**Lab # 02**

**Objective: To become familiar with Stream Socket API.**

**TASK1:**

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

**ConnectionAcceptor :**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package connectionacceptor;

import java.net.\*;

import java.io.\*;

public class ConnectionAcceptor {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

if(args.length!=2){

System.out.println("this program required two connectionline arguments");

}

else{

try{

int portNo=Integer.parseInt(args[0]);

String message=args[1];

ServerSocket connectionSocket=new ServerSocket(portNo);

System.out.println("now to accept a connection");

Socket dataSocket=connectionSocket.accept();

System.out.println("connection accepted");

OutputStream outStream=dataSocket.getOutputStream();

PrintWriter socketOutput=new PrintWriter(new OutputStreamWriter(outStream));

socketOutput.println(message);

socketOutput.flush();

System.out.println("message sent ");

dataSocket.close();

System.out.println("data Socket Closed");

Thread.sleep(2000);

}

catch(Exception exp){

exp.printStackTrace();

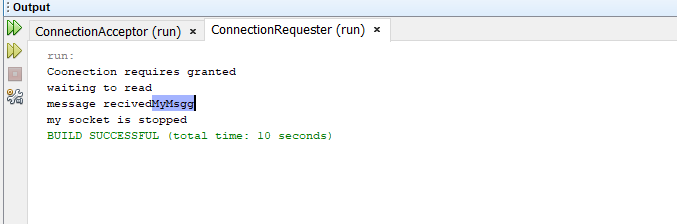
}

}

}

}

outputs



**ConnectionRequester :**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package connectionrequester;

import java.net.\*;

import java.io.\*;

public class ConnectionRequester {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

if(args.length!=2){

System.out.println("Require two arguments");

}

else{

try{

InetAddress acceptorHost=InetAddress.getByName(args[0]);

int acceptorPort=Integer.parseInt(args[1]);

Socket mysSocket=new Socket(acceptorHost,acceptorPort);

System.out.println("Coonection requires granted");

InputStream inputStream=mysSocket.getInputStream();

BufferedReader socketInput=new BufferedReader(new InputStreamReader(inputStream));

System.out.println("waiting to read");

String message=socketInput.readLine();

System.out.println("message recived"+message);

mysSocket.close();

System.out.println("my socket is stopped");

Thread.sleep(10000);

}

catch(Exception exp){

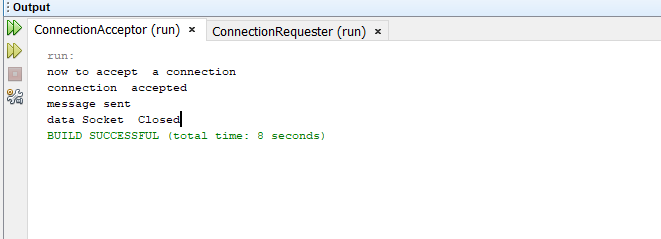
exp.printStackTrace();

}

}

}

}



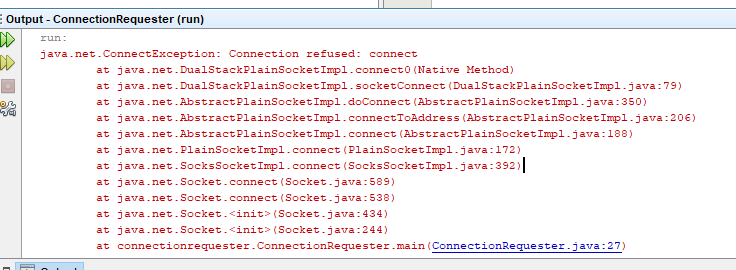
**Description:** First we start the acceptor that accepts the request and responds accordingly and then requestor that request any service. In above case acceptor starts and prepares its service that is a message in this case and then requestor is started in order to make a request that required the name and port number of acceptor in order to establish a connection and get the data or service after it receives the message the acceptor flushes all the data (some like session ends) and close the socket and ends the connection. We can say here the acceptor class is working as a server while requestor is working as receiver.

**TASK2:**

**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.**

Outputs:

ConnectionRequester to run the Exception connection refused



**Description:** Exception occurs as there is no acceptor to accept the requestors connection.

**TASK3:**

**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.**

**connectionacceptor**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package connectionacceptor;

import java.net.\*;

import java.io.\*;

public class ConnectionAcceptor {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

if(args.length!=2){

System.out.println("this program required two connectionline arguments");

}

else{

try{

int portNo=Integer.parseInt(args[0]);

String message=args[1];

ServerSocket connectionSocket=new ServerSocket(portNo);

System.out.println("now to accept a connection");

Socket dataSocket=connectionSocket.accept();

System.out.println("connection accepted");

OutputStream outStream=dataSocket.getOutputStream();

PrintWriter socketOutput=new PrintWriter(new OutputStreamWriter(outStream));

Thread.sleep(5000);

socketOutput.println(message);

socketOutput.flush();

System.out.println("message sent ");

dataSocket.close();

System.out.println("data Socket Closed");

Thread.sleep(2000);

}

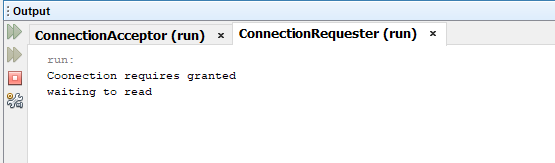
catch(Exception exp){

exp.printStackTrace();

} } }

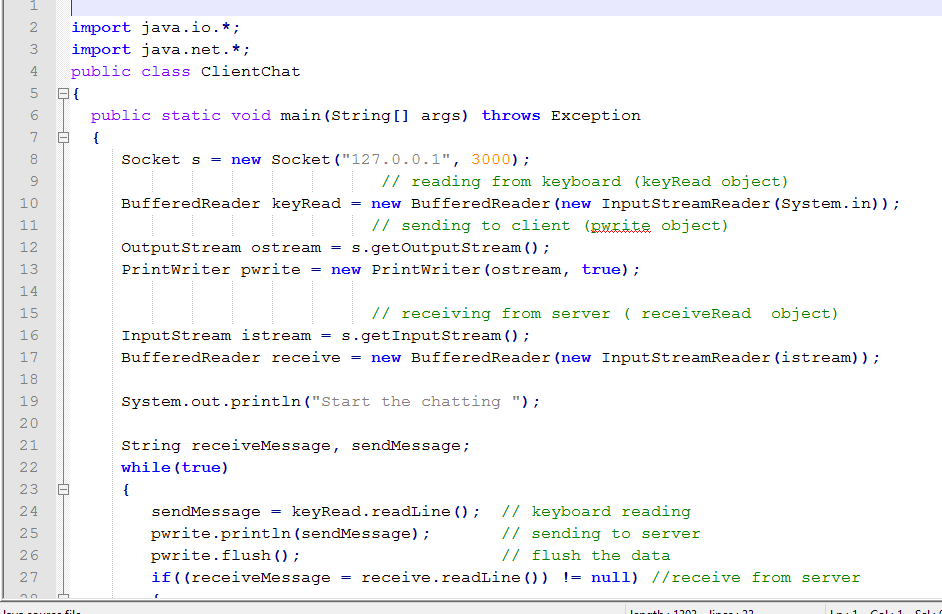
}

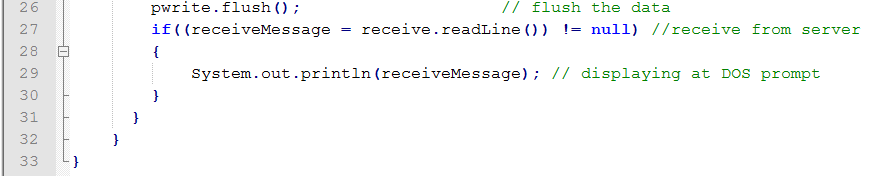
Outputs



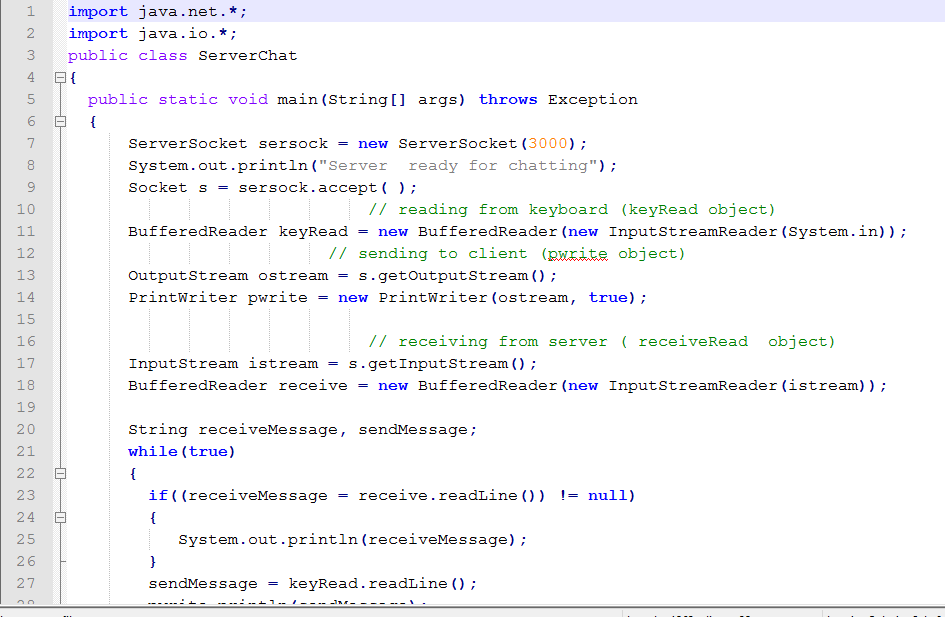
**Task 4:- Modify the sample code to include two way communication between the client and the server.**

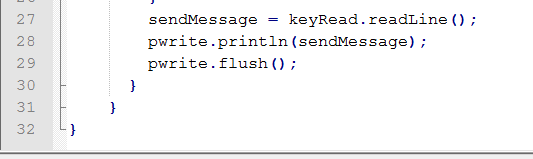
**Client:**

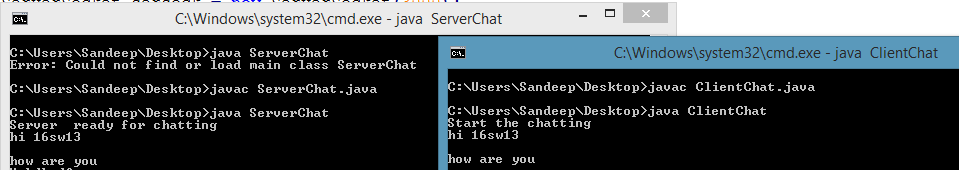




**Server:**

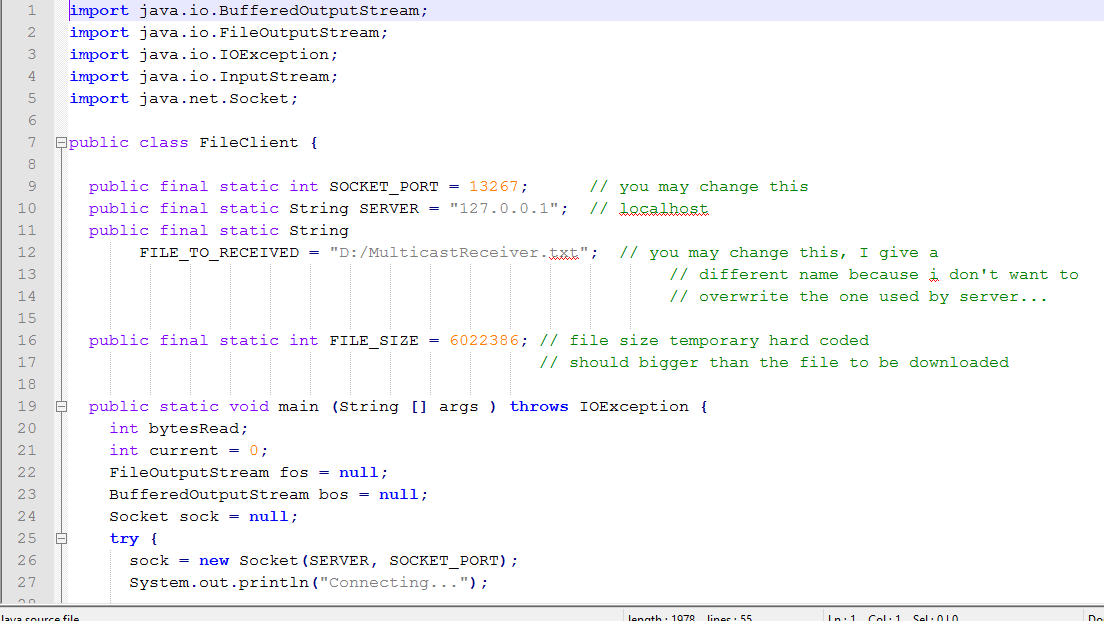


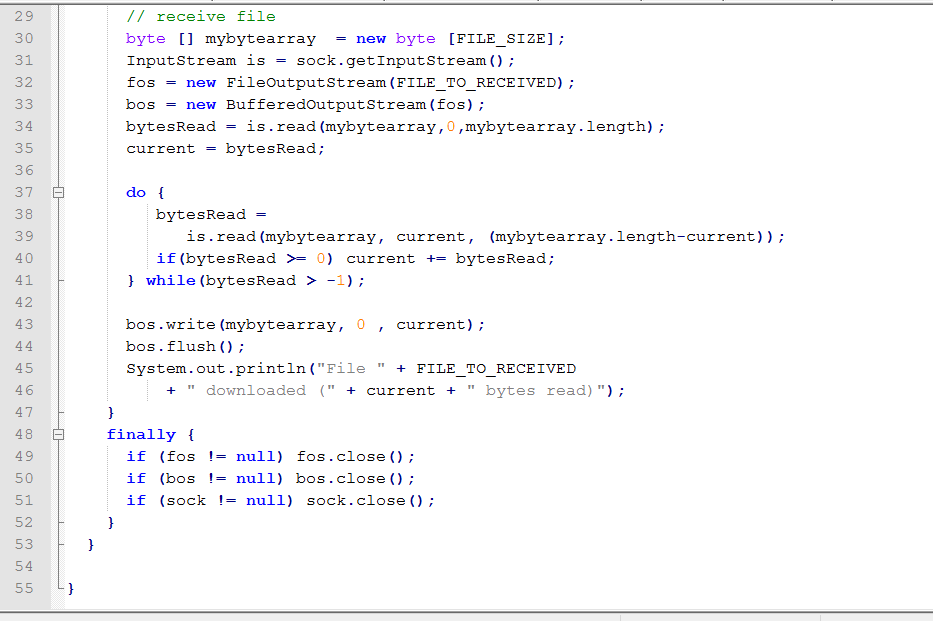




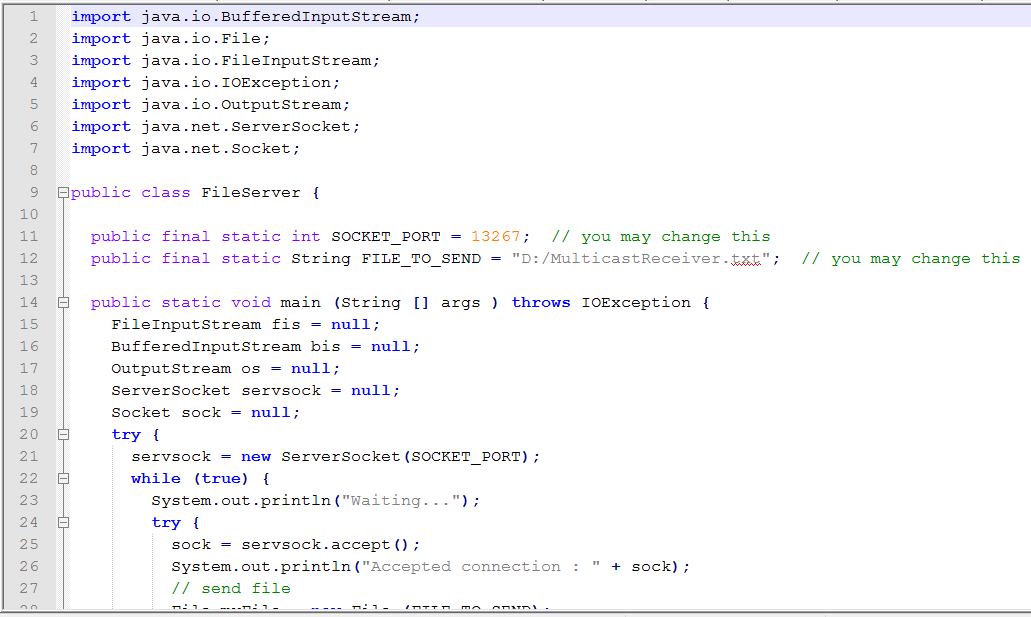
**Task 5:Modify the sample code to send complete files between the client to the server.**

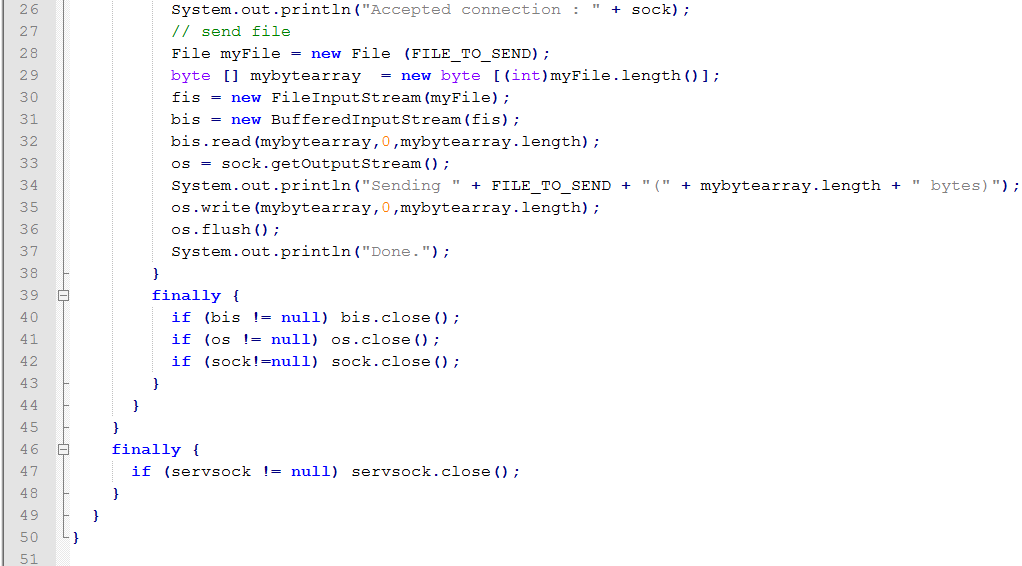
**Client:**

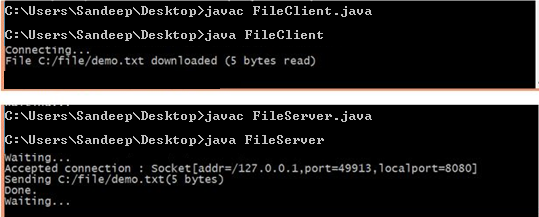




**Server:**

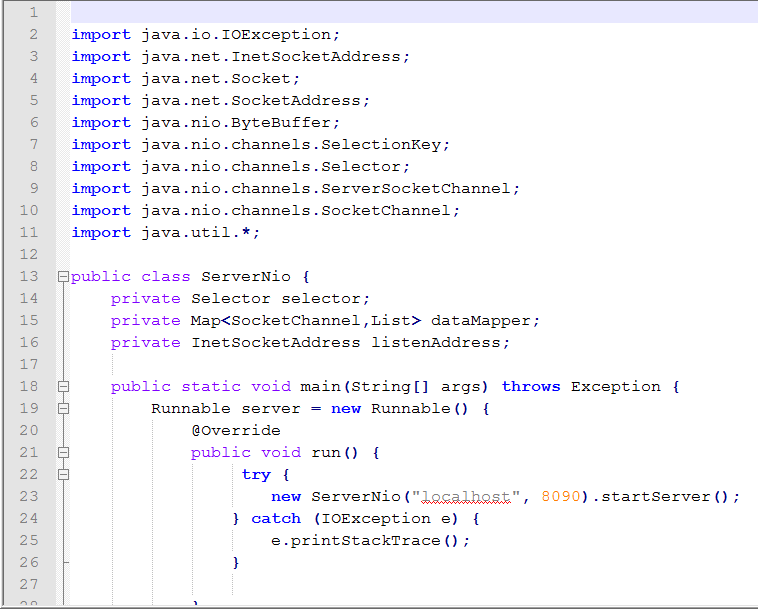


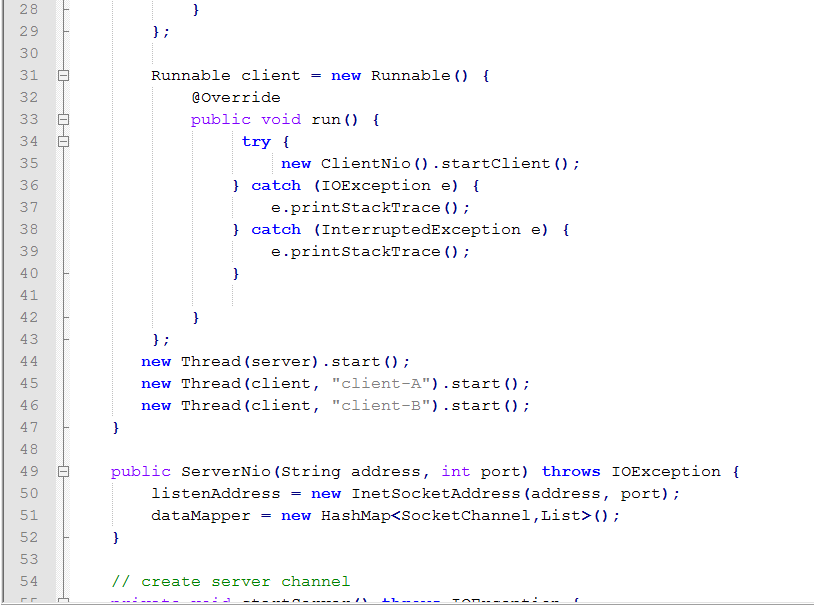


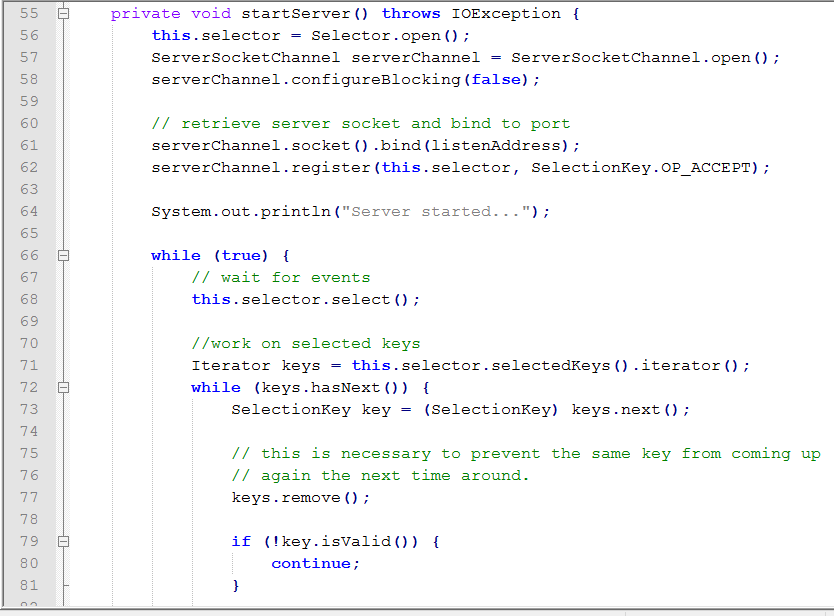


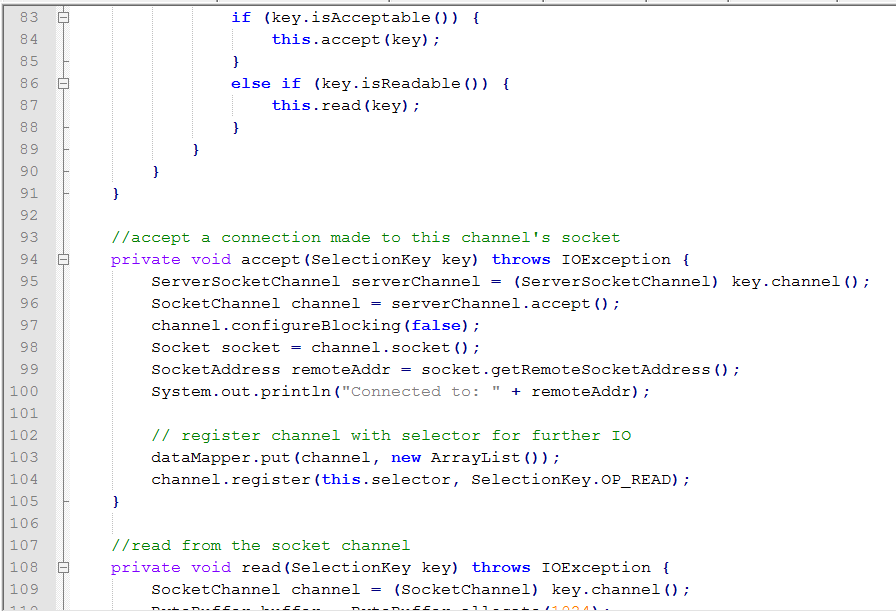
**Task 6: Explore the non-blocking java socket API in the nio package and implement a sample program.**

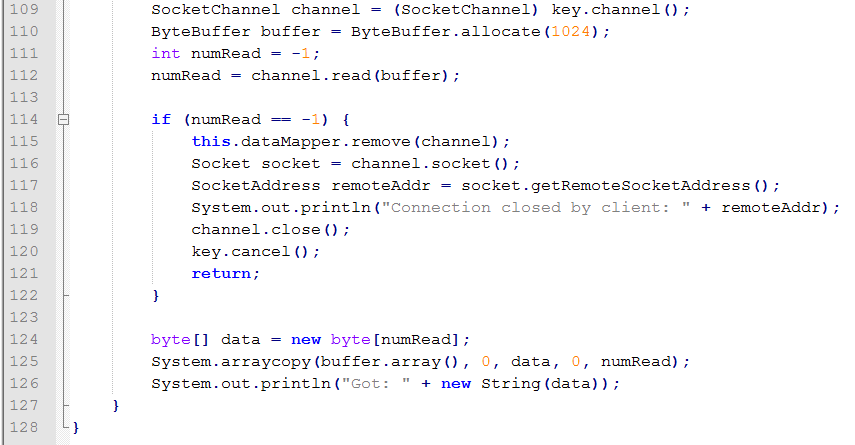
**Server:**











**Client:**

