Assignment No - 1a

Server and Cilent IMPLEMENTATION

CODE:-

QuoteServer.java:-

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



public class QuoteServer {
    private DatagramSocket socket;
    private List<String> listQuotes = new ArrayList<String>();
    private Random random;

    public QuoteServer(int port) throws SocketException {
        socket = new DatagramSocket(port);
        random = new Random();
    }

    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Syntax: QuoteServer <file> <port>");
            return;
        }

        String quoteFile = args[0];
        int port = Integer.parseInt(args[1]);

        try {
            QuoteServer server = new QuoteServer(port);
            server.loadQuotesFromFile(quoteFile);
            server.service();
        } catch (SocketException ex) {
            System.out.println("Socket error: " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("I/O error: " + ex.getMessage());
```

```
        }
    }

    private void service() throws IOException {
        while (true) {
            DatagramPacket request = new DatagramPacket(new byte[1], 1);
            socket.receive(request);

            String quote = getRandomQuote();
            byte[] buffer = quote.getBytes();

            InetAddress clientAddress = request.getAddress();
            int clientPort = request.getPort();

            DatagramPacket response = new DatagramPacket(buffer, buffer.length,
clientAddress, clientPort);
            socket.send(response);
        }
    }

    private void loadQuotesFromFile(String quoteFile) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader(quoteFile));
        String aQuote;

        while ((aQuote = reader.readLine()) != null) {
            listQuotes.add(aQuote);
        }

        reader.close();
    }

    private String getRandomQuote() {
        int randomIndex = random.nextInt(listQuotes.size());
        String randomQuote = listQuotes.get(randomIndex);
        return randomQuote;
    }
}
```

QuoteCilent:-

```
import java.io.*;
import java.net.*;


public class QuoteClient {

   public static void main(String[] args) {
      if (args.length < 2) {
         System.out.println("Syntax: QuoteClient <hostname> <port>");
         return;
      }

      String hostname = args[0];
      int port = Integer.parseInt(args[1]);

      try {
         InetAddress address = InetAddress.getByName(hostname);
         DatagramSocket socket = new DatagramSocket();

         while (true) {

            DatagramPacket request = new DatagramPacket(new byte[1], 1, address, port);
            socket.send(request);

            byte[] buffer = new byte[512];
            DatagramPacket response = new DatagramPacket(buffer, buffer.length);
            socket.receive(response);

            String quote = new String(buffer, 0, response.getLength());

            System.out.println(quote);
            System.out.println();

            Thread.sleep(3000);
         }

      } catch (SocketTimeoutException ex) {
         System.out.println("Timeout error: " + ex.getMessage());
```

```
            ex.printStackTrace();
        } catch (IOException ex) {
            System.out.println("Client error: " + ex.getMessage());
            ex.printStackTrace();
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}
```

TCPServer.java:-

```
// A Java program for a Server
import java.net.*;
import java.io.*;

public class TCPServer
{
        //initialize socket and input stream
        private Socket          socket = null;
        private ServerSocket server = null;
        private DataInputStream in     = null;

        // constructor with port
        public Server(int port)
        {
                // starts server and waits for a connection
                try
                {
                        server = new ServerSocket(port);
                        System.out.println("Server started");

                        System.out.println("Waiting for a client ...");

                        socket = server.accept();
                        System.out.println("Client accepted");

                        // takes input from the client socket
                        in = new DataInputStream(
                                new BufferedInputStream(socket.getInputStream()));
```

```
                String line = "";

                // reads message from client until "Over" is sent
                while (!line.equals("Over"))
                {
                        try
                        {
                                line = in.readUTF();
                                System.out.println(line);

                        }
                        catch(IOException i)
                        {
                                System.out.println(i);
                        }
                }
                System.out.println("Closing connection");

                // close connection
                socket.close();
                in.close();
        }
        catch(IOException i)
        {
                System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

TCPCilent.java:-

```
// A Java program for a Client
import java.net.*;
```

```
import java.io.*;

public class TCPClient
{
    // initialize socket and input output streams
    private Socket socket           = null;
    private DataInputStream  input   = null;
    private DataOutputStream out     = null;

    // constructor to put ip address and port
    public Client(String address, int port)
    {
        // establish a connection
        try
        {
            socket = new Socket(address, port);
            System.out.println("Connected");

            // takes input from terminal
            input  = new DataInputStream(System.in);

            // sends output to the socket
            out    = new DataOutputStream(socket.getOutputStream());
        }
        catch(UnknownHostException u)
        {
            System.out.println(u);
        }
        catch(IOException i)
        {
            System.out.println(i);
        }

        // string to read message from input
        String line = "";

        // keep reading until "Over" is input
        while (!line.equals("Over"))
```

```
        {
            try
            {
                line = input.readLine();
                out.writeUTF(line);
            }
            catch(IOException i)
            {
                System.out.println(i);
            }
        }

        // close the connection
        try
        {
            input.close();
            out.close();
            socket.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```