

# CL-IX Assignment 1B

Roll No. 43104  
Ajinkya Tikhe

## Assignment 1B - UDP Java Sockets

**Aim:** To develop any distributed application through implementing client-server communication programs based on UDP Java Sockets.

**Problem Statment:** String Manipulation: Design a distributed application that consists of client-server communication using TCP, UDP & RMI techniques in Java. Multiple clients can simultaneously connect to the server and multiple clients submit two strings to the server and the server returns the concatenation of the given strings.

File 1:

### Server.java

```
// Java program to illustrate Server side
// Implementation using DatagramSocket
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

public class Server
{
    public static void main(String[] args) throws IOException
    {
        // Step 1 : Create a socket to listen at port 1234
        DatagramSocket ds = new DatagramSocket(1234);
        byte[] receive = new byte[65535];

        StringBuilder answer = new StringBuilder();
```

```

DatagramPacket DpReceive = null;

for(int i=0; i<2; i++)
{

    // Step 2 : create a DatagramPacket to receive the data.
    DpReceive = new DatagramPacket(receive, receive.length);

    // Step 3 : revieve the data in byte buffer.
    ds.receive(DpReceive);

    answer = answer.append(data(receive));

    // Clear the buffer after every message.
    receive = new byte[65535];
}
System.out.println("Concatenation of Input Strings : " + answer);
}

// A utility method to convert the byte array
// data into a string representation.
public static StringBuilder data(byte[] a)
{
    if (a == null)
        return null;
    StringBuilder ret = new StringBuilder();
    int i = 0;
    while (a[i] != 0)
    {
        ret.append((char) a[i]);
        i++;
    }
    return ret;
}
}

```

---

File 2:

### **Client.java**

```
// Java program to illustrate Client side
// Implementation using DatagramSocket
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class Client
{
    public static void main(String args[]) throws IOException
    {
        Scanner sc = new Scanner(System.in);

        // Step 1: Create the socket object for
        // carrying the data.
        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;

        System.out.println("Enter input - ");
        for(int i=0; i<2; i++)
        {

            System.out.print("String " + (i+1) + " : ");

            String inp = sc.nextLine();

            // convert the String input into the byte array.
            buf = inp.getBytes();

            // Step 2 : Create the datagramPacket for sending
```

```

        // the data.
        DatagramPacket DpSend =
            new DatagramPacket(buf, buf.length, ip, 1234);

        // Step 3 : invoke the send call to actually send
        // the data.
        ds.send(DpSend);
    }
}
}

```

---

## Client Console:

The screenshot shows the Eclipse IDE with the 'Client.java' file open. The code is a Java program for a client side of a UDP socket communication. It imports necessary classes like IOException, DatagramSocket, DatagramPacket, InetAddress, and Scanner. The main method creates a Scanner for input, prompts the user to 'Enter input - ', and then reads two lines of input: 'Hi' and 'Ajinkya'.

```

1 // Java program to illustrate Client side
2 // Implementation using DatagramSocket
3 import java.io.IOException;
4 import java.net.DatagramPacket;
5 import java.net.DatagramSocket;
6 import java.net.InetAddress;
7 import java.util.Scanner;
8
9 public class Client
10 {
11     public static void main(String args[]) throws IOException
12     {
13         Scanner sc = new Scanner(System.in);
14
15         // Step 1: Create the socket object for
16         // carrying the data.
17         DatagramSocket ds = new DatagramSocket();
18
19         InetAddress ip = InetAddress.getLocalHost();
20         byte buf[] = null;
21
22         System.out.println("Enter input - ");
23         for(int i=0; i<2; i++)
24         {

```

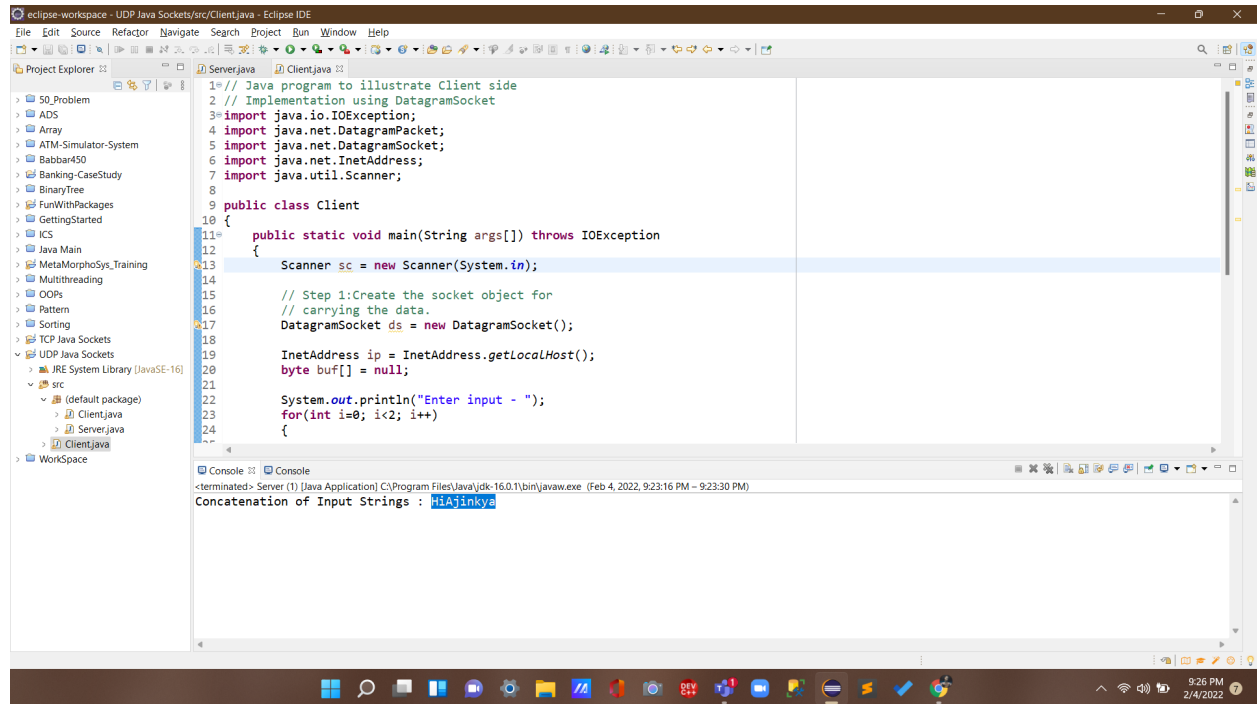
The console output shows the execution of the program:

```

<terminated> Client (1) [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (Feb 4, 2022, 9:23:20 PM - 9:23:30 PM)
Enter input -
String 1 : Hi
String 2 : Ajinkya

```

## Server Console:



The screenshot shows the Eclipse IDE interface. The Project Explorer on the left lists various projects, including 'UDP Java Sockets'. The main editor displays the 'Client.java' file, which contains a Java program to illustrate the client side of a UDP socket implementation. The code includes imports for `IOException`, `DatagramPacket`, `DatagramSocket`, `InetAddress`, and `Scanner`. The `main` method creates a `Scanner` for input, creates a `DatagramSocket`, gets the local host address, and prints a prompt 'Enter input - '. It then enters a loop to read input strings and concatenate them. The Console window at the bottom shows the output: 'Concatenation of Input Strings : HiAjinkya'.

```
1 // Java program to illustrate Client side
2 // Implementation using DatagramSocket
3 import java.io.IOException;
4 import java.net.DatagramPacket;
5 import java.net.DatagramSocket;
6 import java.net.InetAddress;
7 import java.util.Scanner;
8
9 public class Client
10 {
11     public static void main(String args[]) throws IOException
12     {
13         Scanner sc = new Scanner(System.in);
14
15         // Step 1: Create the socket object for
16         // carrying the data.
17         DatagramSocket ds = new DatagramSocket();
18
19         InetAddress ip = InetAddress.getLocalHost();
20         byte buf[] = null;
21
22         System.out.println("Enter input - ");
23         for(int i=0; i<2; i++)
24         {
```

Console Output:  
Concatenation of Input Strings : HiAjinkya

---

**Conclusion:** Developed distributed application through implementing client-server communication programs based on UDP Java Sockets.

---