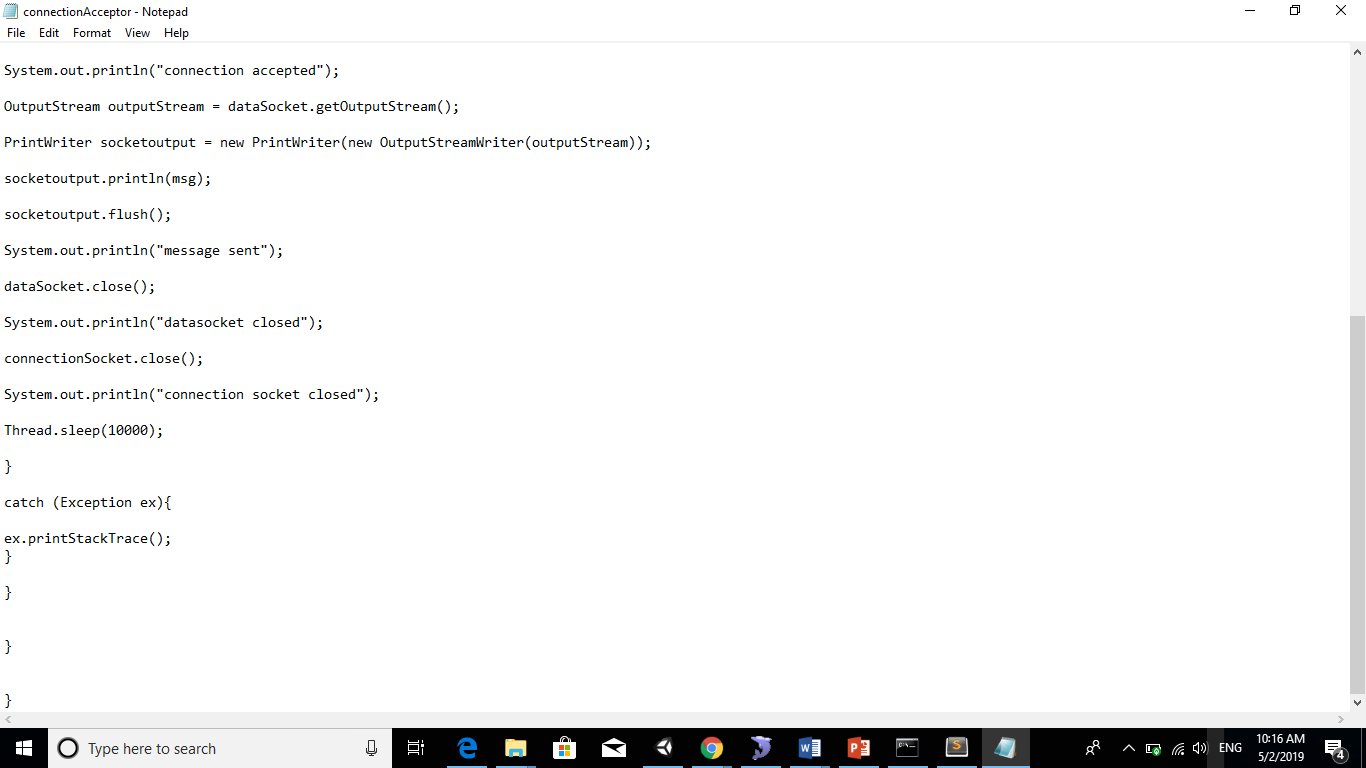
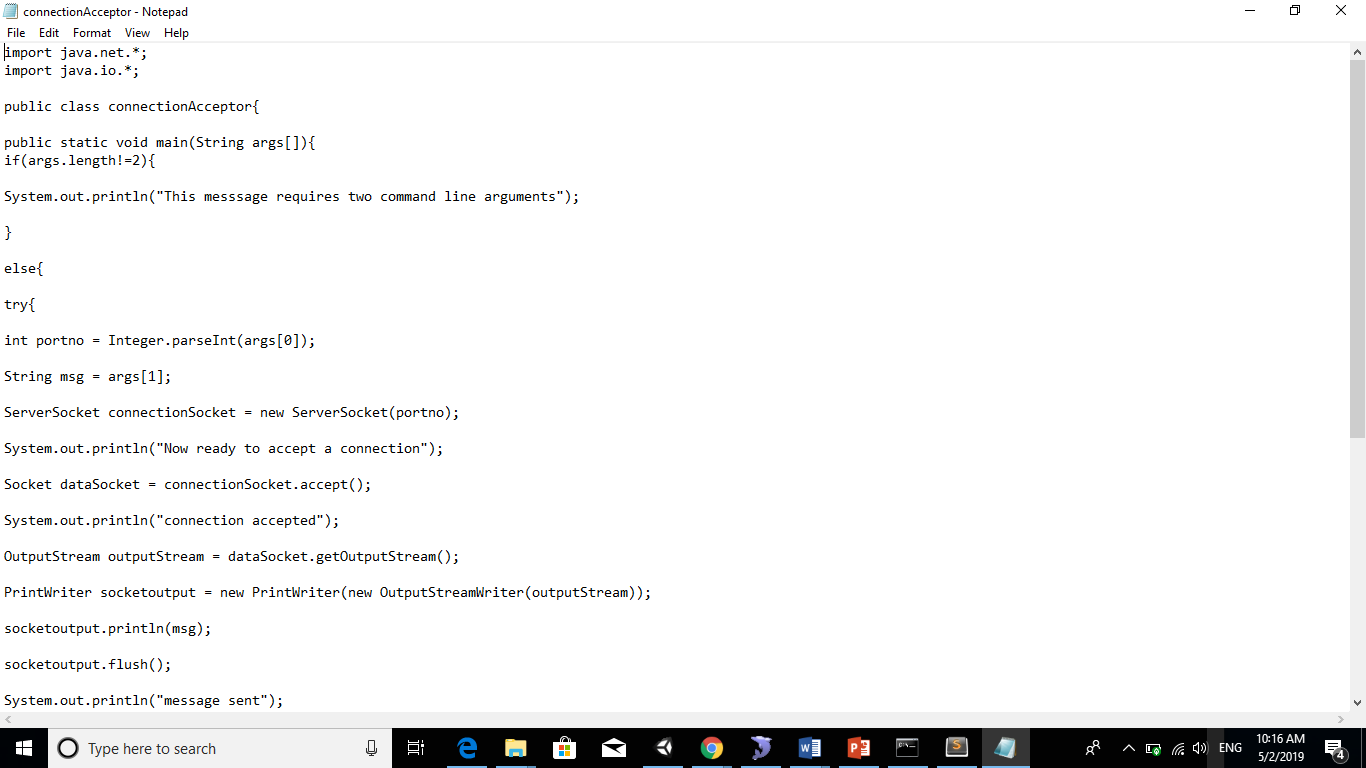
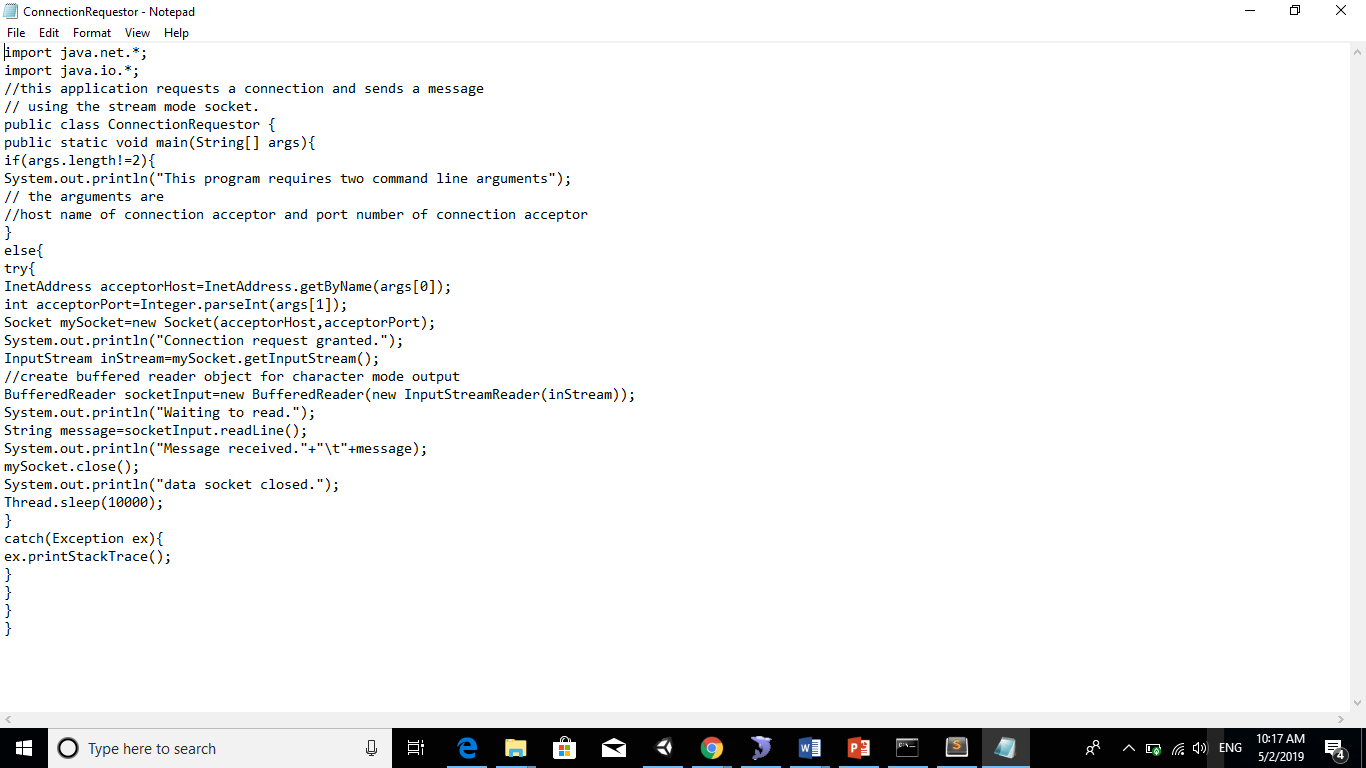
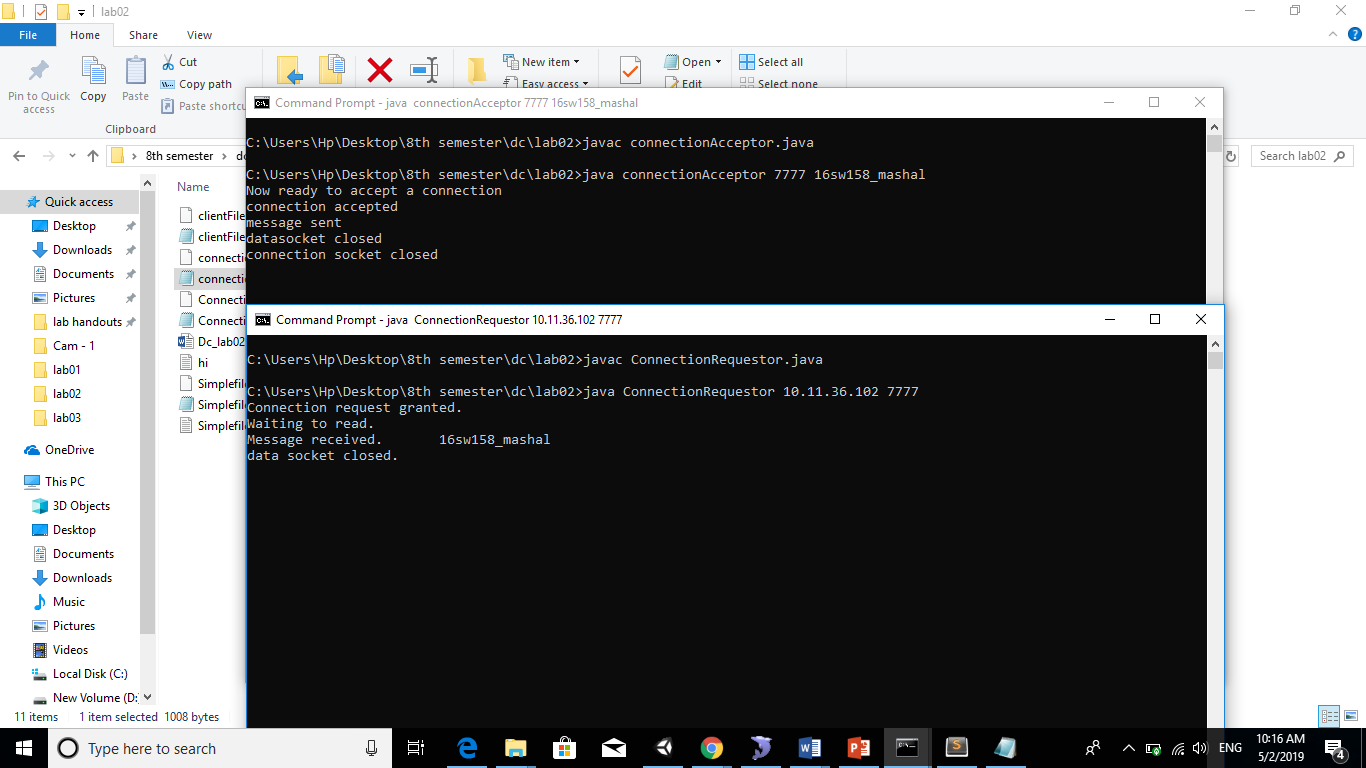
**LAB:02**

1. **Compile and run the above code. Start the acceptor first and then the requestor with appropriate command line arguments. Describe and explain the output.**

****

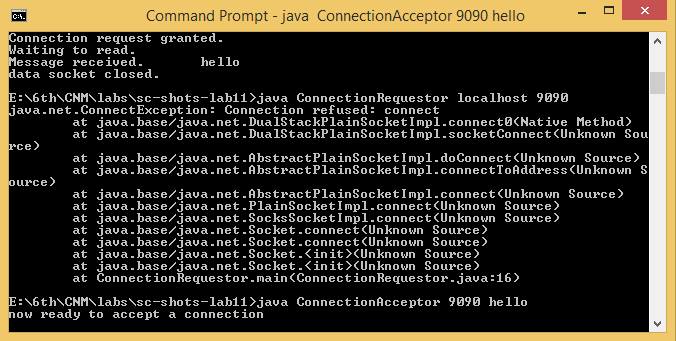
****

****

1. **Now run the code again, but reverse the order of program’s execution. Start the requestor first and then the acceptor. Describe and explain the outcome.**

**CODE:**

|  |  |
| --- | --- |
| **REQUESTOR:** | **ACCEPTOR:** |

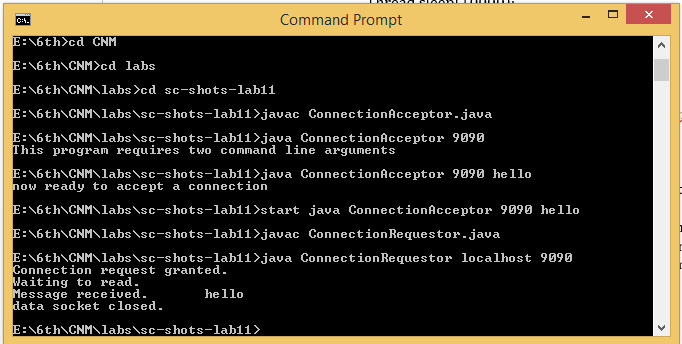
**OUTPUT:** 

1. **Add a time delay of 5 seconds in the ConnectionAcceptor process just before the message is written to the socket, then run the program. This will show you the blocking at the receiver. Show a trace of the output of the processes.**

**CODE:**

|  |  |
| --- | --- |
| **REQUESTOR:** | **ACCEPTOR:** |

**OUTPUT:**

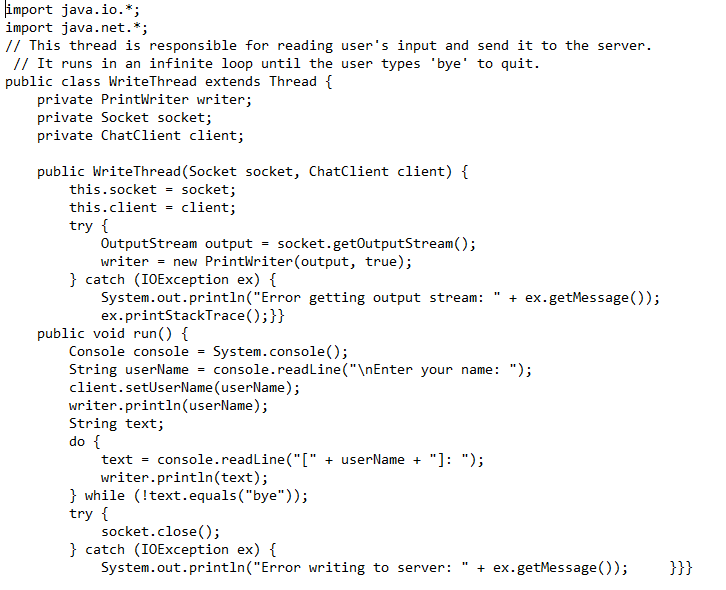


1. **Modify the sample code to include two way communication between the client and the server.**

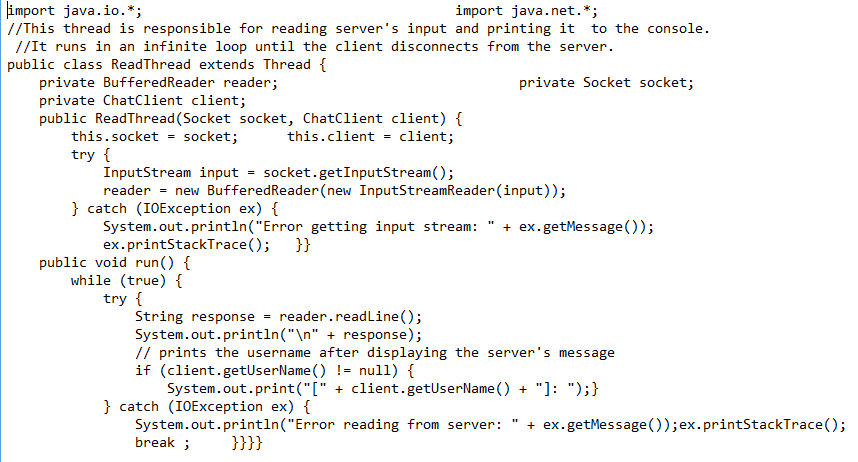
**CODE:**

|  |  |
| --- | --- |
| **SERVER:** | **CLIENT:** |

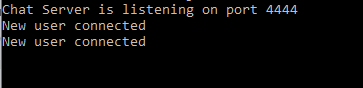
**THREADS:**





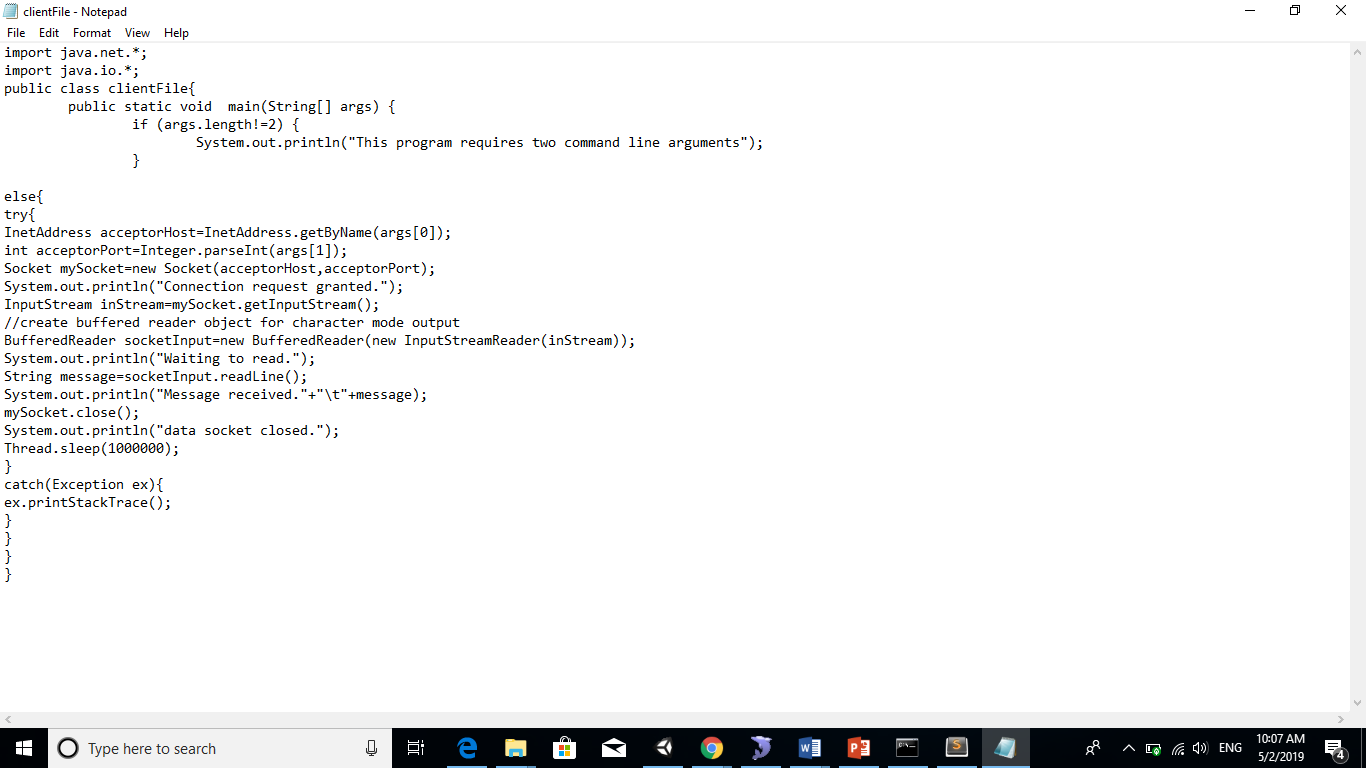


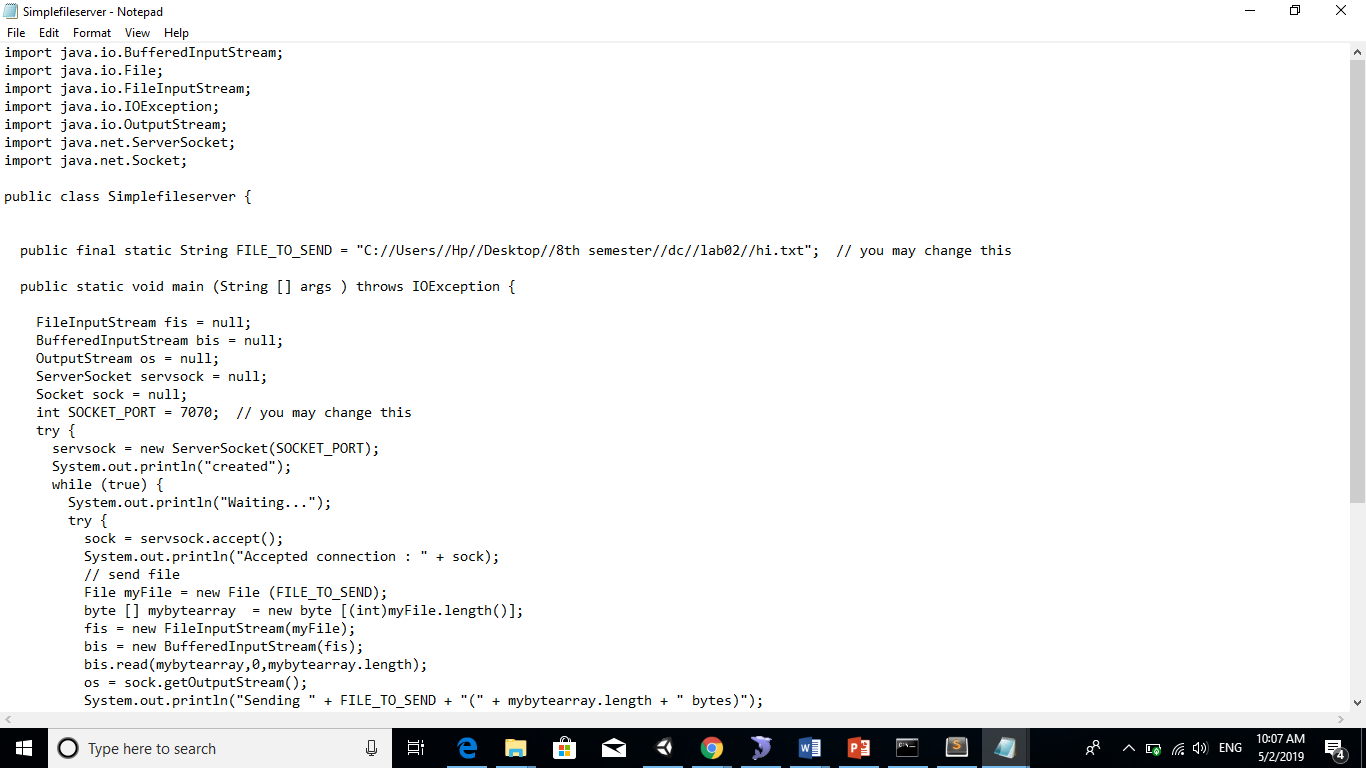
**OUTPUT:**

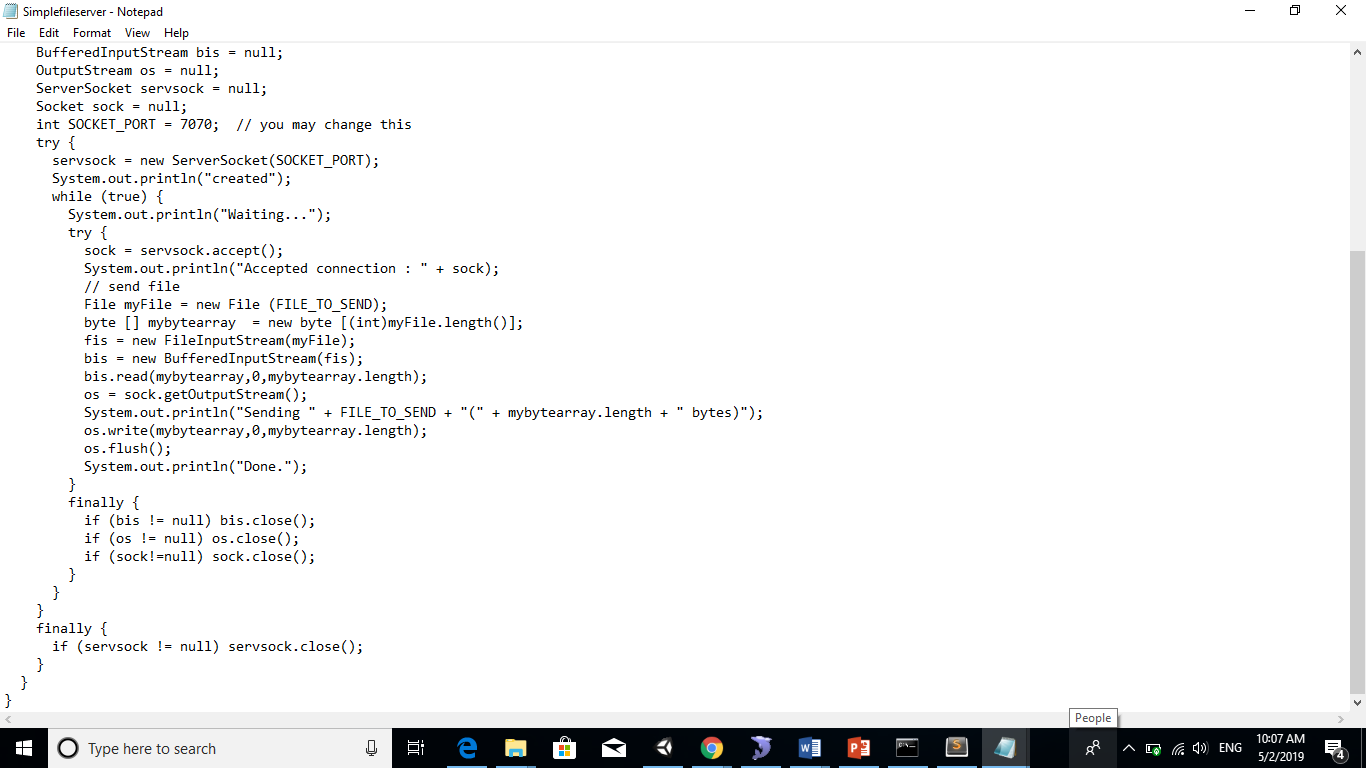
****

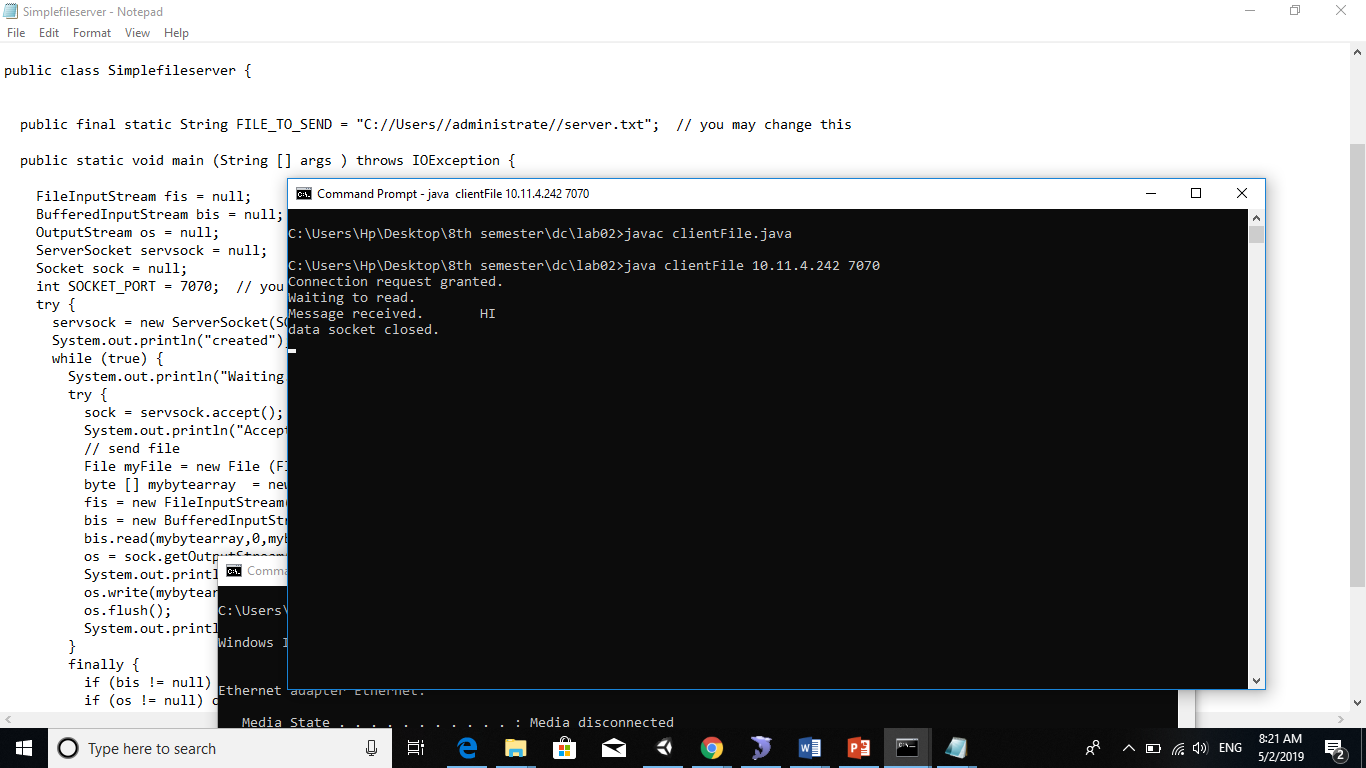
****

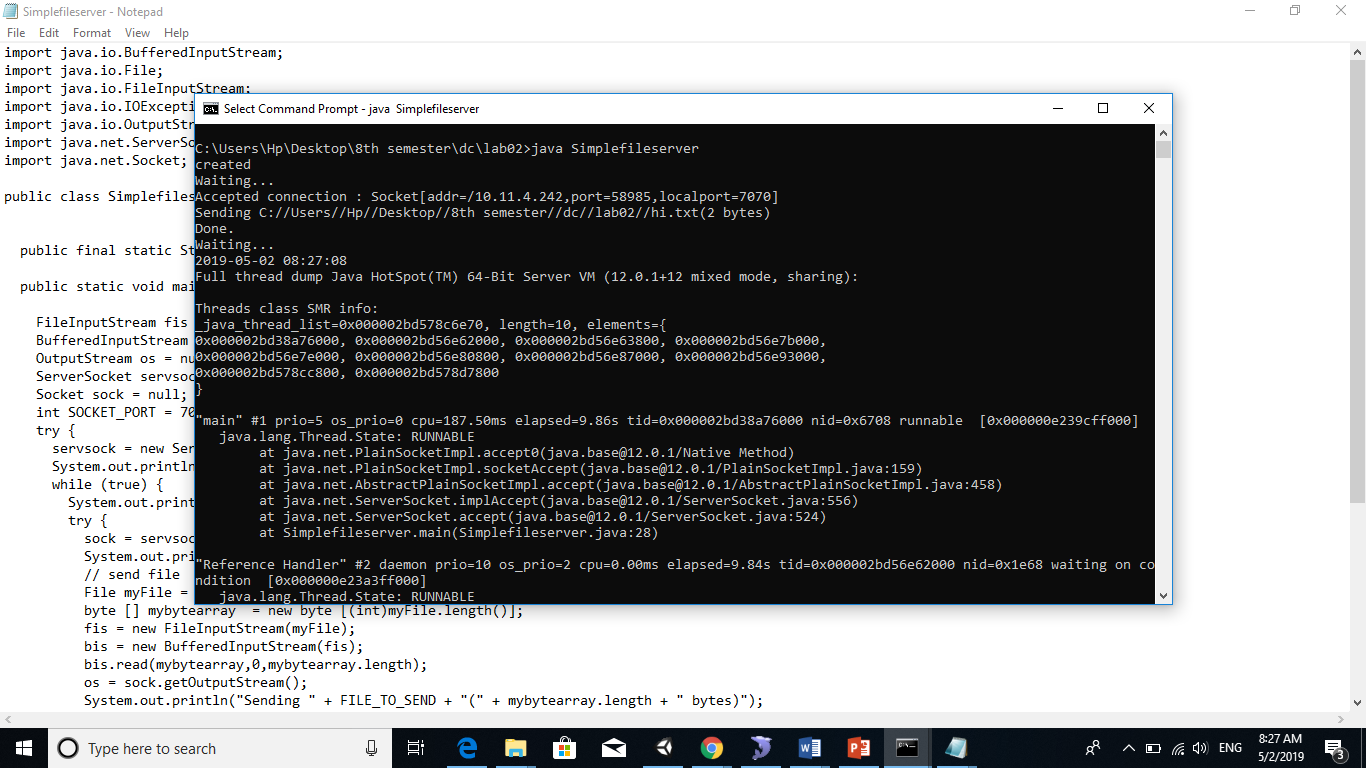
1. **Modify the sample code to send complete files between the client to the server.**







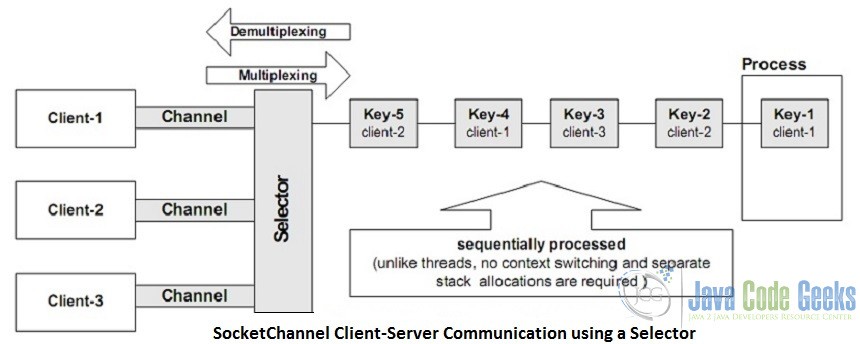


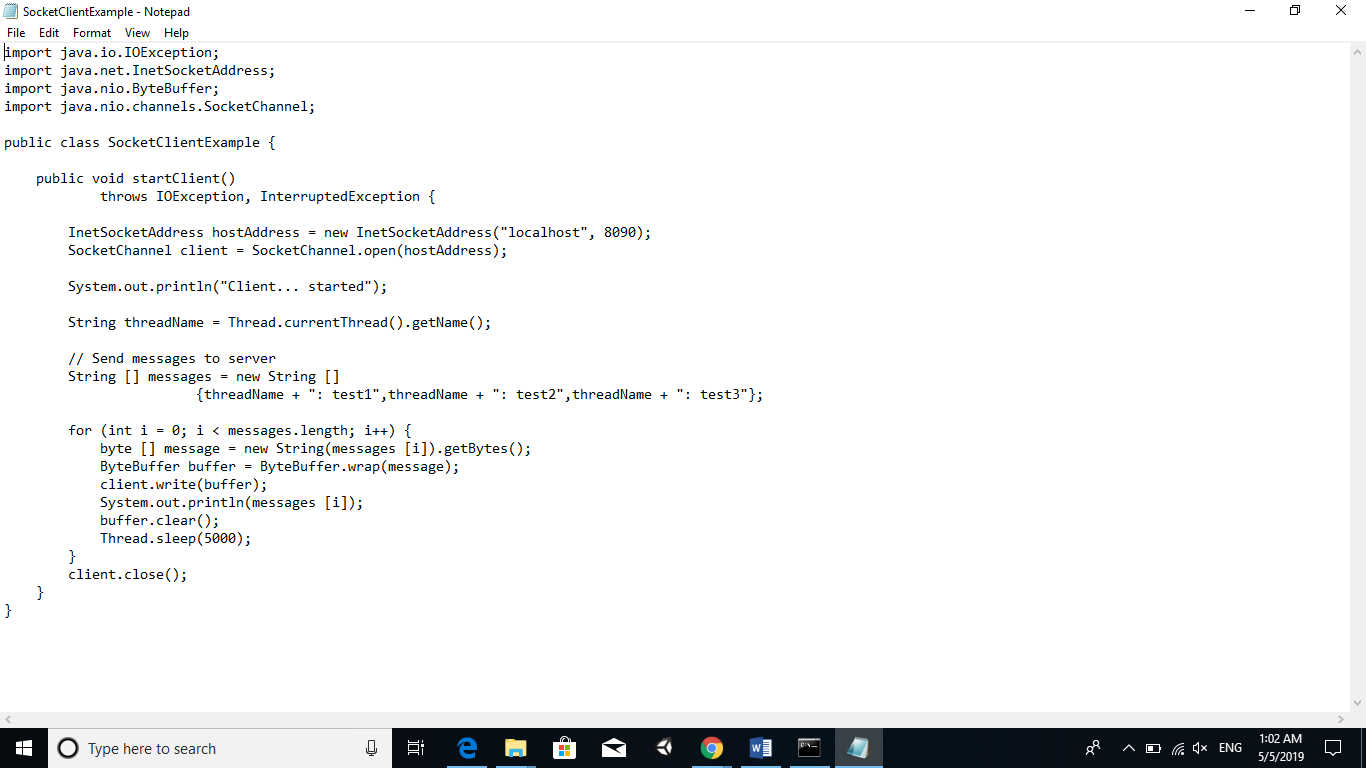


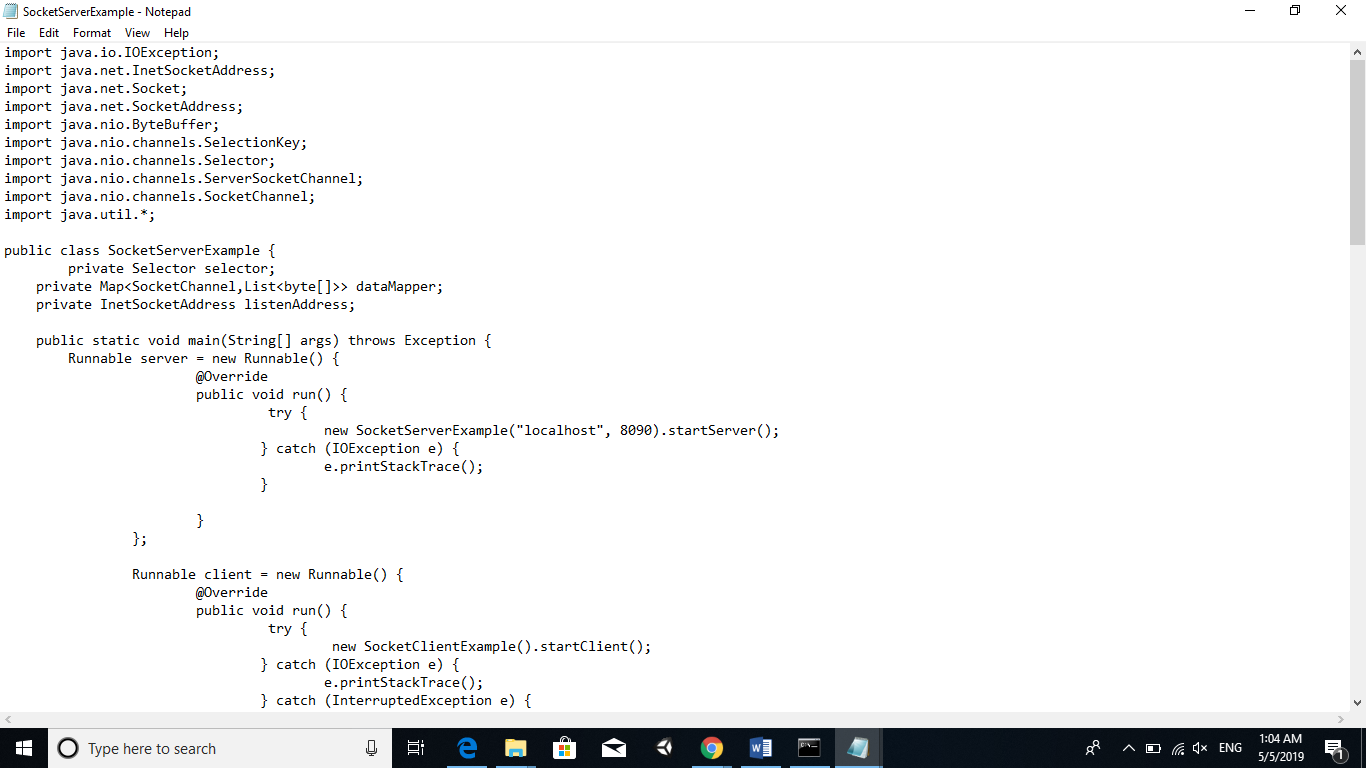
1. **Explore the non-blocking java socket API in the nio package and implement a sample program.**

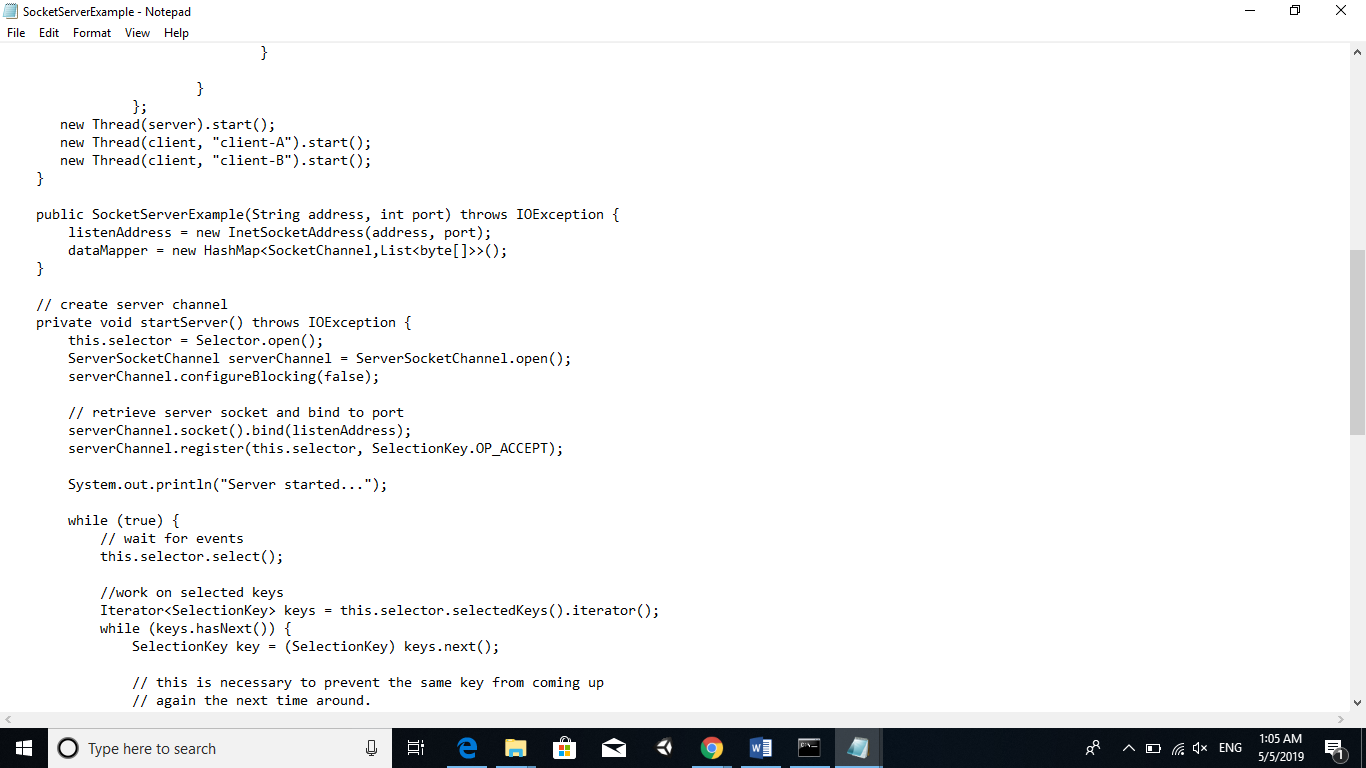
With the standard java sockets, if the server needed to be scalable, the socket had to be passed to another thread for processing so that the server could continue listening for additional connections, meaning call the ServerSocket’s accept()method again to listen for another connection.

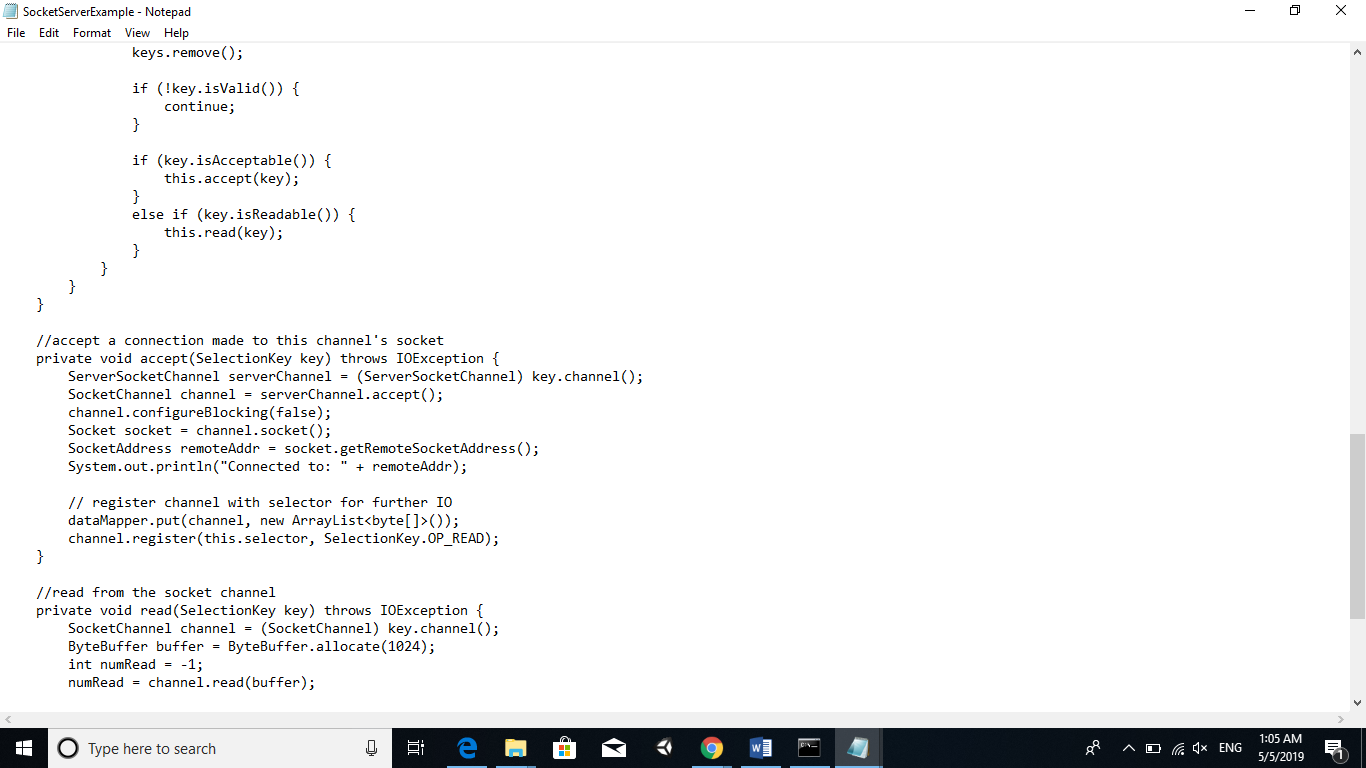
A SocketChannel on the other hand is a non-blocking way to read from sockets, so that you can have one thread communicate with multiple open connections at once. With socket channel we describe the communication channel between client and server. It is identified by the server IP address and the port number. Data passes through the socket channel by buffer items. A selector monitors the recorded socket channels and serializes the requests, which the server has to satisfy. The Keys describe the objects used by the selector to sort the requests. Each key represents a single client sub-request and contains information to identify the client and the type of the request. With non-blocking I/O, someone can program networked applications to handle multiple simultaneous connections without having to manage multiple thread collection, while also taking advantage of the new server scalability that is built in to java.nio. The below image illustrates this procedure:

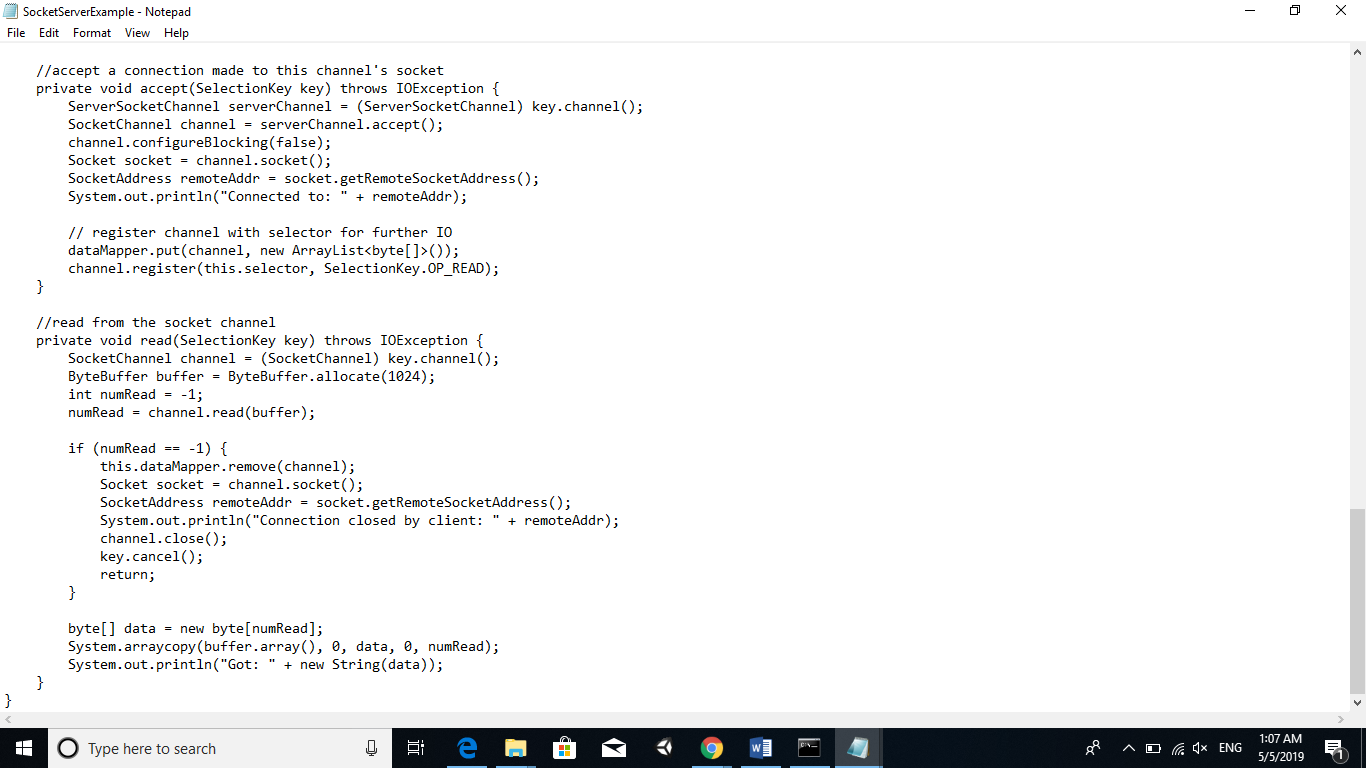












**Output:**

