**华东师范大学数据科学与工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：计算机网络与编程 | **年级**：2021 | **上机实践成绩**： |
| **指导教师**：张召 | **姓名**：唐小卉 | **学号**：10215501437 |
| **上机实践名称**： lab8 | | **上机实践日期**：2023.4.21 |
| **上机实践编号**： | **组号**： | **上机实践时间**：2023.4.21 |

1. **实验目的**

* **对数据发送和接收进行优化**
* **实现信息共享**
* **熟悉阻塞I/O与非阻塞I/O**

1. **实验任务**

* **将数据发送与接收并行，实现全双工通信**
* **实现服务端向所有客户端广播消息**
* **了解非阻塞I/O**

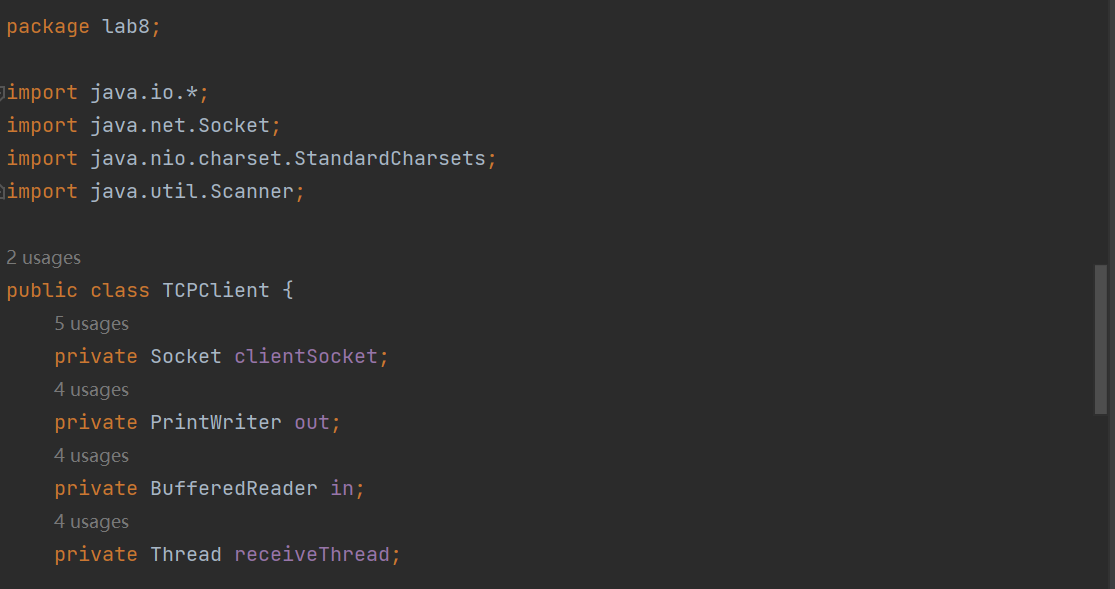
1. **使用环境**

* **IntelliJ IDEA**
* **JDK 版本: Java 19**

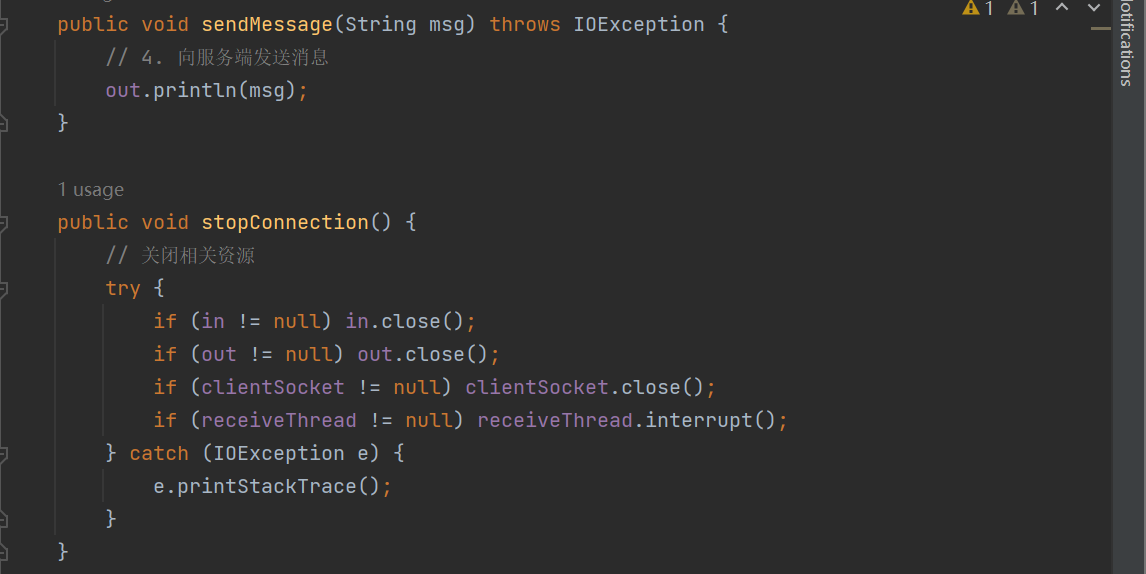
1. **实验过程**

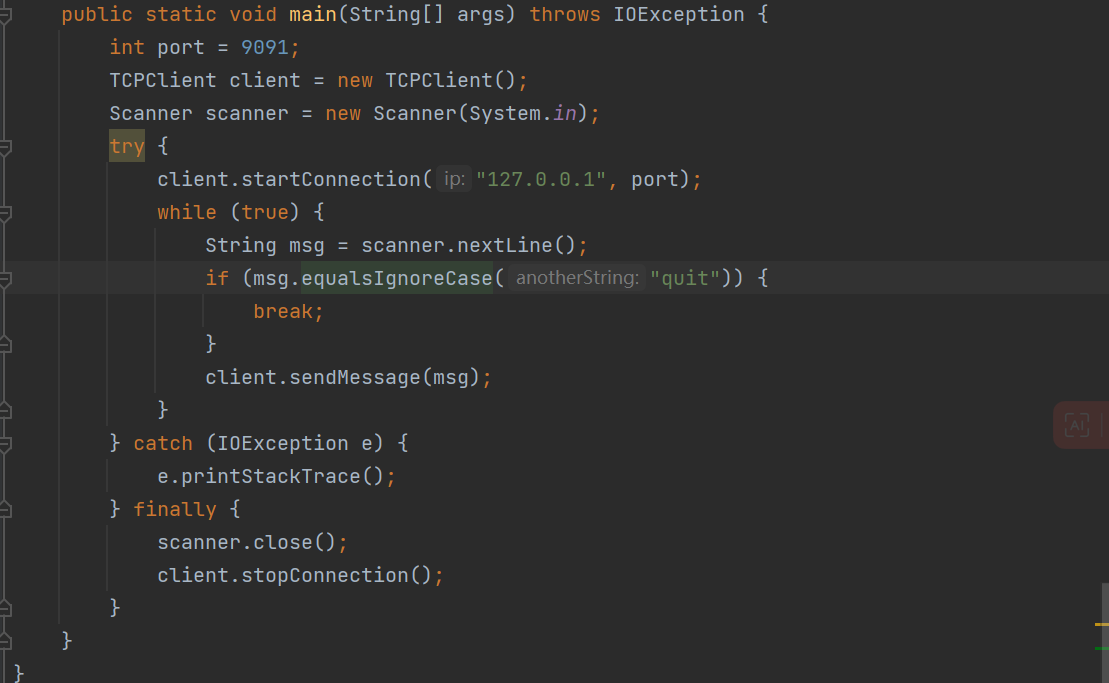
**Task1：**

**TCPClient代码：**

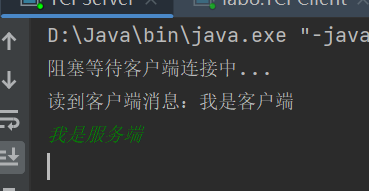


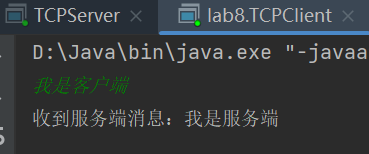






**运行结果：**



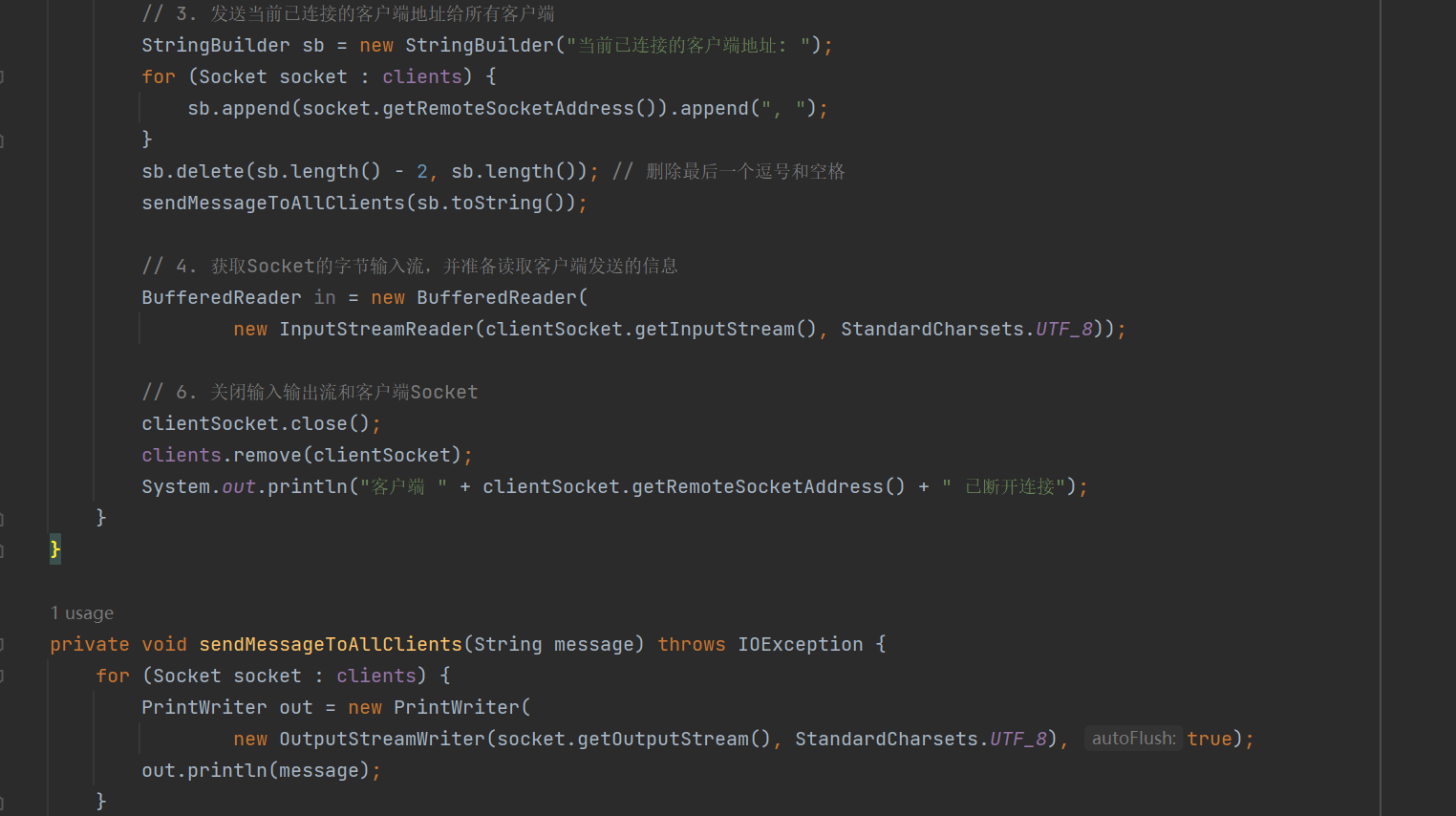


**Task2：**

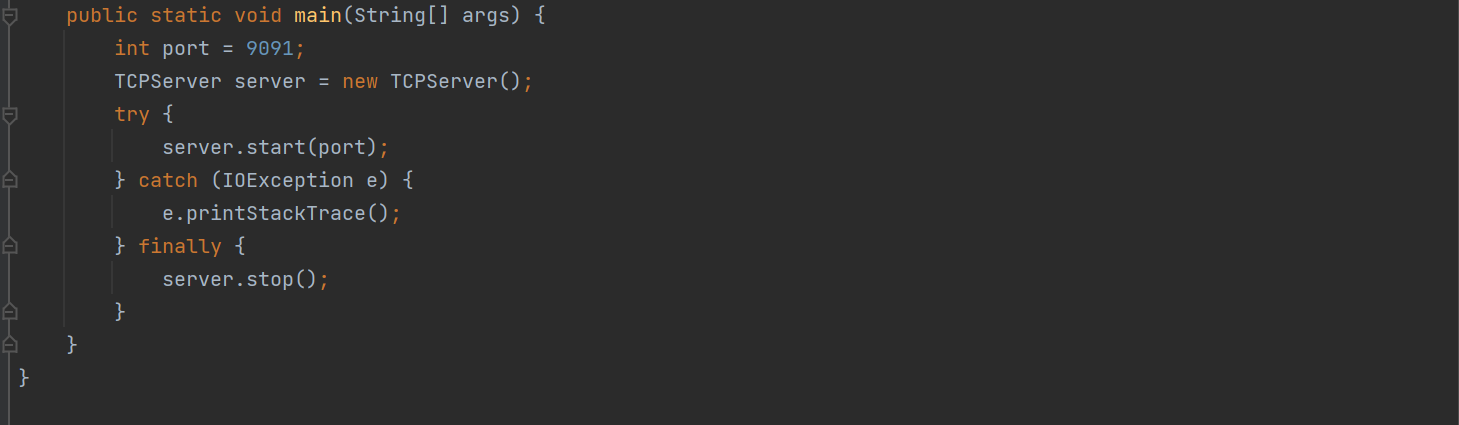
**修改后的代码：**

**TCPServer：**

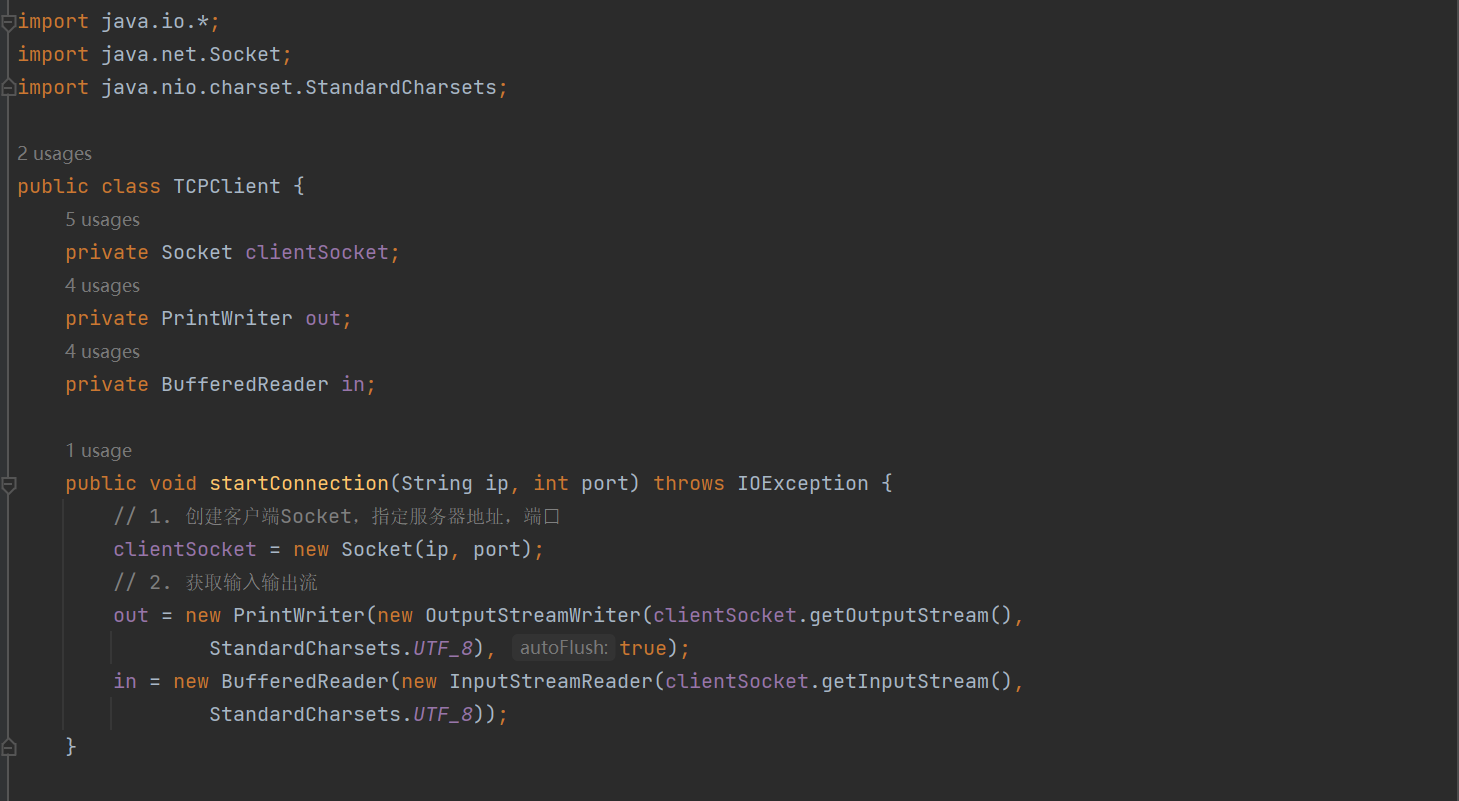


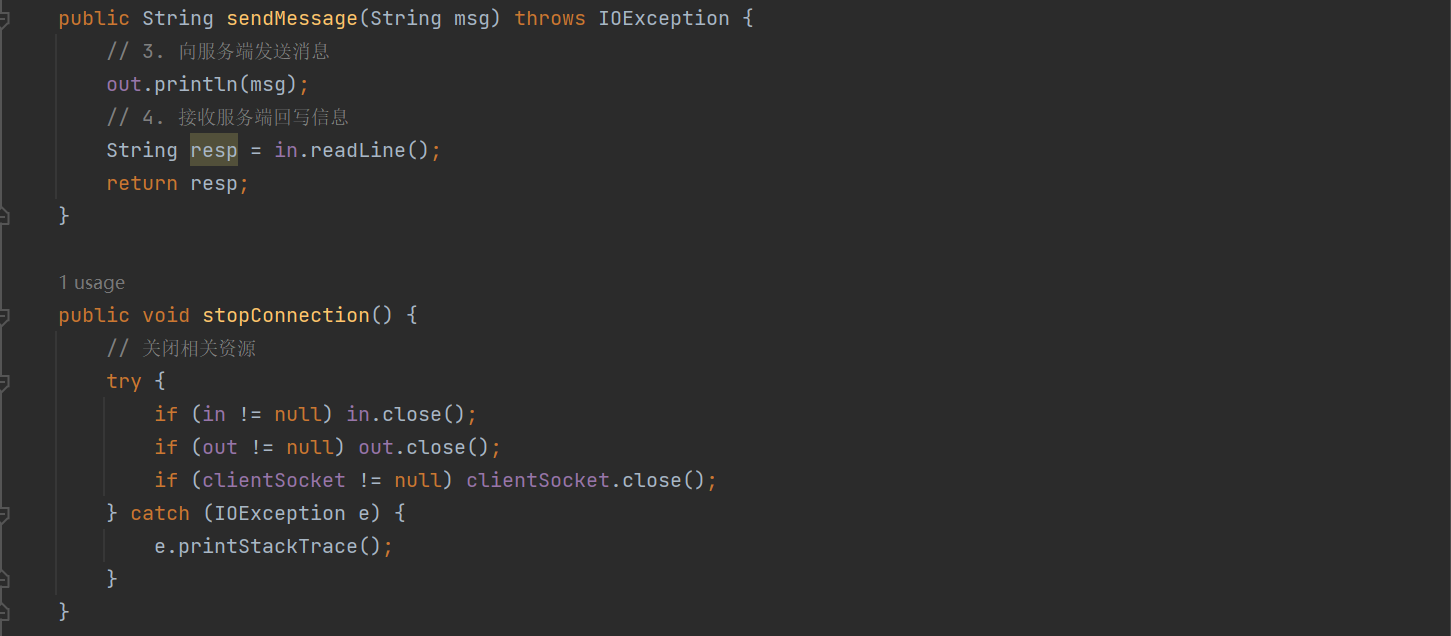


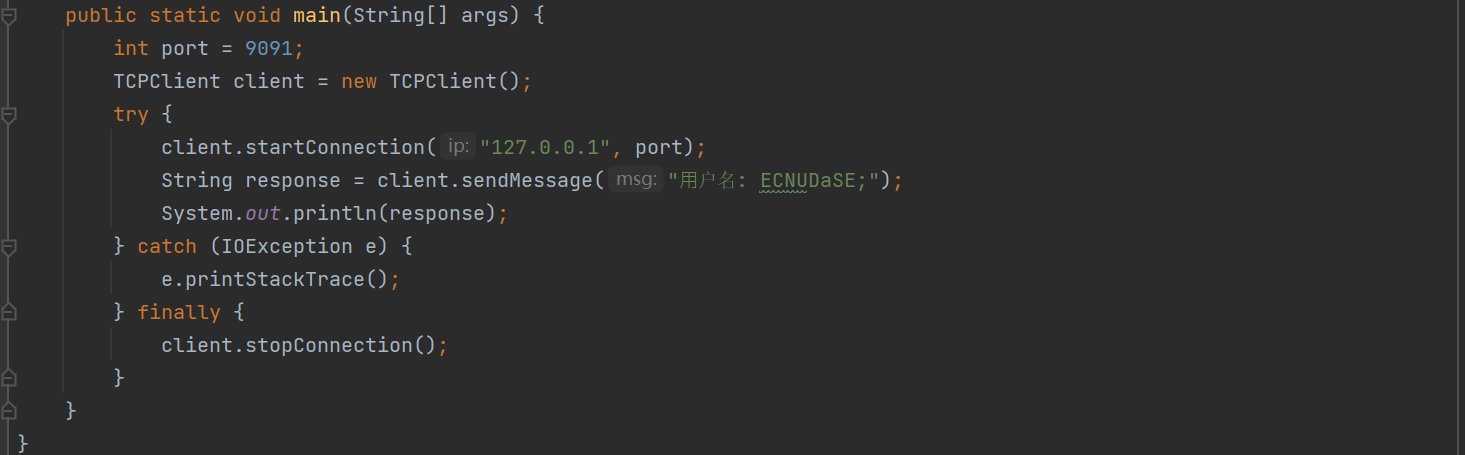




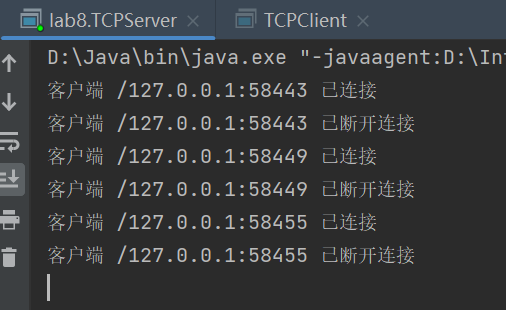
TCPClient：





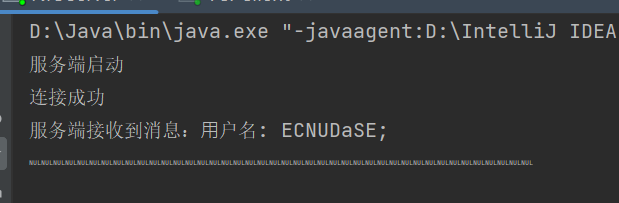


**运行结果：**



**Task3：**

**代码运行结果：**



**TCPServer和NIOServer的不同之处：**

TCPServer是基于传统的阻塞I/O模型，每个客户端连接会创建一个线程，阻塞在readLine()方法上等待读取数据，因此当有大量并发连接时，会创建大量线程，导致系统资源浪费和性能下降。

NIOServer是基于NIO的非阻塞I/O模型，采用了事件驱动机制，通过一个线程遍历多个通道，检查是否有数据可读或可写，从而避免了每个客户端连接创建一个线程的问题，可以处理更多的并发连接，提高系统的性能和可扩展性。

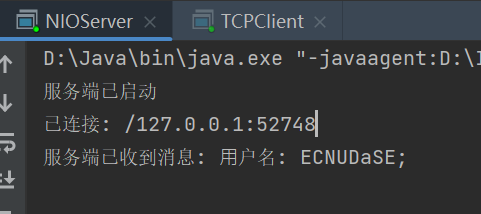
当有并发的1万个客户端(C10K)想要建立连接时，在TCPServer可能会存在以下问题：

1.创建大量线程会占用大量的系统资源，导致系统的负载增加，性能下降，甚至会出现线程资源耗尽的情况。

2.在同时处理大量并发连接时，每个连接的数据读写都会造成阻塞，如果有少量的客户端发送大量数据，可能会阻塞其他客户端的数据读写，导致响应时间变慢。

3.在处理大量并发连接时，如果有少量的客户端发送大量数据，可能会导致内存溢出，因为每个线程都需要分配一定的堆内存用于处理客户端的数据，如果客户端发送的数据太多，可能会导致内存不足，出现内存溢出异常。

**Task4:**



该代码中的I/O多路复用使用了Java NIO库中的Selector类和相关的SelectionKey类，底层实现依赖于操作系统提供的以下几个系统调用：

1.select() 或 poll()：在 Linux 上，这些调用会使用 epoll 实现，而在 macOS 上，它们使用 kqueue 实现。

2.accept()：用于接受传入的连接请求。

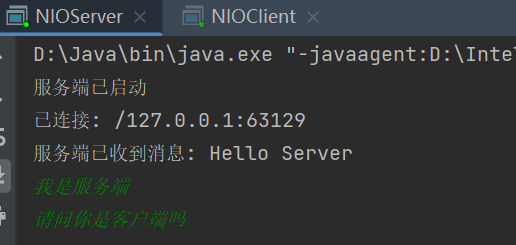
3.read()：用于读取从客户端发送来的数据。

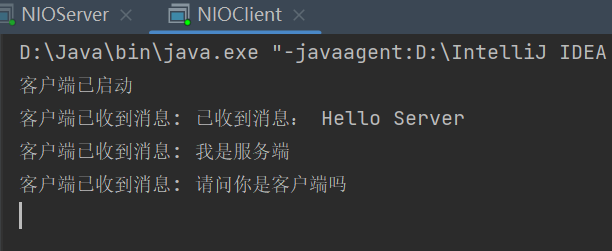
4.write()：用于将数据写回到客户端。

使用 Selector 类的主要原因是**它可以同时处理多个客户端连接**，并且可以在一个线程中完成对这些连接的处理，从而提高服务器的并发性能。在底层实现上，Selector 维护了一个 SelectionKey 集合，该集合包含了所有注册在 Selector 上的 Channel，以及对应的感兴趣的事件类型。Selector 在调用 select() 方法后会阻塞，直到至少有一个 Channel 准备就绪，然后返回已就绪的 Channel 的 SelectionKey 集合，开发者可以通过迭代这个集合，逐个处理每个已经准备就绪的 Channel。

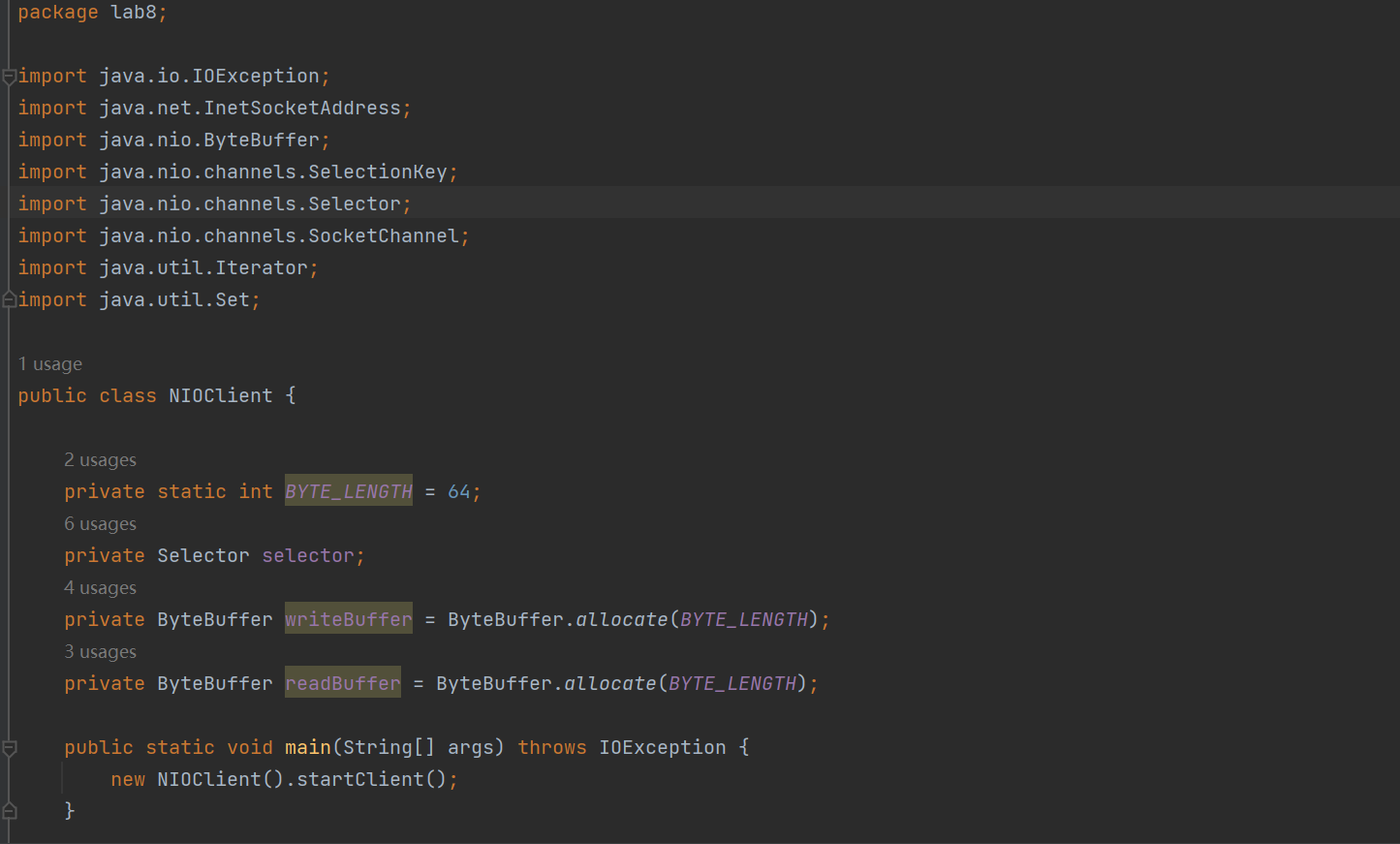
**Task5：**

**代码运行结果：**





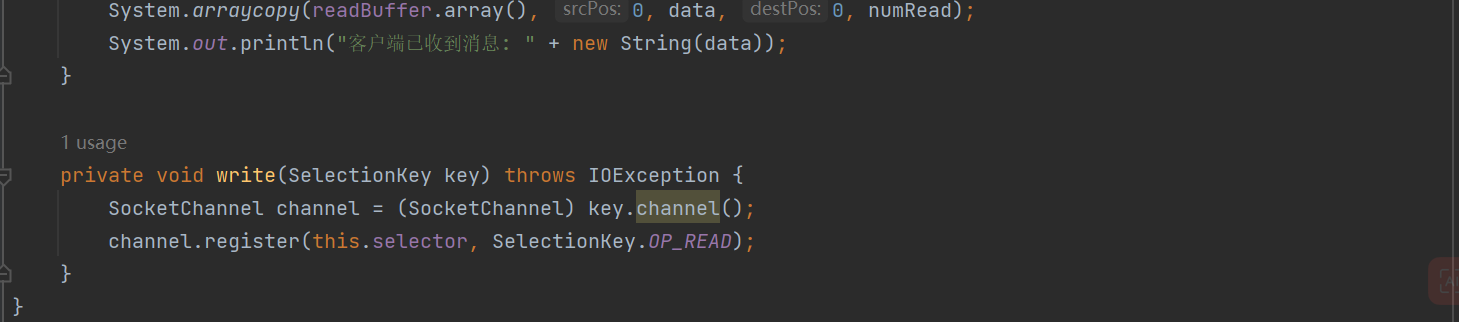
NIOClient：











1. **总结**

**这次的实验难度比较高，花了比较久的时间才能完全做好本次实验。这次实验之后，我对阻塞I/O和非阻塞I/O的了解更进了一步，但是我认为我的了解依然不够深刻，许多内容还是没有掌握，写代码的时候还是会遇到很多困难。之后还会继续努力研究和学习。**