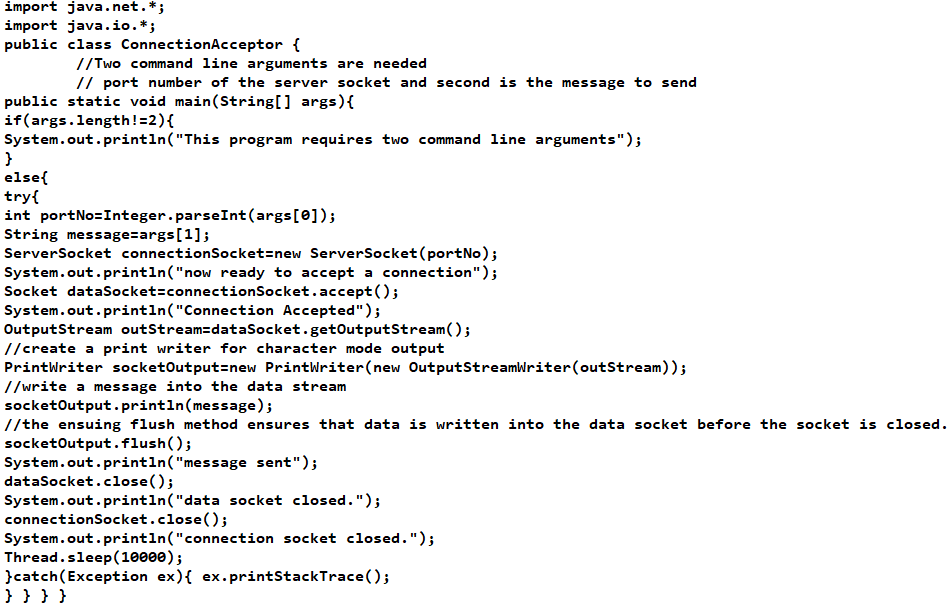
**Lab-02**

**Objective: To learn creation of stream sockets using Socket Programming**

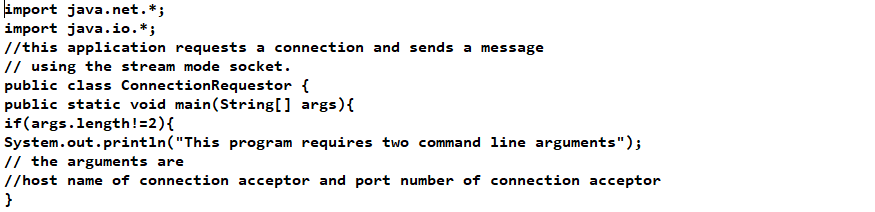
1. **Compile and run the above code, but reverse the order of program’s execution. Start the requestor first and then the acceptor. Describe and explain the outcome.**

**Code:**

**ConnectionAcceptor.java**

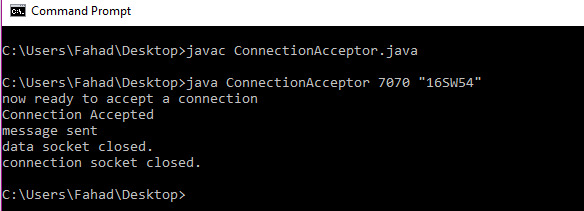


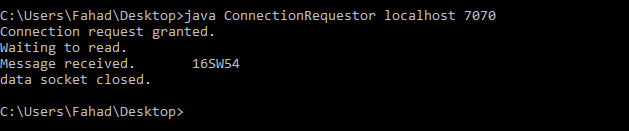
**ConnectionRequestor.java**





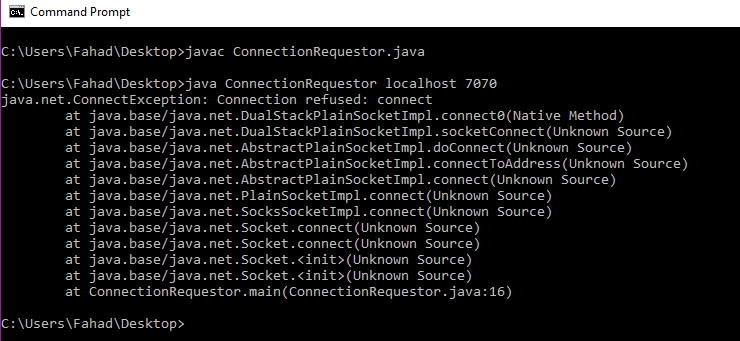
**Output:**





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.**

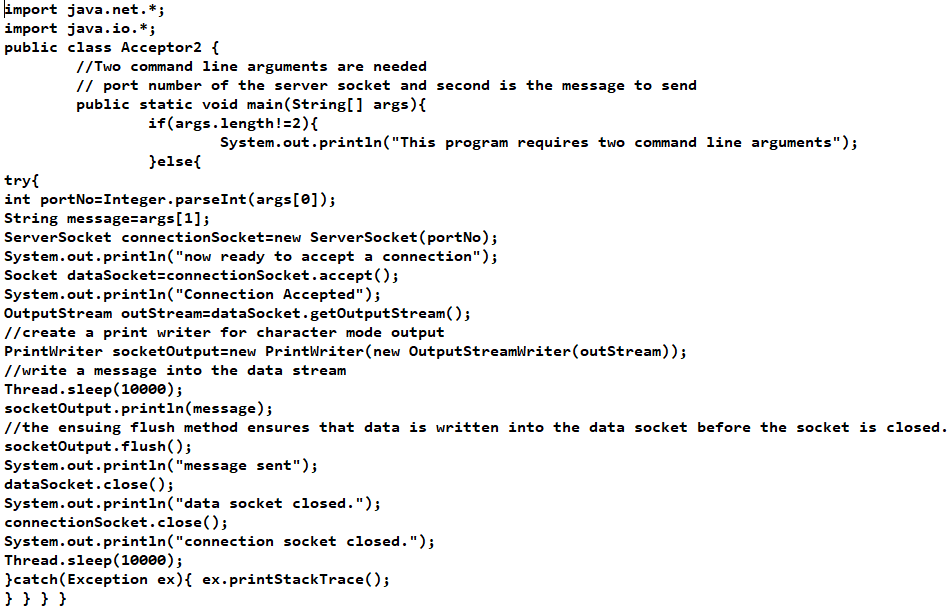
**Output:**



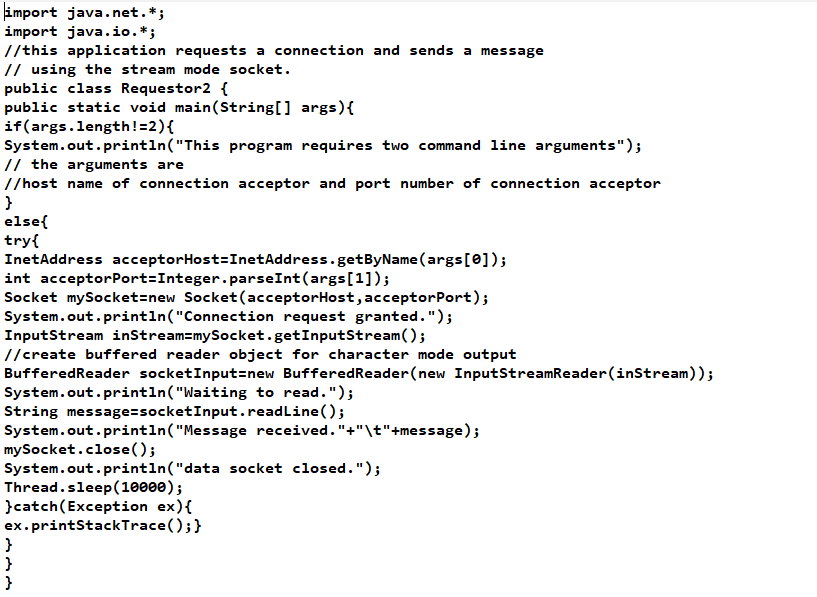
1. **Add a time delay of 10 seconds in the ConnectionAcceptor process just before the message is written to the socket, then run the program. This will show you the blocking at the receiver. Show a trace of the output of the processes.**

**Code:**

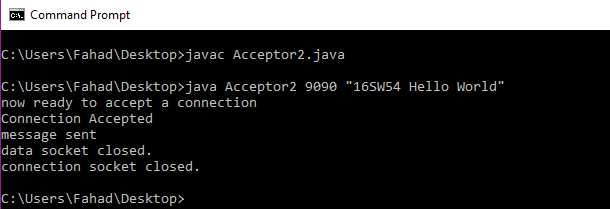
**Acceptor2.java**

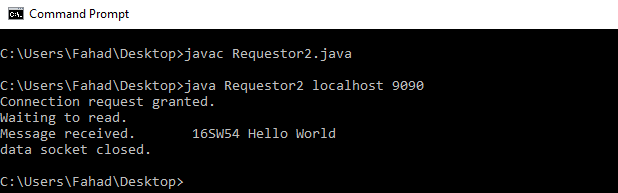


**Requestor2.java**



**Output:**

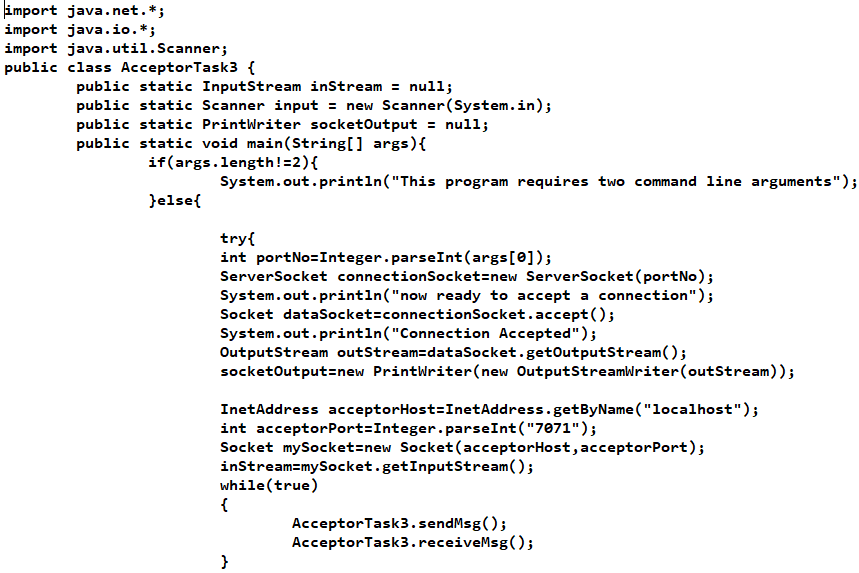


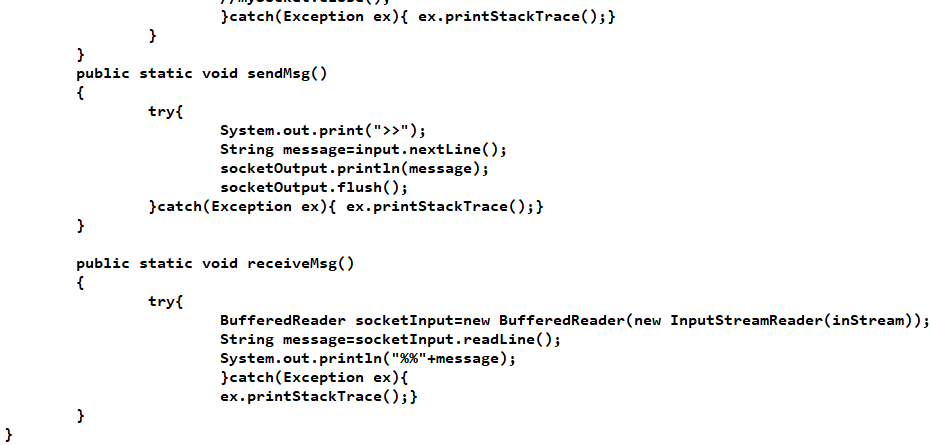


1. **Modify the sample code to include the two way communication between the client and the server.**

**Code:**

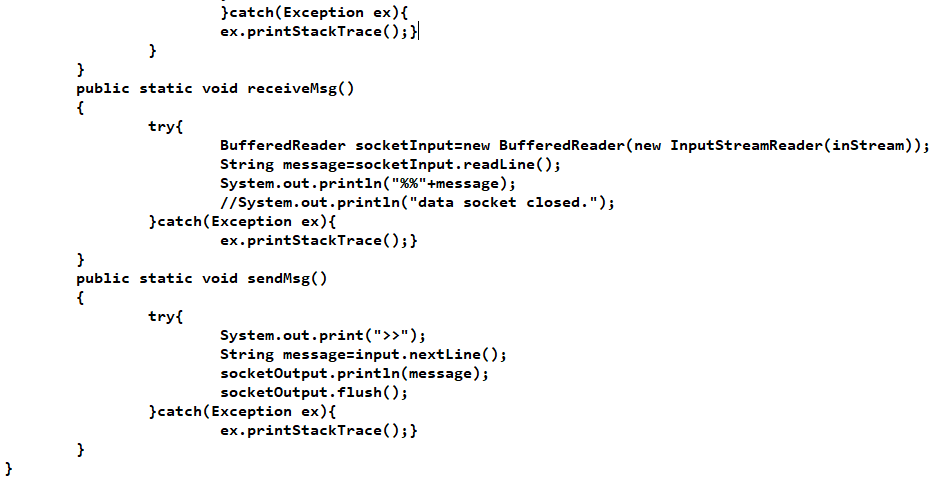
**AcceptorTask3.java**



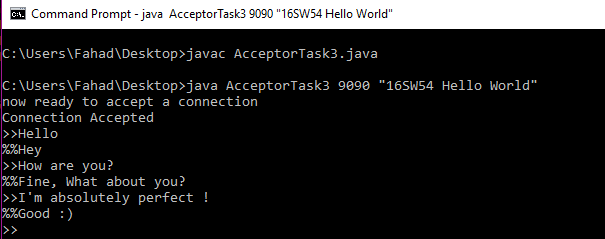


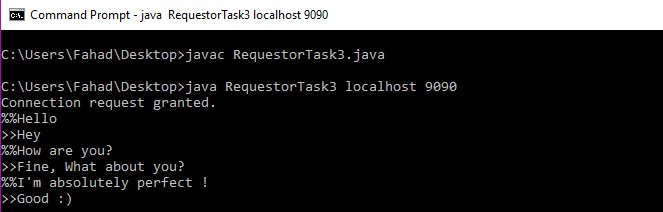
**RequestorTask3.java**





**Output:**

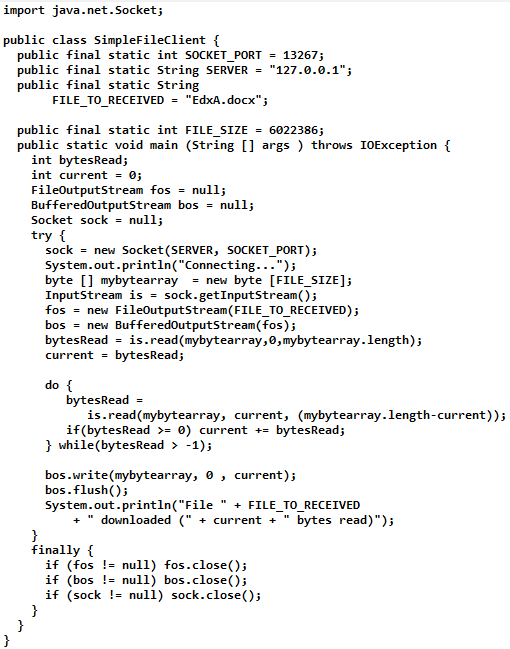




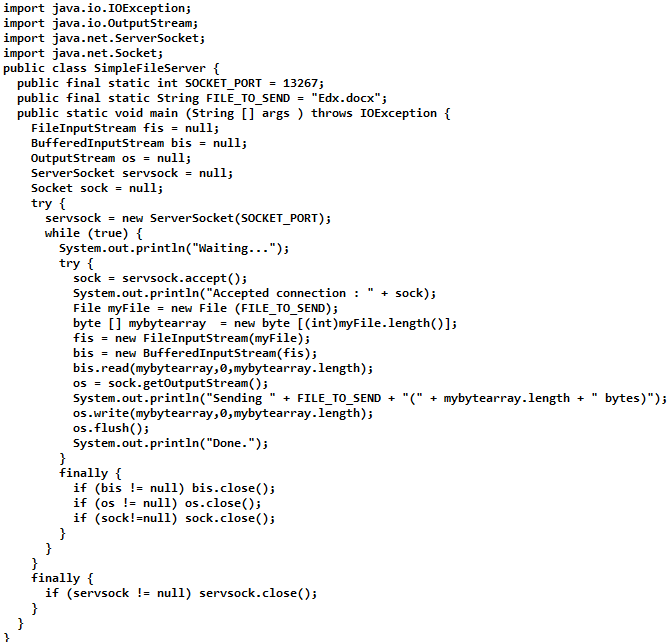
1. **Modify the sample code to send complete files between the client to the server.**

**Code:**

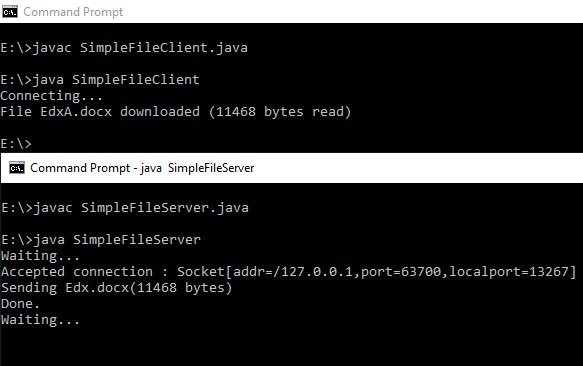
**SimpleFileClient.java**



**SimpleFileServer.java**



**Output:**



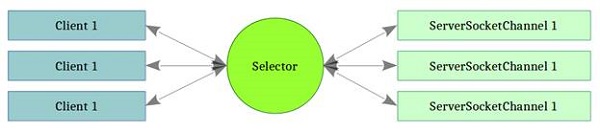
1. **Explore the non-blocking java socket API in the nio package and implement a sample program.**

Starting [JDK 1.4](https://crunchify.com/where-is-java-installed-on-my-mac-osx-system/), NIO was created to allow all [Java programmers](https://crunchify.com/category/java-tutorials/) to implement very high-speed input/output without having to deal with custom native code. NIO uses java.nio.buffer library compare to [simple I/O](https://crunchify.com/java-file-copy-example-simple-way-to-copy-file-in-java/) which drains and fills back buffer internally any operating system.

In this tutorial we will go over java.nio.channels and java.nio.channels.Selectorlibraries.

* channels represent connections to entities that are capable of performing I/O operations, such as files and sockets; defines selectors, for multiplexed, non-blocking I/O operations.
* selector may be created by invoking the open method of this class, which will use the system’s default selector provider to create a new selector.

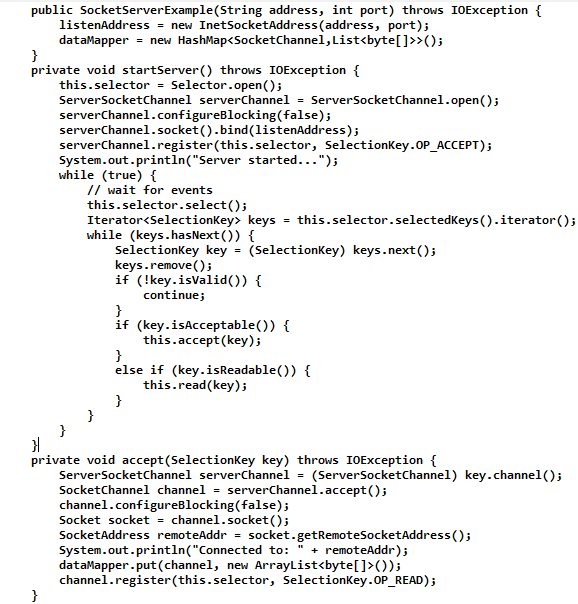
How it works:

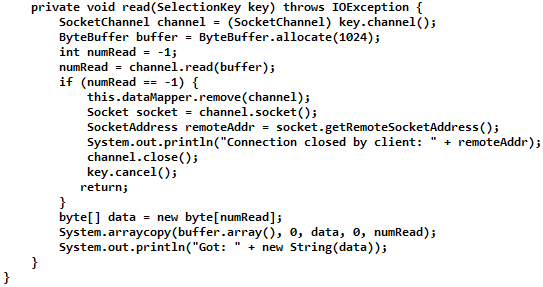


**Code:**

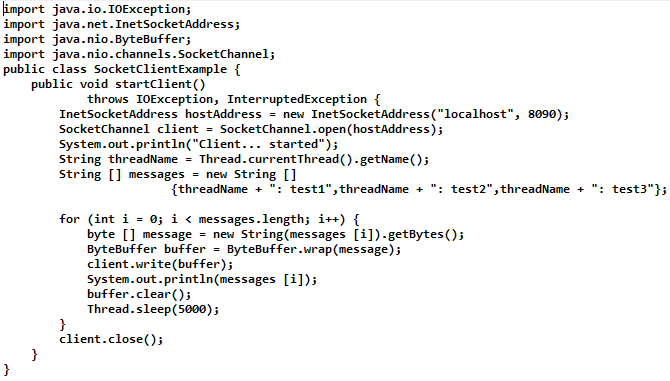
**SocketServerExample.java:**







**SocketClientExample.java:**



**Output:**

