Computer Networks Lab Exercise 1

Code for Development of a Chat Application based on Socket Programming using UDP.

UDPServer.java

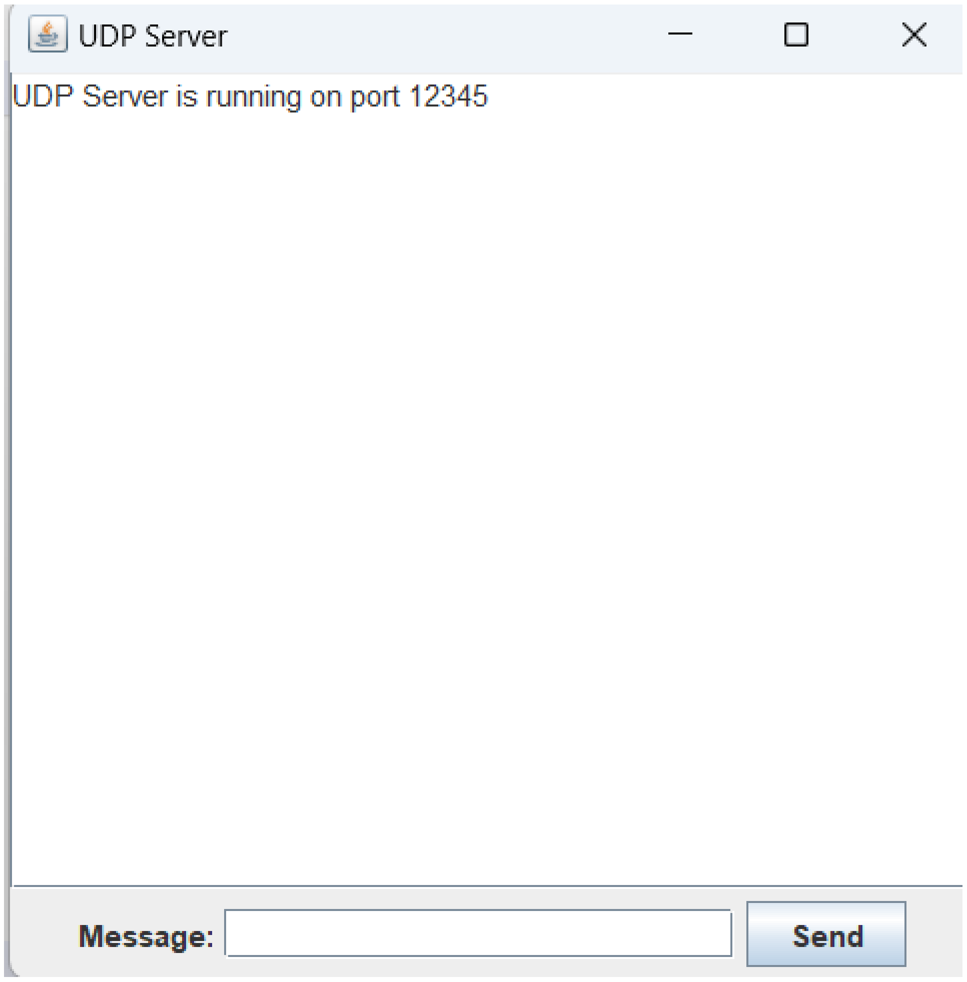
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class UDPServer {
    private DatagramSocket serverSocket;
    private JTextArea logArea;
    private JTextField messageField;
    private JButton sendButton;
    private InetAddress clientAddress;
    private int clientPort;

    private static final int SERVER_PORT = 12345; // Define the server
port here

    public UDPServer() {
        JFrame frame = new JFrame("UDP Server");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);

        logArea = new JTextArea(10, 30);
        logArea.setEditable(false);
        frame.add(new JScrollPane(logArea), BorderLayout.CENTER);

        JPanel controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        JLabel messageLabel = new JLabel("Message:");
        messageField = new JTextField(20);
        sendButton = new JButton("Send");
        sendButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                sendMessage();
            }
        });

        controlPanel.add(messageLabel);
        controlPanel.add(messageField);
        controlPanel.add(sendButton);
        frame.add(controlPanel, BorderLayout.SOUTH);

        frame.setVisible(true);

        try {
            serverSocket = new DatagramSocket(SERVER_PORT);
            appendToLogArea("UDP Server is running on port " +
SERVER_PORT);
            startListening();
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }

    private void startListening() {
        new Thread(() -> {
            try {
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);

                while (true) {
                    serverSocket.receive(receivePacket);

                    String message = new String(receivePacket.getData(),
0, receivePacket.getLength());
```

```
                    clientAddress = receivePacket.getAddress();
                    clientPort = receivePacket.getPort();

                    appendToLogArea("Connected to Client: " +
clientAddress.getHostAddress() + ":" + clientPort);
                    appendToLogArea("Received from " +
clientAddress.getHostAddress() + ":" + clientPort + ": " + message);
                }
            } catch (IOException e) {
                appendToLogArea("Client Disconnected: " +
clientAddress.getHostAddress() + ":" + clientPort);
                e.printStackTrace();
            }
        }).start();
    }

    private void sendMessage() {
        try {
            String message = messageField.getText();
            byte[] sendData = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
            serverSocket.send(sendPacket);

            appendToLogArea("Server: " + message);
            messageField.setText("");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void appendToLogArea(String message) {
        SwingUtilities.invokeLater(() -> {
            logArea.append(message + "\n");
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new UDPServer();
        });
    }
}
```

UDPClient.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class UDPClient {
    private DatagramSocket clientSocket;
    private JTextArea chatArea;
    private JTextField messageField;
    private JButton sendButton;

    private static String SERVER_IP = "localhost"; // Define the server
IP here
    private static final int SERVER_PORT = 12345;

    public UDPClient() {
        JFrame frame = new JFrame("UDP Chat Client");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        chatArea = new JTextArea(10, 30);
        chatArea.setEditable(false);
        JScrollPane chatScrollPane = new JScrollPane(chatArea);
        frame.add(chatScrollPane, BorderLayout.CENTER);

        JPanel controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());

        messageField = new JTextField(20);
        sendButton = new JButton("Send");
        sendButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                sendMessage();
            }
        });
        controlPanel.add(new JLabel("Message: "));
        controlPanel.add(messageField);
        controlPanel.add(sendButton);

        frame.add(controlPanel, BorderLayout.SOUTH);

        frame.setVisible(true);

        try {
            clientSocket = new DatagramSocket();
            updateServerIP();
            startListening();
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }

    private void updateServerIP() {
        String input = JOptionPane.showInputDialog("Enter Server IP
Address:");
        if (input != null && !input.isEmpty()) {
            SERVER_IP = input;
        } else {
            System.exit(0);
        }
    }

    private void startListening() {
        new Thread(() -> {
            try {
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);

                while (true) {
                    clientSocket.receive(receivePacket);
                    String receivedMessage = new
String(receivePacket.getData(), 0, receivePacket.getLength());

                    if (receivedMessage.startsWith("Server: ")) {
                        appendToChatArea(receivedMessage);
                    } else {
                        appendToChatArea("Server: " + receivedMessage);
                    }
                }
            } catch (IOException e) {
                appendToChatArea("Server Unavailable: " + SERVER_IP);
                e.printStackTrace();
            }
        }).start();
    }

    private void sendMessage() {
        try {
            String message = messageField.getText();
            byte[] sendData = message.getBytes();
            InetAddress serverAddress =
InetAddress.getByName(SERVER_IP);
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, SERVER_PORT);
            clientSocket.send(sendPacket);
            appendToChatArea("You: " + message);
            messageField.setText("");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void appendToChatArea(String message) {
        SwingUtilities.invokeLater(() -> {
            chatArea.append(message + "\n");
        });
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new UDPClient();
        });
    }
}
```

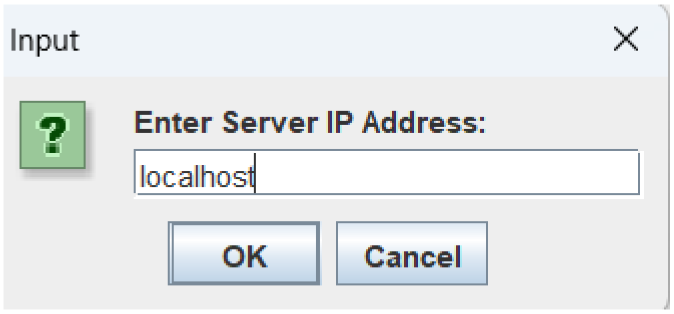
Output



UDP Server

```
UDP Server is running on port 12345
```

Message: Send

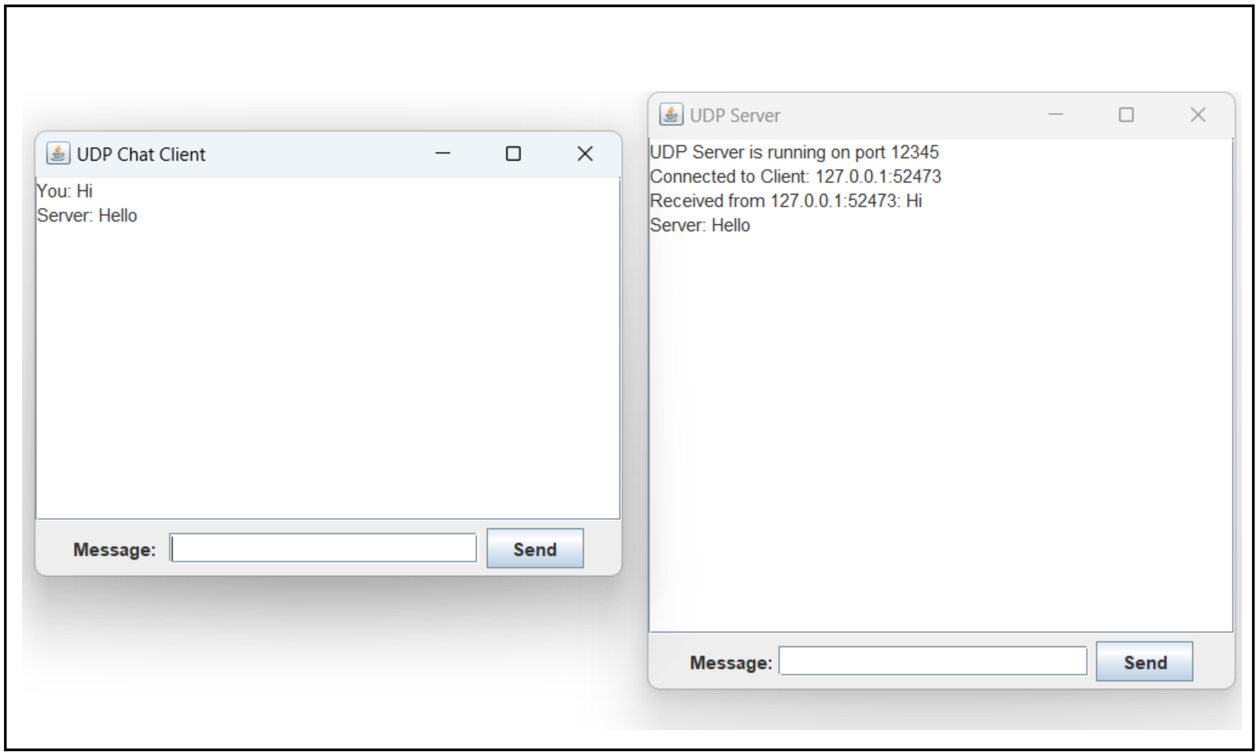


Input

Enter Server IP Address:

localhost

OK Cancel



```
You: Hi
Server: Hello
```

```
UDP Server is running on port 12345
Connected to Client: 127.0.0.1:52473
Received from 127.0.0.1:52473: Hi
Server: Hello
```