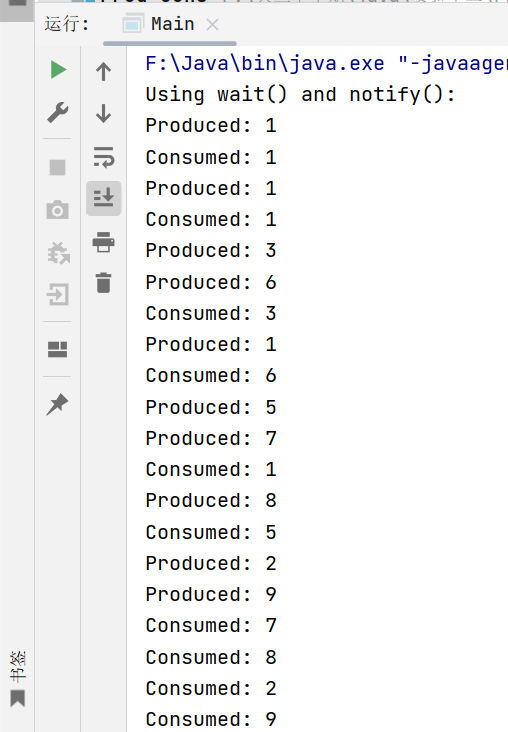
ProducerBQ类：



Main类：



1. 运行结果：



**三、实验总结与心得记录**

通过本次实验，我掌握了多线程编程的基本方法和技巧，包括使用Thread类、Runnable接口和线程池管理线程，并理解了线程同步和通信的重要性。