Computer Networking Coursework Assignment
Birkbeck, University of London
Academic Year 2016/17

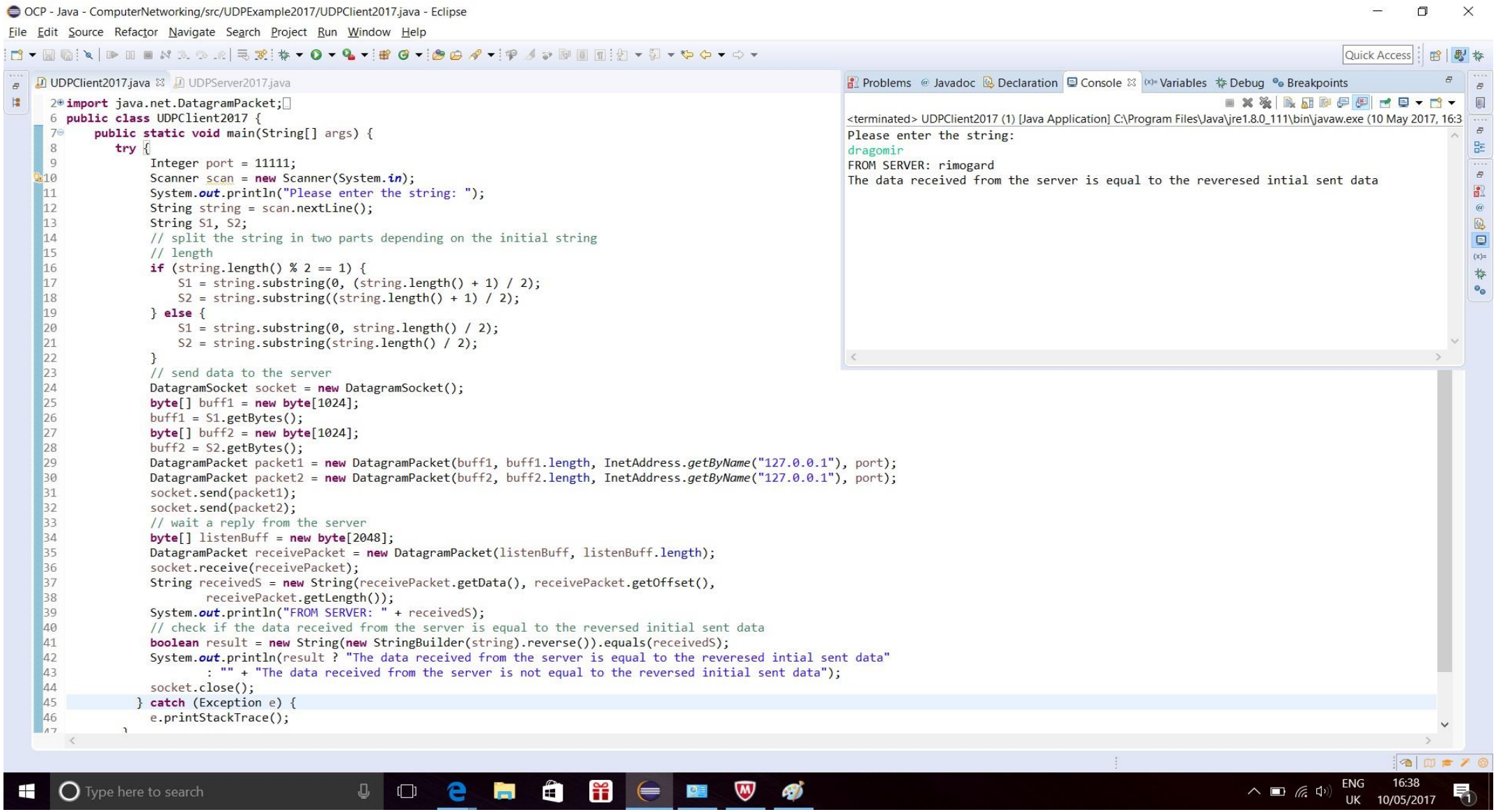
The below pair screenshots represent the code in Java for two programs(client and server) using the User Datagram Protocol. UDP is used to implement a connectionless packet delivery, therefore the order and the delivery of the packets is not guaranteed.

Client program - prompts the user to input a String, which is then split into two strings based on the length of the initial input string. Then the client sends to the server the two new strings in two separate UDP packets - DatagramPacket, by creating a DatagramSocket as the sending or receiving point for a packet delivery service. A third DatagramPacket is created for receiving the reply from the server, which is then printed on the console, and the result of comparing the reversed initial sent message with the message received from the server.

Server program – constantly waits for UDP messages. A DatagramSocket is created as the sending or receiving point for packet delivery service. Two DatagramPackets are created to receive the two distinct messages from the client. Then the messages received are concatenated, and the reversed result of concatenation is sent as an UDP message to the client.

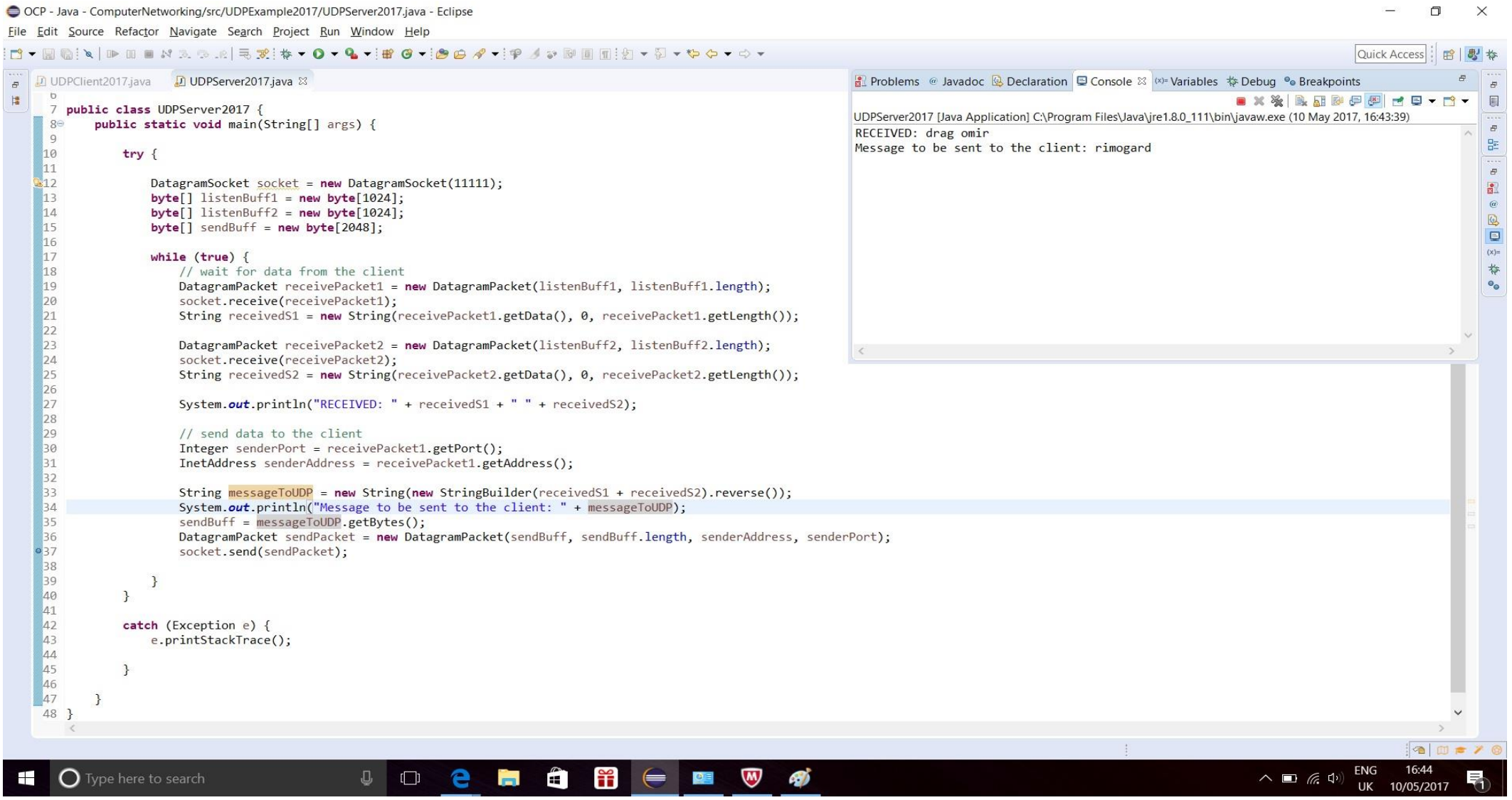
Please see below the two examples showing in the console window the input and output for the server and the client.

EXAMPLE No.1 CLIENT



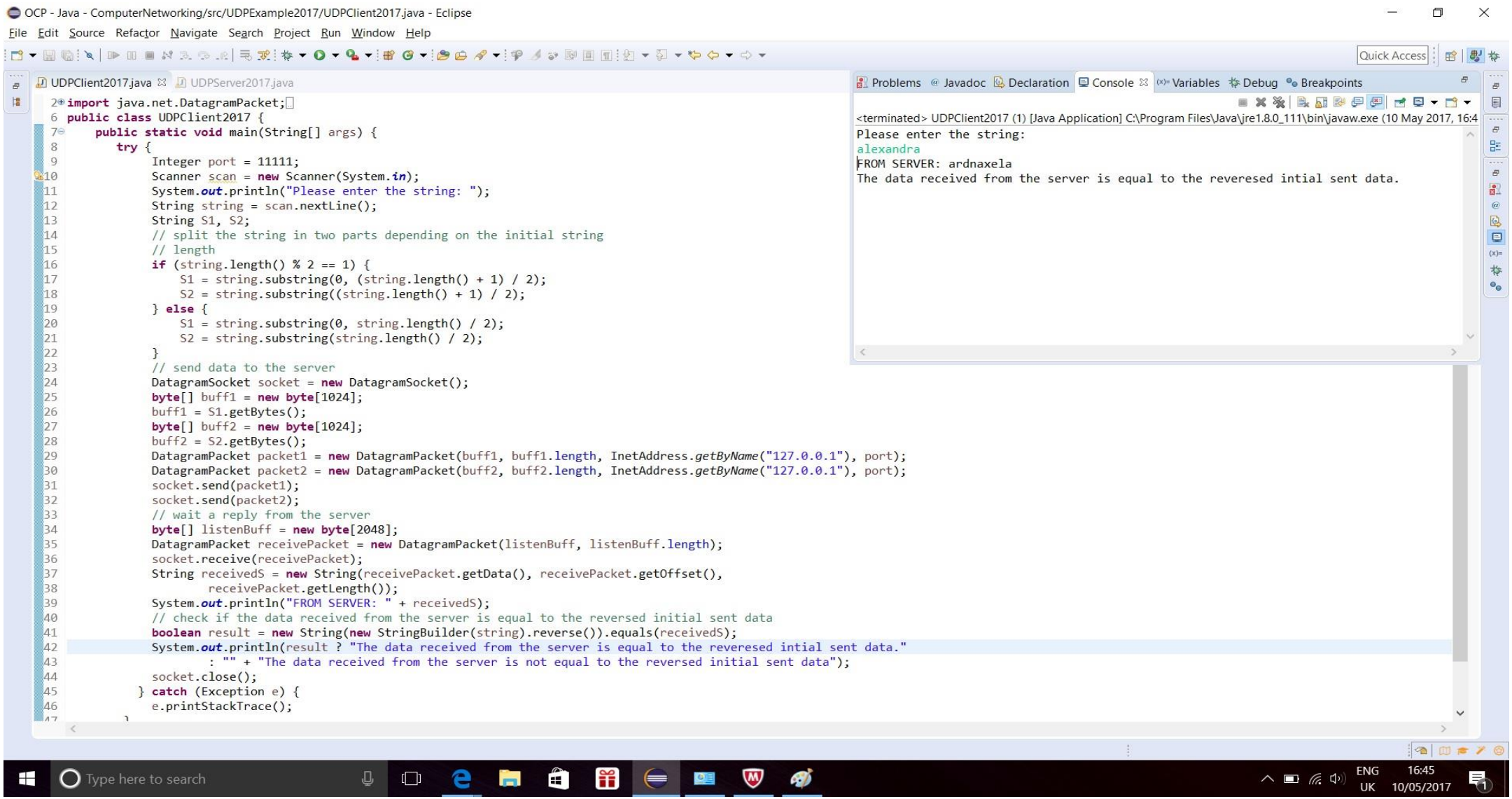

```
import java.net.DatagramPacket;
public class UDPClient2017 {
    public static void main(String[] args) {
        try {
            Integer port = 11111;
            Scanner scan = new Scanner(System.in);
            System.out.println("Please enter the string: ");
            String string = scan.nextLine();
            String S1, S2;
            // split the string in two parts depending on the initial string
            // length
            if (string.length() % 2 == 1) {
                S1 = string.substring(0, (string.length() + 1) / 2);
                S2 = string.substring((string.length() + 1) / 2);
            } else {
                S1 = string.substring(0, string.length() / 2);
                S2 = string.substring(string.length() / 2);
            }
            // send data to the server
            DatagramSocket socket = new DatagramSocket();
            byte[] buff1 = new byte[1024];
            buff1 = S1.getBytes();
            byte[] buff2 = new byte[1024];
            buff2 = S2.getBytes();
            DatagramPacket packet1 = new DatagramPacket(buff1, buff1.length, InetAddress.getByName("127.0.0.1"), port);
            DatagramPacket packet2 = new DatagramPacket(buff2, buff2.length, InetAddress.getByName("127.0.0.1"), port);
            socket.send(packet1);
            socket.send(packet2);
            // wait a reply from the server
            byte[] listenBuff = new byte[2048];
            DatagramPacket receivePacket = new DatagramPacket(listenBuff, listenBuff.length);
            socket.receive(receivePacket);
            String receivedS = new String(receivePacket.getData(), receivePacket.getOffset(),
                    receivePacket.getLength());
            System.out.println("FROM SERVER: " + receivedS);
            // check if the data received from the server is equal to the reversed initial sent data
            boolean result = new String(new StringBuilder(string).reverse()).equals(receivedS);
            System.out.println(result ? "The data received from the server is equal to the reveresed intial sent data"
                    : "" + "The data received from the server is not equal to the reversed initial sent data");
            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

EXAMPLE No.1 SERVER



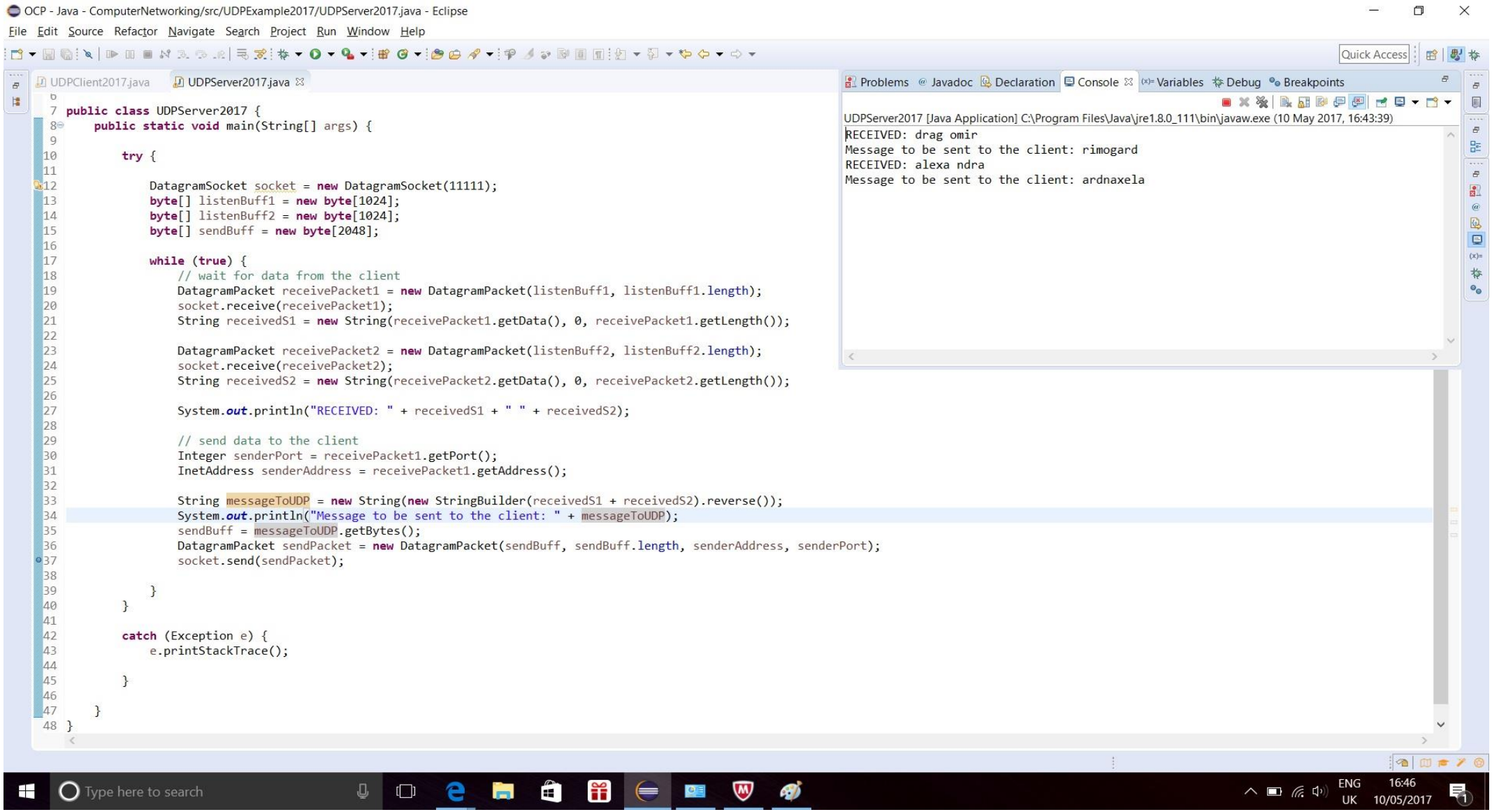

```
public class UDPServer2017 {
    public static void main(String[] args) {

        try {

            DatagramSocket socket = new DatagramSocket(11111);
            byte[] listenBuff1 = new byte[1024];
            byte[] listenBuff2 = new byte[1024];
            byte[] sendBuff = new byte[2048];

            while (true) {
                // wait for data from the client
                DatagramPacket receivePacket1 = new DatagramPacket(listenBuff1, listenBuff1.length);
                socket.receive(receivePacket1);
                String receivedS1 = new String(receivePacket1.getData(), 0, receivePacket1.getLength());

                DatagramPacket receivePacket2 = new DatagramPacket(listenBuff2, listenBuff2.length);
                socket.receive(receivePacket2);
                String receivedS2 = new String(receivePacket2.getData(), 0, receivePacket2.getLength());

                System.out.println("RECEIVED: " + receivedS1 + " " + receivedS2);

                // send data to the client
                Integer senderPort = receivePacket1.getPort();
                InetAddress senderAddress = receivePacket1.getAddress();

                String messageToUDP = new String(new StringBuilder(receivedS1 + receivedS2).reverse());
                System.out.println("Message to be sent to the client: " + messageToUDP);
                sendBuff = messageToUDP.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendBuff, sendBuff.length, senderAddress, senderPort);
                socket.send(sendPacket);

            }
        }

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

        }

    }
}
```

EXEMPLE No.2 CLIENT




```
import java.net.DatagramPacket;
public class UDPClient2017 {
    public static void main(String[] args) {
       try {
            Integer port = 11111;
            Scanner scan = new Scanner(System.in);
            System.out.println("Please enter the string: ");
            String string = scan.nextLine();
            String S1, S2;
            // split the string in two parts depending on the initial string
            // length
            if (string.length() % 2 == 1) {
                S1 = string.substring(0, (string.length() + 1) / 2);
                S2 = string.substring((string.length() + 1) / 2);
            } else {
                S1 = string.substring(0, string.length() / 2);
                S2 = string.substring(string.length() / 2);
            }
            // send data to the server
            DatagramSocket socket = new DatagramSocket();
            byte[] buff1 = new byte[1024];
            buff1 = S1.getBytes();
            byte[] buff2 = new byte[1024];
            buff2 = S2.getBytes();
            DatagramPacket packet1 = new DatagramPacket(buff1, buff1.length, InetAddress.getByName("127.0.0.1"), port);
            DatagramPacket packet2 = new DatagramPacket(buff2, buff2.length, InetAddress.getByName("127.0.0.1"), port);
            socket.send(packet1);
            socket.send(packet2);
            // wait a reply from the server
            byte[] listenBuff = new byte[2048];
            DatagramPacket receivePacket = new DatagramPacket(listenBuff, listenBuff.length);
            socket.receive(receivePacket);
            String receivedS = new String(receivePacket.getData(), receivePacket.getOffset(),
                    receivePacket.getLength());
            System.out.println("FROM SERVER: " + receivedS);
            // check if the data received from the server is equal to the reversed initial sent data
            boolean result = new String(new StringBuilder(string).reverse()).equals(receivedS);
            System.out.println(result ? "The data received from the server is equal to the reveresed intial sent data."
                    : "" + "The data received from the server is not equal to the reversed initial sent data");
            socket.close();
          } catch (Exception e) {
            e.printStackTrace();
        }
```

EXAMPLE No.2 SERVER




```
public class UDPServer2017 {
    public static void main(String[] args) {

        try {

            DatagramSocket socket = new DatagramSocket(11111);
            byte[] listenBuff1 = new byte[1024];
            byte[] listenBuff2 = new byte[1024];
            byte[] sendBuff = new byte[2048];

            while (true) {
                // wait for data from the client
                DatagramPacket receivePacket1 = new DatagramPacket(listenBuff1, listenBuff1.length);
                socket.receive(receivePacket1);
                String receivedS1 = new String(receivePacket1.getData(), 0, receivePacket1.getLength());

                DatagramPacket receivePacket2 = new DatagramPacket(listenBuff2, listenBuff2.length);
                socket.receive(receivePacket2);
                String receivedS2 = new String(receivePacket2.getData(), 0, receivePacket2.getLength());

                System.out.println("RECEIVED: " + receivedS1 + " " + receivedS2);

                // send data to the client
                Integer senderPort = receivePacket1.getPort();
                InetAddress senderAddress = receivePacket1.getAddress();

                String messageToUDP = new String(new StringBuilder(receivedS1 + receivedS2).reverse());
                System.out.println("Message to be sent to the client: " + messageToUDP);
                sendBuff = messageToUDP.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendBuff, sendBuff.length, senderAddress, senderPort);
                socket.send(sendPacket);

            }
        }

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

        }

    }
}
```