Ex No: 4

Date:

IMPLEMENTATION OF UDP USING SOCKET PROGRAMMING

AIM:

To study the concept of Socket Programming using UDP and execute the same using Java Programming in command prompt.

THEORY:

The **User Datagram Protocol (UDP)** is a connectionless protocol that provides fast, lightweight communication between devices over a network. Unlike TCP, UDP does not establish a connection or guarantee reliable data delivery, making it suitable for real-time applications like video streaming, VoIP, and online gaming.

In **socket programming**, UDP is implemented using sockets for sending and receiving datagrams. It operates on a "fire-and-forget" model, where packets (datagrams) are transmitted without waiting for acknowledgments or establishing a connection.

Key characteristics of UDP include:

- **Connectionless Communication**: No prior handshake is required; data is sent directly to the recipient.
- **No Error Recovery**: Lost or corrupted packets are not retransmitted, reducing overhead.
- **Low Latency**: Due to the absence of acknowledgment mechanisms, UDP provides faster data transmission.

Socket programming with UDP involves:

1. **Server-Side**:
    - Create a socket.
    - Bind the socket to a specific port.
    - Receive datagrams from clients.
    - Optionally send responses.
2. **Client-Side**:
    - Create a socket.

- Send datagrams to the server's address and port.
- Optionally receive responses.

UDP is ideal for applications where speed is critical, and occasional data loss is acceptable. However, it requires the application to handle reliability and error correction if needed.

ALGORITHM:

Server:

1. Import the necessary Libraries which include util,io,net.
2. Inside main method use a try block for writing the code establishing the connection.
3. For Server side connection, we create object of the DatagramSocket class.
4. To connect to the localhost we use InetAddress object and pass the localhost name to the constructor.
5. Then we use object of DatagramPacket class to send the messages from command prompt.
6. From the command prompt we get the Serve's command as a String and convert it inot byte array and the send it to client using send() method.
7. We use another object of Datagram Class to receive the reply from the client.
8. We convert the received message into byte array and then to string and print the same in command prompt.

Client:

1. Import the necessary Libraries which include util,io,net.
2. Inside main method use a try block for writing the code establishing the connection.
3. For Server side connection, we create object of the DatagramSocket class and pass the port number as argument for the constructor.
4. We create an object for DatagramPacket class to receive the data sent by the Server.
5. We receive the data in the form of byte array by using the receive() method of DatagramPacket class and then convert it into String and display it.
6. We send an Ok message to the Server by converting it into byte and sending it using send method of DatagramPackect class.

PROGRAM:

SERVER PROGRAM:

```
import java.net.*;
import java.util.Scanner;

public class udpServer {
    public static void main(String[] args) {
        DatagramSocket serverSocket = null;

        try {
            // Create a server socket bound to port 9876
            serverSocket = new DatagramSocket(9876);

            InetAddress clientAddress = InetAddress.getByName("localhost"); // Client
address
            Scanner scanner = new Scanner(System.in);

            System.out.println("Server is ready to send messages to the client.");

            while (true) {
                System.out.print("Enter message: ");
                String message = scanner.nextLine();
                byte[] sendData = message.getBytes();

                // Create a datagram packet to send data to the client on port 9875
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, 9875);

                // Send the packet
                serverSocket.send(sendPacket);

                // Exit the loop if the user types "exit"
                if (message.equalsIgnoreCase("exit")) {
                    System.out.println("Server exiting...");
                    break;
                }
            }
            scanner.close();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (serverSocket != null && !serverSocket.isClosed()) {
```

```
            serverSocket.close();
        }
    }
  }
}
```

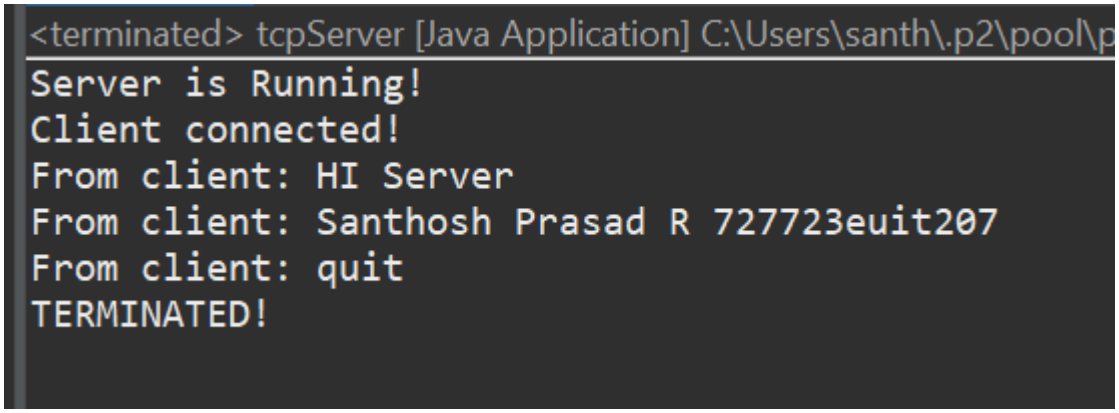
CLIENT PROGRAM:

```
import java.net.*;

public class udpClient {
   public static void main(String[] args) {
      DatagramSocket clientSocket = null;

      try {
         // Create a socket to listen on port 9875
         clientSocket = new DatagramSocket(9875);
         byte[] receiveData = new byte[1024]; // Buffer for receiving data

         System.out.println("Client is ready to receive messages.");

         while (true) {
            // Create a datagram packet to receive data
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);

            // Receive data from the server
            clientSocket.receive(receivePacket);

            // Convert the byte array into a string
            String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());

            // Display the message
            System.out.println("Server says: " + message);

            // Exit if the message is "exit"
            if (message.equalsIgnoreCase("exit")) {
               System.out.println("Client exiting...");
               break;
            }
         }

      } catch (Exception e) {
         e.printStackTrace();
```

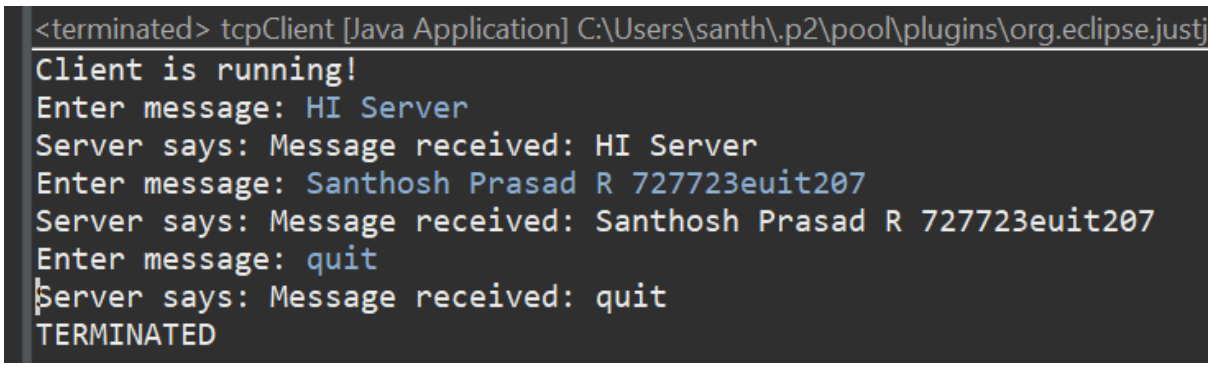
```
        } finally {
            if (clientSocket != null && !clientSocket.isClosed()) {
                clientSocket.close();
            }
        }
    }
}
```

SCREENSHOT OF OUTPUT:

Server :



Client :



RESULT:

Thus, the concept of Socket Programming is studied and implemented using Java Programming and is successfully executed in Eclipse IDE and verified.