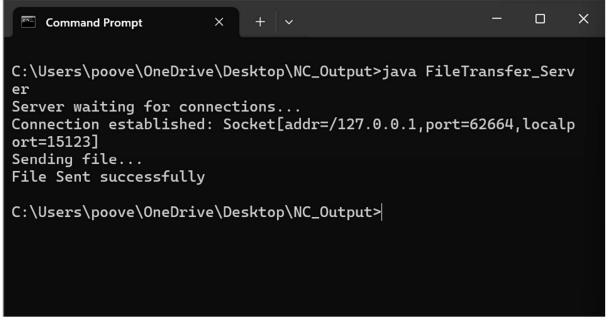
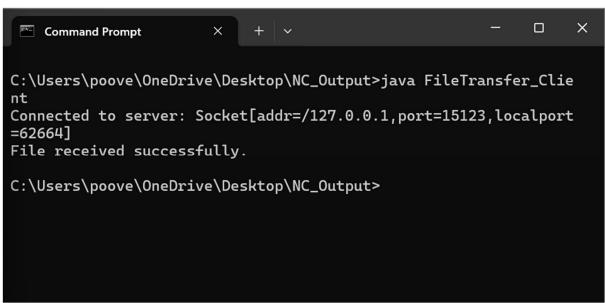
File Transfer Server:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
class FileTransfer_Server
{
  public static void main(String[] args) throws IOException
 {
    ServerSocket serverSocket = new ServerSocket(15123);
    System.out.println("Server waiting for connections...");
    Socket socket = serverSocket.accept();
    System.out.println("Connection established: " + socket);
    File transferFile = new File("FileTransfer_File.txt");
    byte[] bytearray = new byte[(int) transferFile.length()];
    FileInputStream fin = new FileInputStream(transferFile);
    BufferedInputStream bin = new BufferedInputStream(fin);
    bin.read(bytearray, 0, bytearray.length);
    OutputStream os = socket.getOutputStream();
    System.out.println("Sending file...");
    os.write(bytearray, 0, bytearray.length);
    os.flush();
    System.out.println("File Sent successfully");
    socket.close();
    serverSocket.close();
 }
}
```

File Transfer Client:

```
import java.io.*;
import java.net.Socket;
public class FileTransfer_Client
{
  public static void main(String[] args) throws IOException
 {
    Socket socket = new Socket("127.0.0.1", 15123);
    System.out.println("Connected to server: " + socket);
    InputStream is = socket.getInputStream();
    FileOutputStream fos = new FileOutputStream("FileTransfer_File.txt");
    BufferedOutputStream bos = new BufferedOutputStream(fos);
    int bufferSize = 1024;
    byte[] bytearray = new byte[bufferSize];
    int bytesRead;
    while ((bytesRead = is.read(bytearray, 0, bufferSize)) != -1)
       bos.write(bytearray, 0, bytesRead);
    bos.flush();
    bos.close();
    System.out.println("File received successfully.");
    socket.close();
 }
}
```



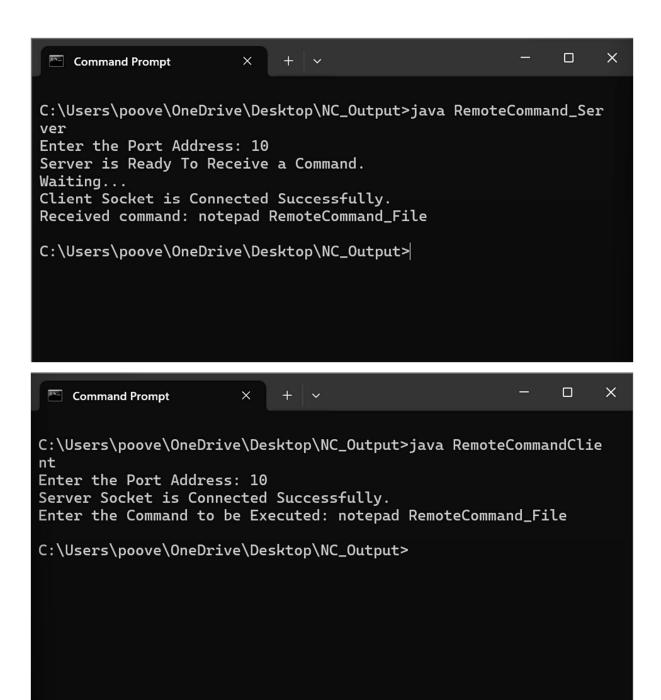


Remote Command Server:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
class RemoteCommand_Server
 public static void main(String[] args) throws IOException
 {
   Scanner scanner = new Scanner(System.in);
   System.out.print("Enter the Port Address: ");
   int port = Integer.parseInt(scanner.nextLine());
   try (ServerSocket serverSocket = new ServerSocket(port)) {
     System.out.println("Server is Ready To Receive a Command.");
     System.out.println("Waiting...");
     try (Socket clientSocket = serverSocket.accept()) {
       if (clientSocket.isConnected())
         System.out.println("Client Socket is Connected Successfully.");
       InputStream in = clientSocket.getInputStream();
       OutputStream out = clientSocket.getOutputStream();
       BufferedReader reader = new BufferedReader(new InputStreamReader(in));
       PrintWriter writer = new PrintWriter(out, true);
       String cmd = reader.readLine();
       System.out.println("Received command: " + cmd);
       ProcessBuilder processBuilder = new ProcessBuilder(cmd.split("\\s+"));
       Process process = processBuilder.start();
       try (BufferedReader commandOutput = new BufferedReader(new
InputStreamReader(process.getInputStream()))) {
         String outputLine;
         while ((outputLine = commandOutput.readLine()) != null)
           writer.println(outputLine);
       }
     }
   }
 }
}
```

Remote Command Client:

```
import java.io.*;
import java.net.Socket;
import java.util.Scanner;
class RemoteCommandClient
  public static void main(String[] args) throws IOException
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the Port Address: ");
    int port = Integer.parseInt(scanner.nextLine());
    try (Socket socket = new Socket("localhost", port)) {
     if (socket.isConnected())
       System.out.println("Server Socket is Connected Successfully.");
      InputStream in = socket.getInputStream();
      OutputStream out = socket.getOutputStream();
BufferedReader userInputReader = new BufferedReader(new InputStreamReader(System.in));
      BufferedReader serverReader = new BufferedReader(new InputStreamReader(in));
      PrintWriter writer = new PrintWriter(out, true);
     System.out.print("Enter the Command to be Executed: ");
     String command = userInputReader.readLine();
     writer.println(command);
     String serverOutput;
     while ((serverOutput = serverReader.readLine()) != null)
       System.out.println("Server Output: " + serverOutput);
   }
  }
}
```

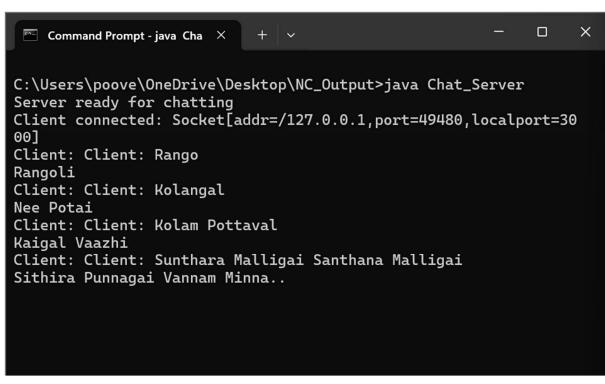


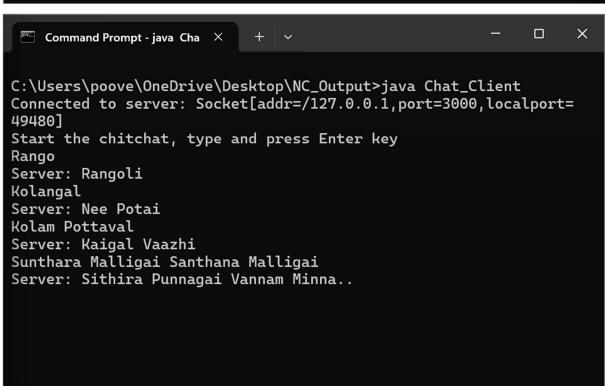
Chat Server:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
class Chat_Server
  public static void main(String[] args) throws IOException
 {
   ServerSocket serverSocket = new ServerSocket(3000);
   System.out.println("Server ready for chatting");
    Socket clientSocket = serverSocket.accept();
    System.out.println("Client connected: " + clientSocket);
    BufferedReader keyRead = new BufferedReader(new
InputStreamReader(System.in));
    OutputStream ostream = clientSocket.getOutputStream();
    PrintWriter pwrite = new PrintWriter(ostream, true);
    InputStream istream = clientSocket.getInputStream();
    BufferedReader receiveRead = new BufferedReader(new
InputStreamReader(istream));
    String receiveMessage, sendMessage;
   while (true) {
     if ((receiveMessage = receiveRead.readLine()) != null)
       System.out.println("Client: " + receiveMessage);
     sendMessage = keyRead.readLine();
     pwrite.println("Server: " + sendMessage);
     pwrite.flush();
   }
 }
}
```

Chat Client:

```
import java.io.*;
import java.net.Socket;
class Chat_Client
  public static void main(String[] args) throws IOException
   Socket socket = new Socket("127.0.0.1", 3000);
   System.out.println("Connected to server: " + socket);
    BufferedReader keyRead = new BufferedReader(new
InputStreamReader(System.in));
    OutputStream ostream = socket.getOutputStream();
    PrintWriter pwrite = new PrintWriter(ostream, true);
    InputStream istream = socket.getInputStream();
    BufferedReader receiveRead = new BufferedReader(new
InputStreamReader(istream));
    String receiveMessage, sendMessage;
    System.out.println("Start the chitchat, type and press Enter key");
   while (true) {
     sendMessage = keyRead.readLine();
     pwrite.println("Client: " + sendMessage);
     pwrite.flush();
     if ((receiveMessage = receiveRead.readLine()) != null)
       System.out.println(receiveMessage);
   }
 }
}
```





Concurrent Server Server:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
class ConcurrentServer_Server
{
  public static void main(String[] args) throws IOException
 {
   ServerSocket ss = new ServerSocket(8500);
   System.out.println("Waiting for clients...");
   ExecutorService executorService = Executors.newFixedThreadPool(5);
   while (true) {
     Socket clientSocket = ss.accept();
     System.out.println("Client connected: " + clientSocket);
     executorService.execute(new ConcurrentServer_ClientHandler(clientSocket));
   }
 }
}
class ConcurrentServer_ClientHandler implements Runnable
{
  private final Socket clientSocket;
  public ConcurrentServer_ClientHandler(Socket clientSocket)
 {
   this.clientSocket = clientSocket;
 }
  @Override
  public void run()
   try {
     BufferedReader br = new BufferedReader(new
```

```
InputStreamReader(clientSocket.getInputStream()));
    String clientName = br.readLine();
    System.out.println("\nCLIENT NAME: " + clientName);
    int num = Integer.parseInt(br.readLine());
    int square = num * num;
    PrintWriter pw = new PrintWriter(clientSocket.getOutputStream(), true);
    pw.println(square);
    System.out.println("OUTPUT - The square of " + num + " is " + square);
    clientSocket.close();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```

Concurrent Server Client1:

```
import java.io.*;
import java.net.Socket;
public class ConcurrentServer_Client1
 public static void main(String[] args) throws IOException
   Socket socket = new Socket("localhost", 8500);
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   int num = Integer.parseInt(br.readLine());
   PrintWriter pw = new PrintWriter(socket.getOutputStream(), true);
   pw.println("Client 1");
   pw.println(num);
   BufferedReader br1 = new BufferedReader(new
   InputStreamReader(socket.getInputStream()));
   int square = Integer.parseInt(br1.readLine());
   System.out.println("Square of " + num + " is " + square + "\n");
   socket.close();
 }
}
Concurrent Server Client2:
import java.io.*;
import java.net.Socket;
public class ConcurrentServer_Client2
 public static void main(String[] args) throws IOException
   Socket s = new Socket("localhost", 8500);
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   int num = Integer.parseInt(br.readLine());
   PrintWriter pw = new PrintWriter(s.getOutputStream(), true);
   pw.println("Client 2");
   pw.println(num);
   BufferedReader br1 = new BufferedReader(new
InputStreamReader(s.getInputStream()));
   double squareRoot = Double.parseDouble(br1.readLine());
   System.out.println("Square of " + num + " is " + squareRoot);
   s.close();
 }
}
```

