**Laboratory 2**

Title of the Laboratory Exercise: Basic Client server Programs

1. Introduction and Purpose of Experiment

A basic one-way Client and Server setup where a Client connects, sends messages to server and the server shows them using socket connection.

Aim and Objectives

Aim

* To do socket programming with java

2. Experimental Procedure

* + 1. Analyse the problem statement
    2. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
    3. Implement the algorithm in C language
    4. Compile the C program
    5. Test the implemented program
    6. Document the Results
    7. Analyse and discuss the outcomes of your experiment

1. **Questions**

Implement the following using Java

* Basic Client Server Communication using UDP
* Basic Client Server Communication using TCP

1. **Calculations/Computations/Algorithms**

**Server TCP ( Algorithm)**

1. Start
2. Class server\_TCP
3. Initialize all the sockets and input streams
4. Create a constructor with port
   1. Try Block
      1. Create a Server Object with Port Number
      2. Accept the Connection
      3. Get the input buffer
      4. While Loop Until ( Over Keyword is passed on )
         1. Try Block
            1. Read the Line using UTF
         2. Catch Block
            1. Catch the exception if there is
      5. Close the connection

b. Catch Block

1. Catch the exception if there

5. Main Function

a. Create an Object of the Class (server\_TCP) and listen at port 5000

6. Stop

**Client TCP ( Algorithm)**

1. Start
2. Class Client\_TCP
3. Initialize all the sockets and input and Output streams
4. Create a constructor with port and IP Address
   1. Try Block
      1. Create a Server Object with Port Number and IP Address
      2. Connected
      3. Input the buffer steam
      4. Output the buffer steam

b. Catch Block

1. Catch the Exceptions if the host is not known

c. Catch Block

1. Catch the Exceptions if the IO is not proper

5. While Loop Until ( Over Keyword is passed on )

* + - 1. Try Block
         1. Input the Line using UTF
      2. Catch Block
         1. Catch the exception if there is

1. Close the connection

b. Catch Block

1. Catch the exception if there

5. Main Function

a. Create an Object of the Class (Client\_TCP ) and listen at port 5000

6. Stop

**Server UDP ( Algorithm)**

1. Start
2. Class Server\_UDP
3. Initialize the port at 8080
4. Main Function
   1. Try Block
      1. Create Datagram Socket Object with the Port Number
      2. Take the incoming Data Buffer
      3. Receive the Input Buffer
      4. Get the data from the input Buffer
      5. Obtain the IP Address and Port from the client
      6. Create a UDP packet to send to client
      7. Send the created packet
      8. Close the connection
   2. Catch Block
      1. Catch the exception if there

4. Stop

**Client UDP ( Algorithm)**

1. Start
2. Class Client\_UDP
3. Initialize the port at 8080
4. Main Function
   1. Try Block
      1. Create Datagram Socket Object with the Port Number
      2. Get the IP Address
      3. Receive the Input Buffer
      4. Get the data from the input Buffer and Send it to that Port
      5. Receive the data if there is from the server
      6. Close the connection
   2. Catch Block
      1. Catch the exception if there

4. Stop

1. **Presentation of Results**

**Server TCP Code**

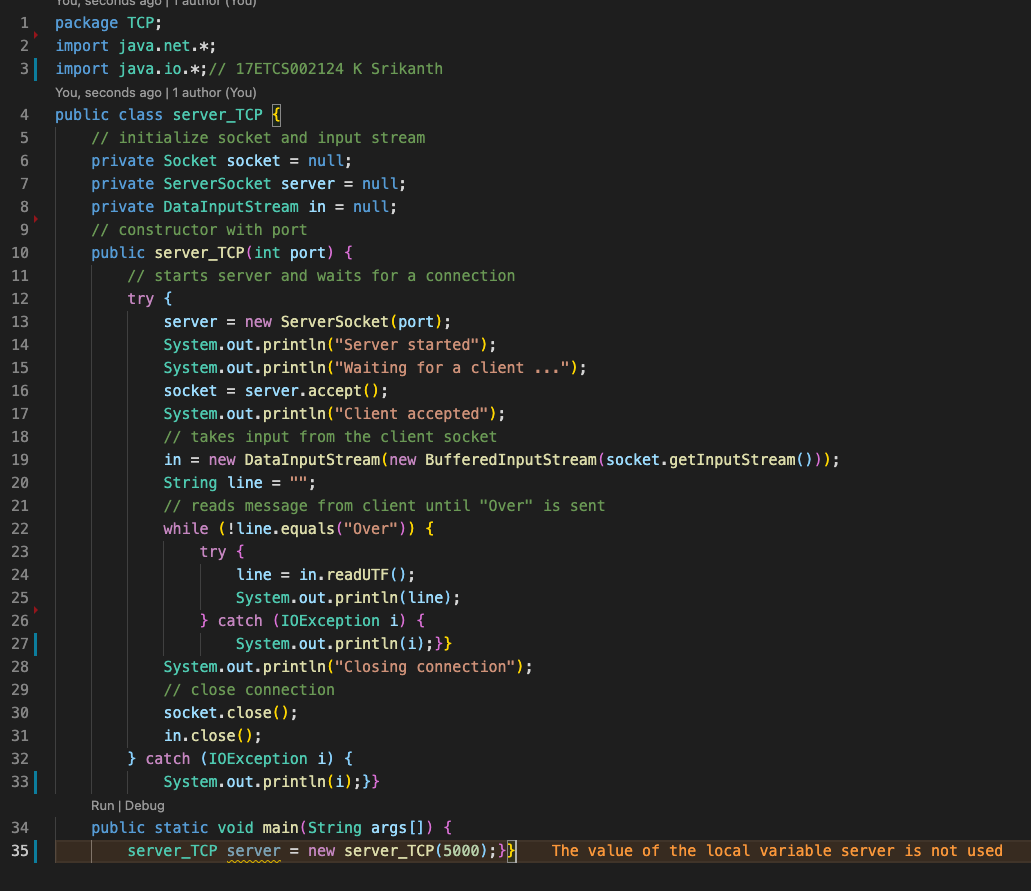
****

Figure 1 Java Program for TCP Server Side

**Client TCP Code**

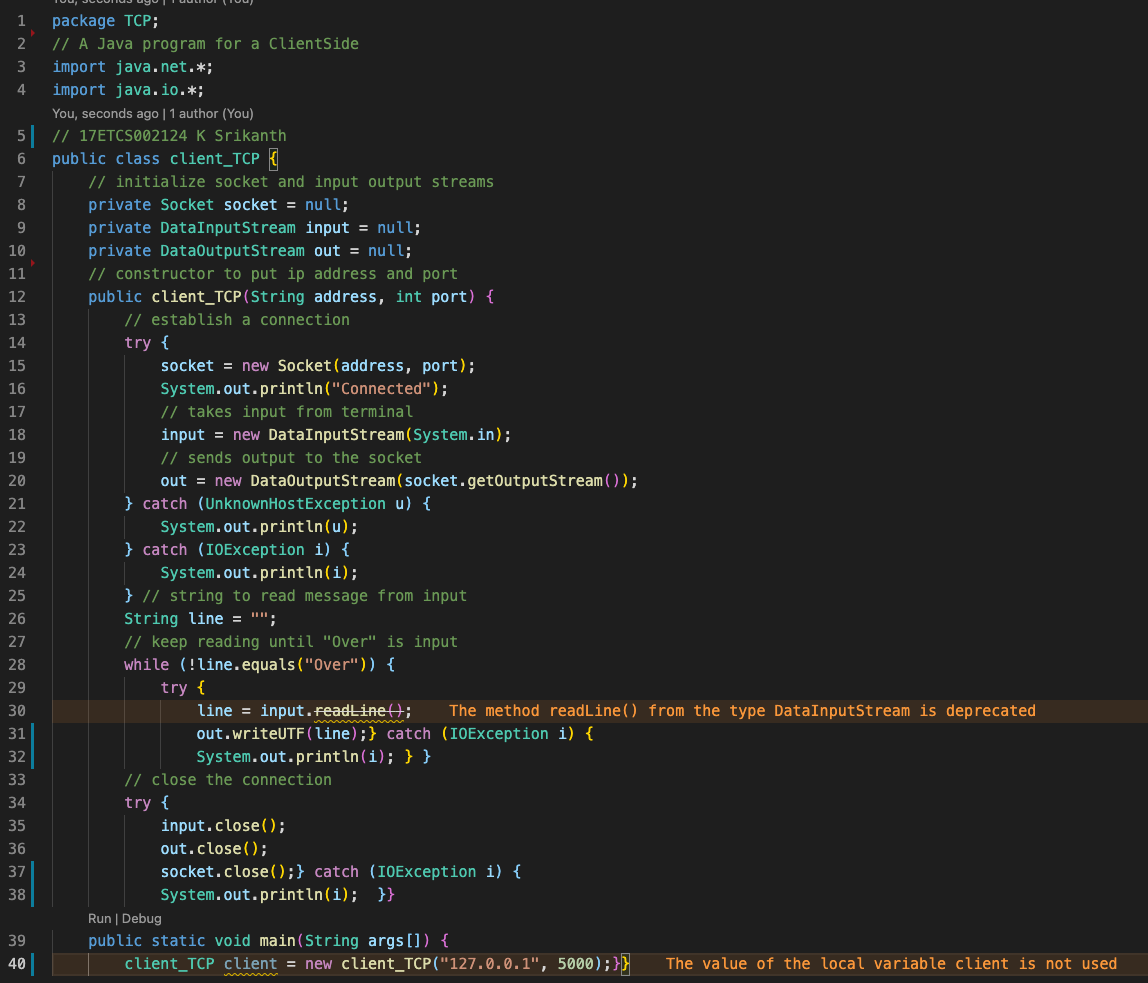
****

Figure 2 Java Program for TCP Client Side

**TCP Output for Server and Client**

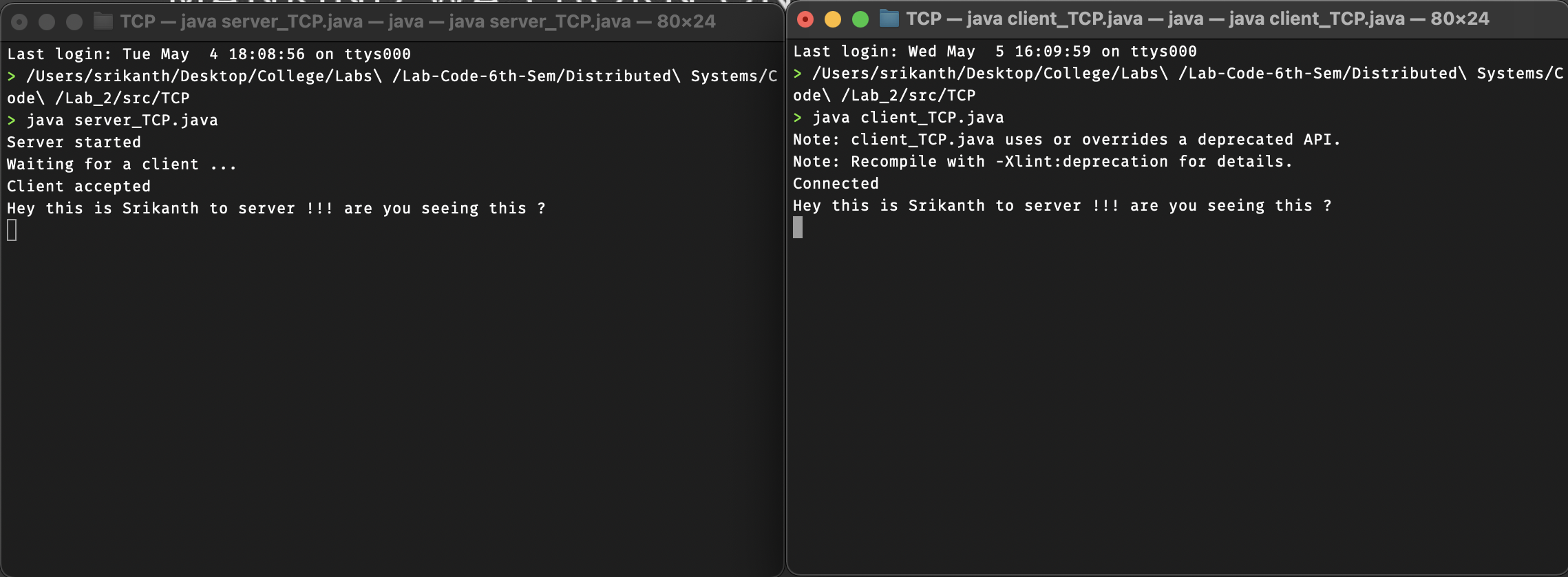
****

Figure 3 Java Program Output for TCP Connection

**Server UDP Code**

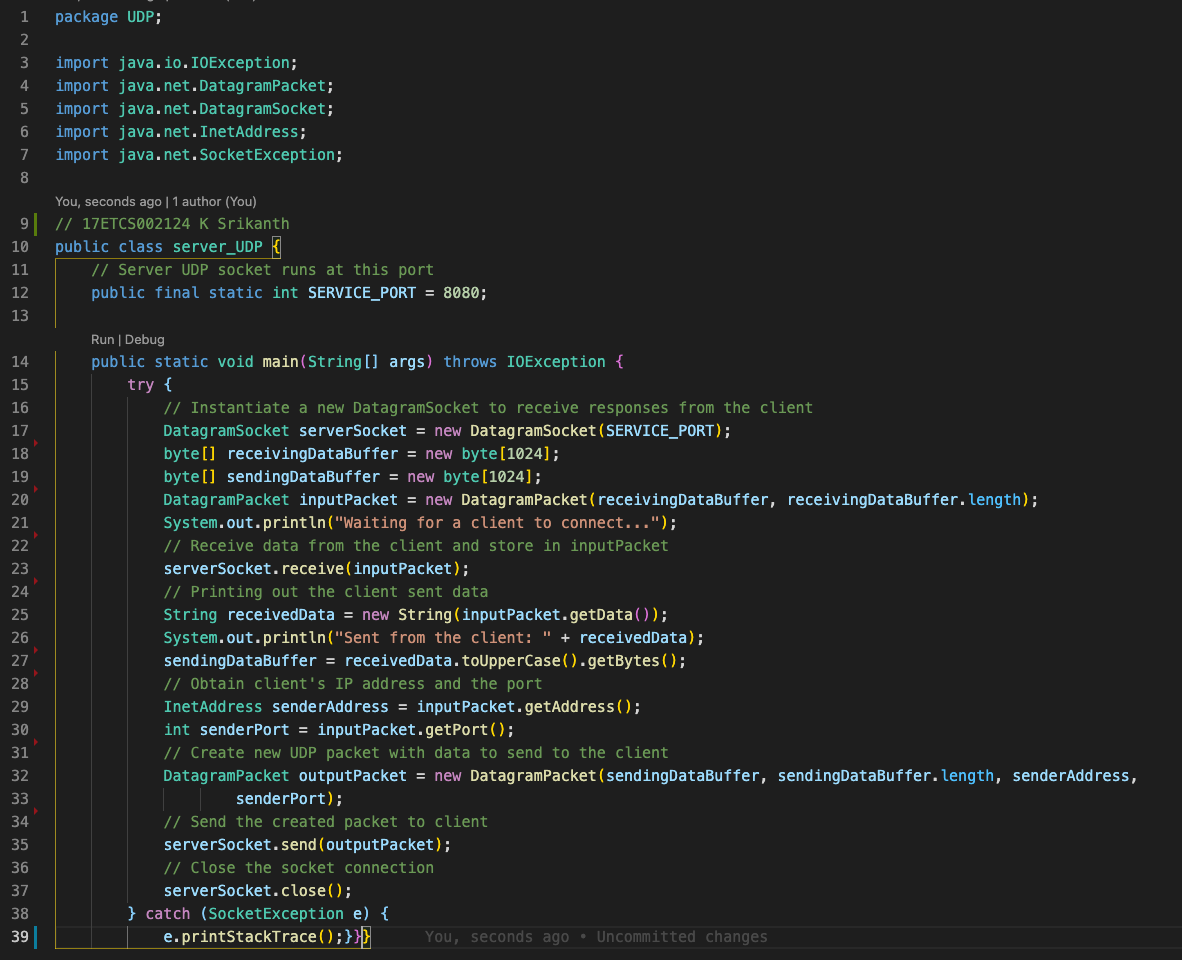
****

Figure 4 Java Program for UDP Server Side

**Client UDP Code**

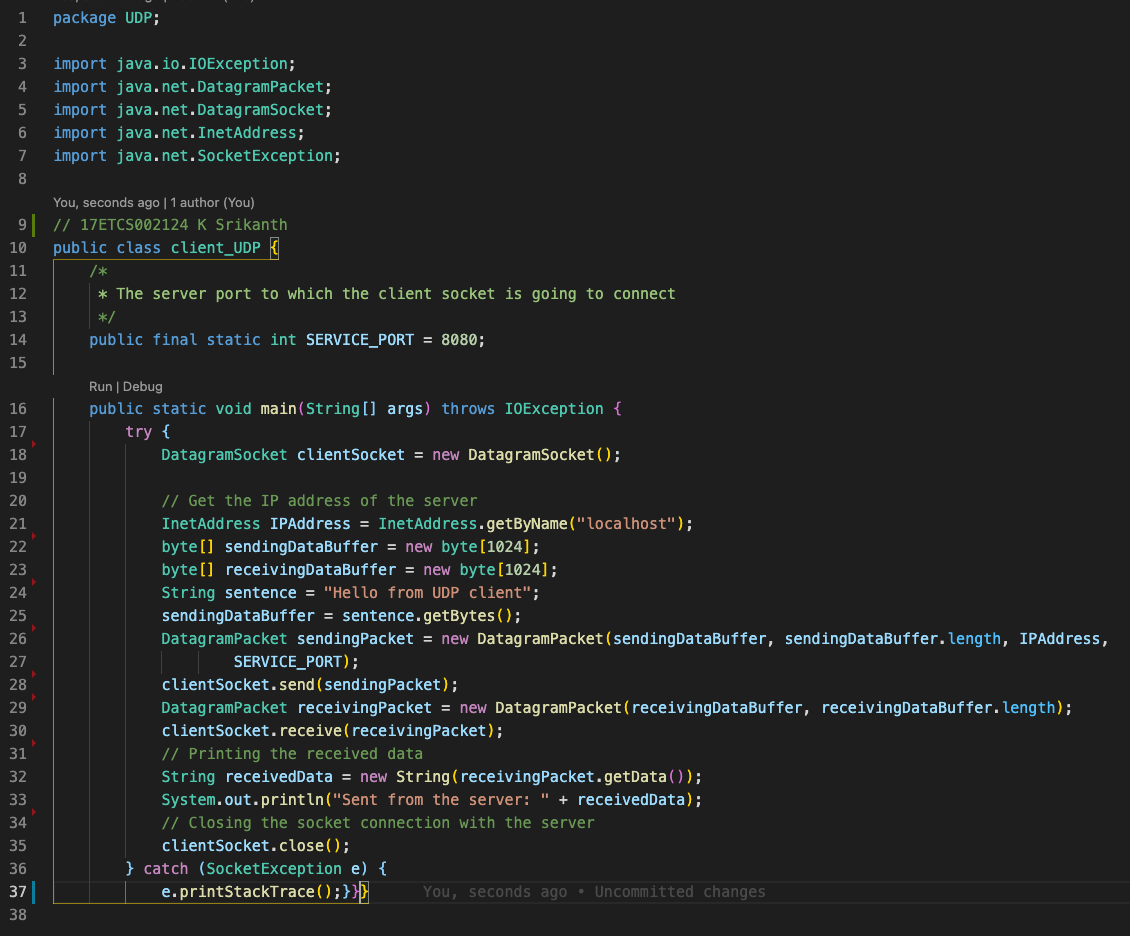


Figure 5 Java Program for UDP Client Side

**UDP Output for Server and Client**

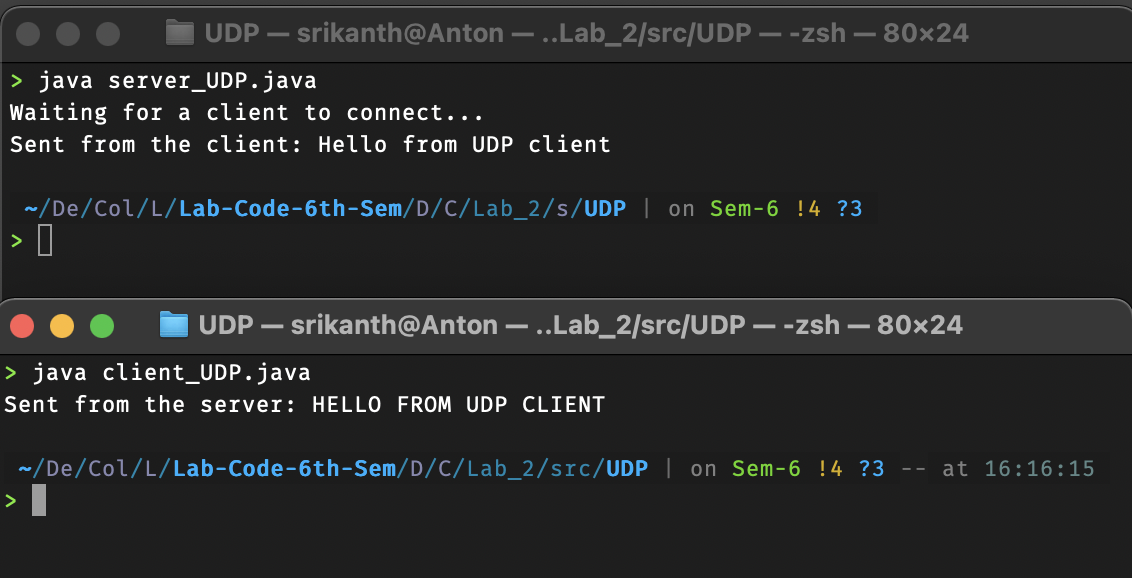
****

Figure 6 Java Program Output for UDP Connection

1. **Analysis and Discussions**

|  |  |
| --- | --- |
| **TCP (Transmission Control Protocol)** | **UDP (User Datagram Protocol)** |
| 1. It is a communications protocol, using which the data is transmitted between systems over the network. In this, the data is transmitted into the form of packets. It includes error-checking, guarantees the delivery and preserves the order of the data packets. | 1. It is same as the TCP protocol except this doesn’t guarantee the error-checking and data recovery.  If you use this protocol, the data will be sent continuously, irrespective of the issues in the receiving end. |
| 2. TCP is a connection oriented protocol | 2. UDP is a connection less protocol. |
| 3. As TCP provides error checking support and also guarantees delivery of data to the destination router this make it more reliable as compared to UDP. | 3. While on other hand UDP does provided only basic error checking support using checksum so the delivery of data to the destination cannot be guaranteed in UDP as compared to that in case of TCP. |
| 4. In TCP the data is transmitted in a particular sequence which means that packets arrive in-order at the receiver. | 4. On other hand there is no sequencing of data in UDP in order to implement ordering it has to be managed by the application layer. |
| 5. TCP is slower and less efficient in performance as compared to UDP. Also TCP is heavy-weight as compared to UDP. | 5. On other hand UDP is faster and more efficient than TCP. |
| 6. Retransmission of data packets is possible in TCP in case packet get lost or need to resend. | 6. On other hand retransmission of packets is not possible in UDP. |

**Datagram Packets Class**

Datagram packets are used to implement a connectionless packet delivery service. Each message is routed from one machine to another based solely on information contained within that packet. Multiple packets sent from one machine to another might be routed differently, and might arrive in any order. Packet delivery is not guaranteed.

* **DatagramPacket(byte[] buf, int length)**

Constructs a DatagramPacket for receiving packets of length (length).

* **DatagramPacket(byte[] buf, int length, InetAddress address, int port)**

Constructs a datagram packet for sending packets of length (length) to the specified port number on the specified host.

**Datagram Socket Class**

A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

**1. Limitations of Experiments**

Data corruption is a common occurrence on the Internet, UDP is not good at error detection. Also No compensation for lost packets and the Packets can arrive out of order as this on the problem that No congestion control UDP may be light weight, but not that reliable.

**2. Limitations of Results**

UDP Packets are not that reliable

**3. Learning happened**

Learned about Socket Programming of UDP and TCP Protocols in Java.