**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 programing

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. **Algorithm/Pseudocodes**

**For Basic Client Server Communication using UDP**

**UDP Server :**

Create UDP socket.

Bind the socket to server address.

Wait until datagram packet arrives from client.

Process the datagram packet and send a reply to client.

Go back to Step 3.

**UDP Client :**

Create UDP socket.

Send message to server.

Wait until response from server is recieved.

Process reply and go back to step 2, if necessary.

Close socket descriptor and exit.

In this Program I have used bye as a keyword to end conversation between Client and Server.

**For Basic Client Server Communication using TCP**

**TCP Client :**

1. The client initiates connection to a server specified by hostname/IP address and port number.

2. Send data to the server using an OutputStream.

3. Read data from the server using an InputStream.

4. Close the connection.

The steps 2 and 3 can be repeated many times depending on the nature of the communication.

**TCP Server:**

1.Create a server socket and bind it to a specific port number

2. Listen for a connection from the client and accept it. This results in a client socket is created for the connection.

3. Read data from the client via an InputStream obtained from the client socket.

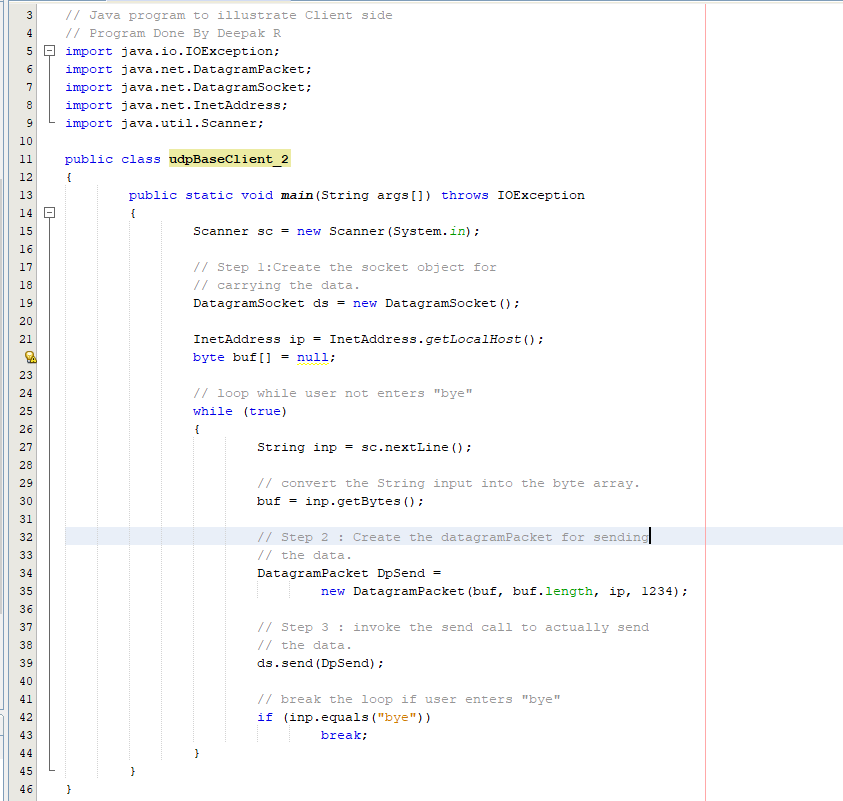
4. Send data to the client via the client socket’s OutputStream.

5. Close the connection with the client.-

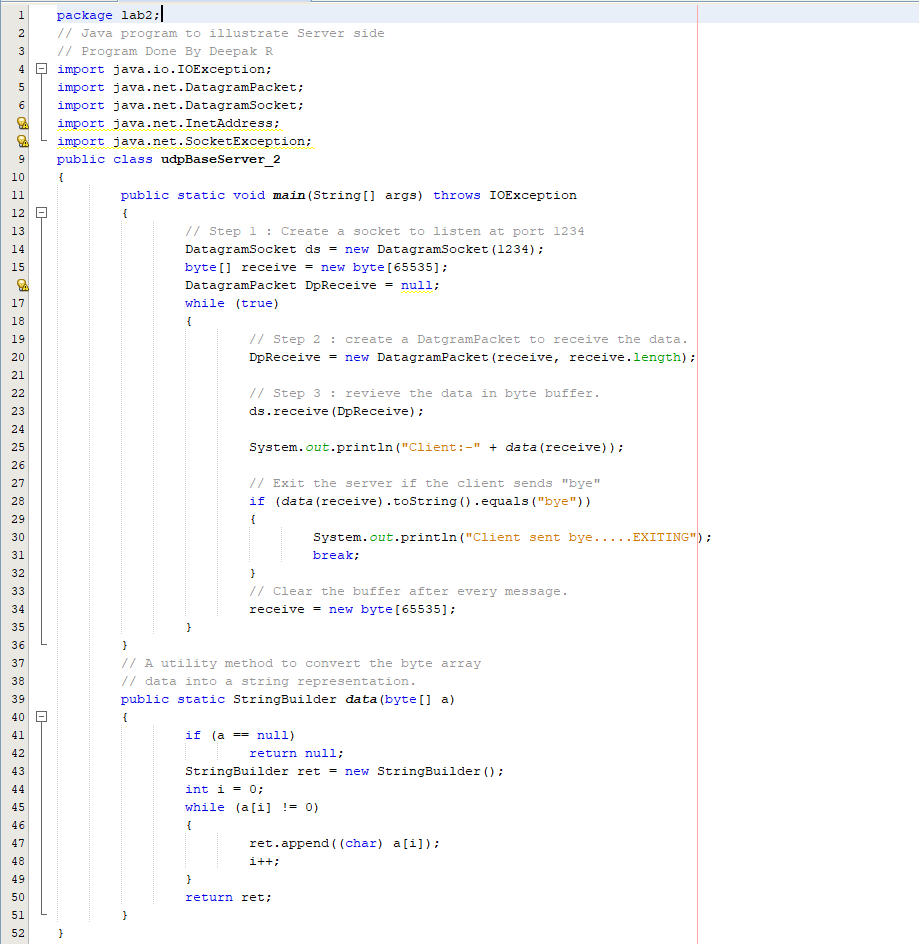
**5.Programs (Source Code)**

**For Basic Client Server Communication using UDP**

**Client Side**

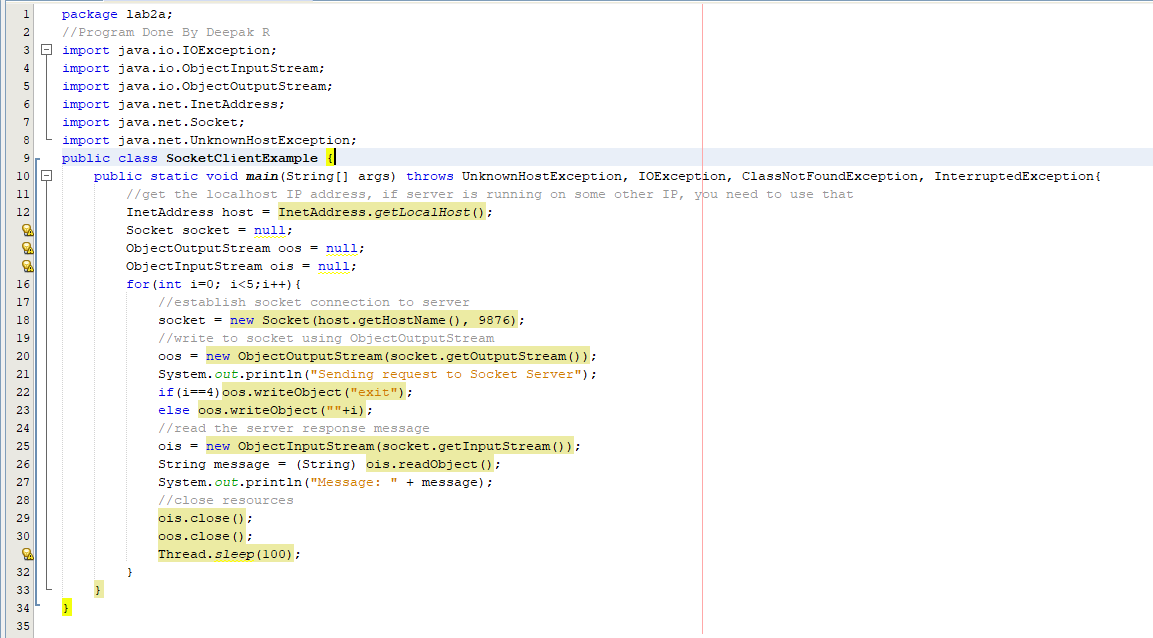


**Server Side**

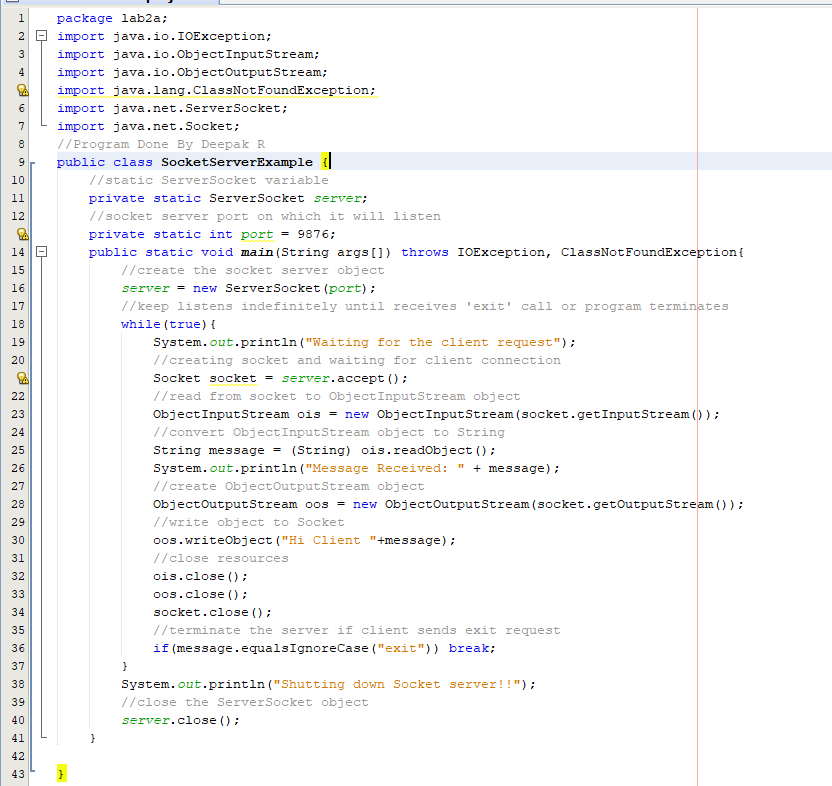


**For Basic Client Server Communication using UDP**

**Client Side**



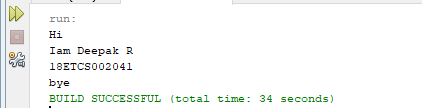
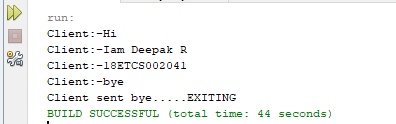
**Server Side**

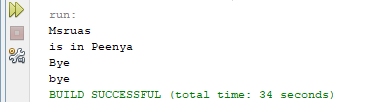
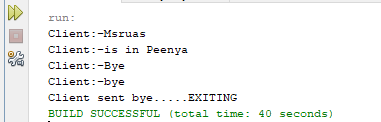


**Outputs/Testcases**

**For Basic Client Server Communication using UDP**

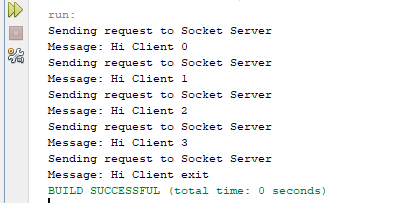
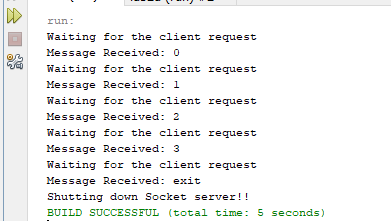
**Client Side** **Server Side**

**For Basic Client Server Communication using UDP**

**Client Side** **Server Side**

1. **Analysis and Discussions**

If we are creating a connection between client and server using TCP then it has few functionality like, TCP is suited for applications that require high reliability, and transmission time is relatively less critical. It is used by other protocols like HTTP, HTTPs, FTP, SMTP, Telnet. TCP rearranges data packets in the order specified. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.

In UDP, the client does not form a connection with the server like in TCP and instead, It just sends a datagram. Similarly, the server need not to accept a connection and just waits for datagrams to arrive. We can call a function called connect() in UDP but it does not result anything like it does in TCP. There is no 3 way handshake. It just checks for any immediate errors and store the peer’s IP address and port number. connect() is storing peers address so no need to pass server address and server address length arguments in sendto().

**1. Limitations of Experiments**

None

**2. Limitations of Results**

Checked only for Small Samples it results may or may not vary if checked for large data.

**3. Learning happened**

We learned about Basic Client Server Communication using UDP and Basic Client Server Communication using TCP which are basic one-way Client and Server setup where a Client connects, sends messages to server and the server shows them using socket connection.

**4. Recommendations**

None