1. WAP URL Socket Connection

```
public class egthree {
   public static void main(String[] args) {
      try {
         Socket socket = new Socket("www.prime.edu.np",80);
         System.out.println("Connected to " +socket.getInetAddress()+" on port
"+socket.getPort()+" from port "+socket.getLocalPort()+" of "+socket.getLocalAddress());
      }
      catch (UnknownHostException ex){
         System.err.println("I cant find the host");
      }
      catch (SocketException ex){
         System.err.println("Could not connect to host");
      }
      catch (IOException ex){
         System.err.println(ex);
      }
   }
}
```

2. WAp for Socket Creation

```
import java.io.*;

import java.net.*;

public class EchoClient {

   public static void main(String[] args) {

      try {

         Socket socket = new Socket("localhost",443);

         System.out.println("Connected");

         BufferedReader in_socket = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

         BufferedReader in_user = new BufferedReader(new InputStreamReader(System.in));

         PrintWriter out_socket = new PrintWriter(socket.getOutputStream(),true);

         String message;

         while (true){

            message = in_user.readLine();

            if (message.endsWith("."))break;
```

```
            out_socket.println(message);
            System.out.println(in_socket.readLine());
        }
        in_socket.close();
        out_socket.close();
        socket.close();;
      }catch (UnknownHostException e){
        System.err.println("Unknown");
      }catch (IOException e){
        System.err.println("Socket cant be created");
      }
    }
}
```

2.WAP for the higher Port

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

public class HighPortScanner {
   public static void main(String[] args) {
      try {
         InetAddress address = InetAddress.getByName("Localhost");
         for (int i=1024; i<65536;i++){
            try {
               Socket socket = new Socket(address, i);
               System.out.println("There is a server on port "+i+"of Localhost.");
            }catch (IOException e){
//                System.err.println("There is a server off port "+i+" of localhost");
            }
         }
      }catch (UnknownHostException e1){
         System.out.println("unknwon Host.");
      }
   }
}
```

3.WAP low port scanner

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

public class LowPortScanner {

   public static void main(String[] args) {
      for (int i=1;i<1024;i++){
```

```
        try{
           Socket socket = new Socket("localhost",80);
           System.out.println("There is server on port "+i+" of localhost.");
        }
        catch (UnknownHostException e){
           System.out.println(e);
        }
        catch (IOException e){
           System.err.println("there is server off port "+i+" of localhost");
        }
      }
   }

}
```

4. Client Socket Connection

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

public class client {

        public static void main (String [] args) {
                try {
                        System.out.println("Waiting for connection...");
```

```
                Socket clientSocket=new Socket("localhost", 4567);        //client needs server's
ip/address and port to connect
                System.out.println("Connected to server...");


                BufferedReader br=new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));      //read input data
                PrintWriter pw=new PrintWriter(clientSocket.getOutputStream(), true);
        //write/send output data
                Scanner scanner=new Scanner(System.in);         //take input from console


                while(true) {                                                   //loop
continues until user enter 'quit' in console
                        System.out.println("Enter text: ");
                        String inputLine=scanner.nextLine();      //take input from console
                        if(inputLine.equalsIgnoreCase("quit")) {  //to end chat/connection
                                break;
                        }
                        pw.println(inputLine);                                  //send typed
message in console to server


                        String response=br.readLine();                  //server echo the
message sent by client, so getting response
                        System.out.println("Server: " + response);          //printing server's
response
                }


        } catch (UnknownHostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }



        }
}
```

5.WAP Server Socket Connection

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class server {

    public static void main(String [] args) {

        try {

            ServerSocket serverSocket=new ServerSocket(4567);      //server listening at port 4567
            System.out.println("Waiting for connection...");
            Socket clientSocket=serverSocket.accept();                  //serverSocket will block this call until it receives a request from a client
            System.out.println("Connection established..."); //when request is received the accept method will return a socket class instance.
```

```
                    BufferedReader br=new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter pw=new PrintWriter(clientSocket.getOutputStream(),true);


                    String inputLine;
                    while((inputLine=br.readLine())!=null) {
                        System.out.println("\nServer:" + inputLine);
                        pw.println(inputLine);
                    }



                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }




        }
}
```

6. Chat Server

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class SimpleChatServer {

    private static final int PORT = 9000;
    private static List<ClientHandler> clients = new ArrayList<>();
    private static int nextUserId = 1;

    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server is running on port " + PORT);

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected");

                ClientHandler clientHandler = new ClientHandler(clientSocket, "User" + nextUserId);
                nextUserId++;
```

```
            clients.add(clientHandler);
            new Thread(clientHandler).start();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

static class ClientHandler implements Runnable {
    private Socket clientSocket;
    private PrintWriter out;
    private BufferedReader in;
    private String username;

    public ClientHandler(Socket socket, String username) {
        this.clientSocket = socket;
        this.username = username;
        try {
            out = new PrintWriter(clientSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        try {
            // Notify client of their username
```

```
            out.println("You are connected as: " + username);

            String message;
            while ((message = in.readLine()) != null) {
                System.out.println(username + ": " + message);
                broadcastMessage(username + ": " + message, this);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                in.close();
                out.close();
                clientSocket.close();
                clients.remove(this);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private void broadcastMessage(String message, ClientHandler sender) {
        for (ClientHandler client : clients) {
            if (client != sender) {
                client.sendMessage(message);
            }
        }
    }
```

```
    private void sendMessage(String message) {
        out.println(message);
    }
  }
}
```

8. SimpleNetworkClientGUI

```
public class SimpleNetworkClientGUI extends JFrame {

  private JTextField serverAddressField;
  private JTextField portField;
  private JTextArea chatArea;
  private JTextArea messageArea;
  private JButton sendButton;

  private Socket socket;
  private PrintWriter out;
  private BufferedReader in;

  public SimpleNetworkClientGUI() {
    setTitle("Simple Network Client");
    setSize(600, 400);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    // Panel for connection details
    JPanel connectionPanel = new JPanel(new FlowLayout());
    serverAddressField = new JTextField("localhost", 15);
    portField = new JTextField("9000", 5);
```

```
JButton connectButton = new JButton("Connect");
connectButton.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      connectToServer();
   }
});

connectionPanel.add(new JLabel("Server Address:"));
connectionPanel.add(serverAddressField);
connectionPanel.add(new JLabel("Port:"));
connectionPanel.add(portField);
connectionPanel.add(connectButton);

// Panel for chat area
JPanel chatPanel = new JPanel(new BorderLayout());
chatArea = new JTextArea(15, 50);
chatArea.setEditable(false);
JScrollPane chatScrollPane = new JScrollPane(chatArea);
chatPanel.add(chatScrollPane, BorderLayout.CENTER);

// Panel for message area and send button
JPanel messagePanel = new JPanel(new BorderLayout());
messageArea = new JTextArea(5, 50);
messageArea.setLineWrap(true);
messageArea.setWrapStyleWord(true);
JScrollPane messageScrollPane = new JScrollPane(messageArea);
messagePanel.add(messageScrollPane, BorderLayout.CENTER);
```

```
    sendButton = new JButton("Send");
    sendButton.setPreferredSize(new Dimension(80, 30));
    sendButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            sendMessage();
        }
    });
    messagePanel.add(sendButton, BorderLayout.EAST);

    // Add panels to the main frame
    add(connectionPanel, BorderLayout.NORTH);
    add(chatPanel, BorderLayout.CENTER);
    add(messagePanel, BorderLayout.SOUTH);
}

private void connectToServer() {
    String serverAddress = serverAddressField.getText().trim();
    int port = Integer.parseInt(portField.getText().trim());

    try {
        socket = new Socket(serverAddress, port);
        out = new PrintWriter(socket.getOutputStream(), true);
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        appendToChatArea("Connected to server at " + serverAddress + ":" + port);

        // Start a thread to continuously read messages from server
        new Thread(new Runnable() {
            @Override
```

```
            public void run() {
                try {
                    String message;
                    while ((message = in.readLine()) != null) {
                        appendToChatArea(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();

    } catch (IOException e) {
        appendToChatArea("Error connecting to server: " + e.getMessage());
        e.printStackTrace();
    }
}

private void sendMessage() {
    if (out != null) {
        String message = messageArea.getText().trim();
        if (!message.isEmpty()) {
            out.println("You: " + message); // Prefix with "You:"
            appendToChatArea("You: " + message); // Append to chatArea with prefix
            messageArea.setText(""); // Clear message area
        }
    }
}
```

```
    private void appendToChatArea(String message) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                chatArea.append(message + "\n");
                chatArea.setCaretPosition(chatArea.getDocument().getLength());
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                SimpleNetworkClientGUI clientGUI = new SimpleNetworkClientGUI();
                clientGUI.setVisible(true);
            }
        });
    }
}
```

1. WAP URL Socket Connection
2. WAp for Socket Creation
3. WAP for the higher Port
4. WAP low port scanner
5. Client Socket Connection
6. WAP Server Socket Connection
7. Chat Server
8. Simple Network Client GUI

Extra Questions

- Write a program to display the socket information address,port,localaddress, localportaddress, port, local address, local portaddress,port,localaddress,localport.

- Write a server side program for daytime service using socket.