Ex. No : 2a	Implementation of the Stop-and-Wait Protocol
Date :	

Aim :-

To implement and execute stop and wait protocol using java.

Algorithm :-

Sender :

Step 1 : Start the program.

Step 2 : Set Sequence = 0

Step 3 : Accept new packet and assign a sequence to it.

Step 4 : Send packet sequence with sequence number.

Step 5 : Set timer for recently sent packets.

Step 6 : If error free acknowledgment from receiver and Next Frame Expected -> sequence then sequence -> Next Frame Expected.

Step 7 : If time out then go to step 3.

Step 8 : Stop the program.

Receiver :

Step 1 : Start the program.

Step 2 : Next frame expected = 0, repeat step 3 forever.

Step 3 : If error free frame received and sequence -> Next Frame Expected, the pass packet to higher layer and Next Frame Expected -> Next Frame Expected +1 (modulo 2).

Step 4 : Stop the program.

Programs :-

Sender.java

```
import java.io.*;

import java.net.*;

public class Sender {

    public static void main(String[] args) {

        String serverAddress = "localhost";

        int serverPort = 9876;

        int timeout = 5000;  // Increased timeout
```

```
        try (DatagramSocket socket = new DatagramSocket()) {
            InetAddress serverInetAddress = InetAddress.getByName(serverAddress);
            String[] messages = {"Message 1", "Message 2", "Message 3"};
            for (String message : messages) {
                byte[] sendData = message.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverInetAddress, serverPort);
                socket.send(sendPacket);
                System.out.println("Sent: " + message);
                socket.setSoTimeout(timeout);
                boolean ackReceived = false;
                long startTime = System.currentTimeMillis();
                while (System.currentTimeMillis() - startTime < timeout) {
                    try {
                        byte[] receiveData = new byte[1024];
                        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                        socket.receive(receivePacket);
                        String ackMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                        if (ackMessage.equals("ACK")) {
                            ackReceived = true;
                            System.out.println("Received ACK for: " + message);
                            break;
                        }
                    } catch (SocketTimeoutException e) {
                        System.out.println("Timeout expired, resending: " + message);
                        socket.send(sendPacket);  // Resend if no ACK is received
                    }
                }
                if (!ackReceived) {
                    System.out.println("No ACK received after timeout, retrying...");
```

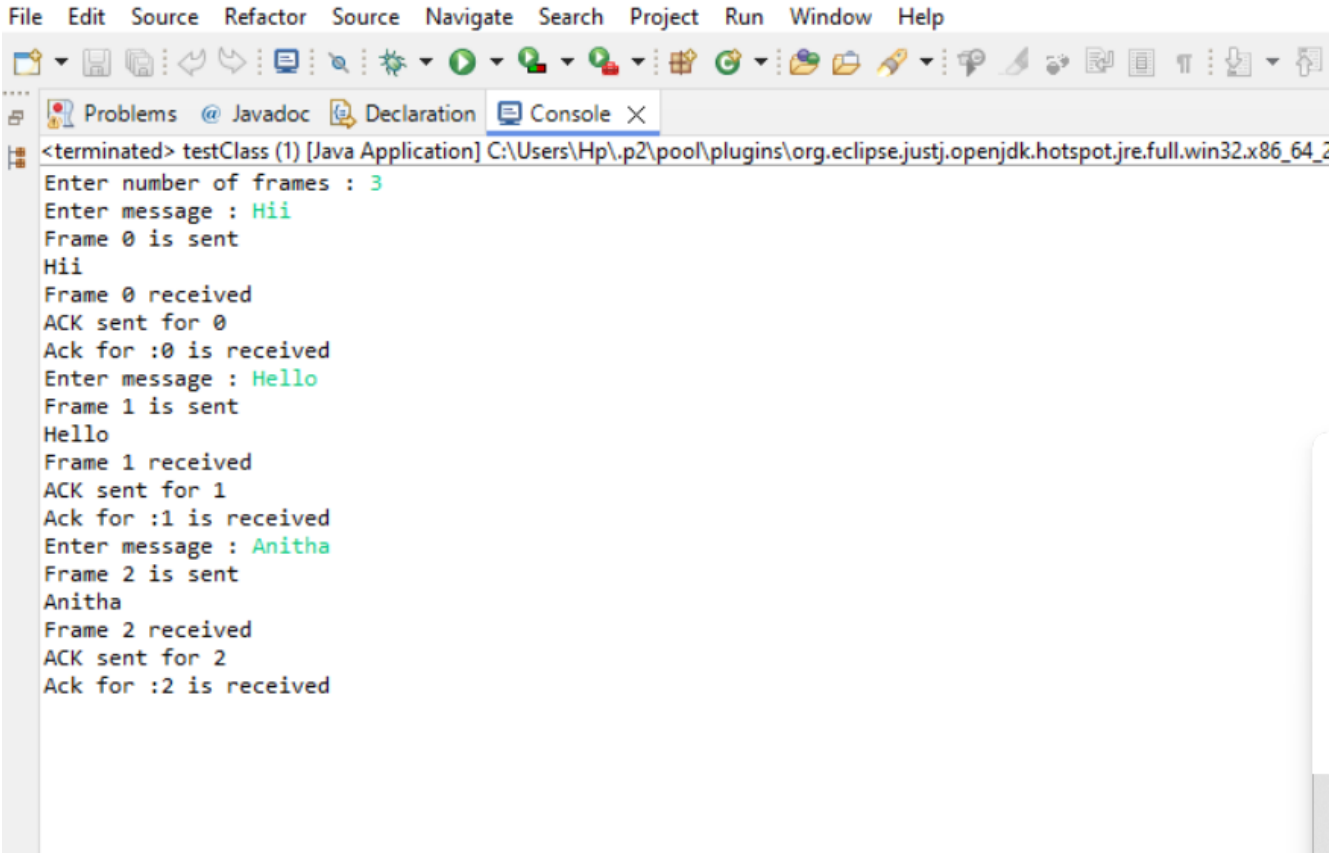
```
                socket.send(sendPacket);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Receiver.java

```
import java.io.*;
import java.net.*;
public class Receiver {
  public static void main(String[] args) {
    int serverPort = 9876;
    try (DatagramSocket socket = new DatagramSocket(serverPort)) {
        byte[] receiveData = new byte[1024];
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            socket.receive(receivePacket);
            String receivedMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Received: " + receivedMessage);
            Thread.sleep(500);
            InetAddress senderAddress = receivePacket.getAddress();
            int senderPort = receivePacket.getPort();
            String ackMessage = "ACK";
            byte[] sendData = ackMessage.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
senderAddress, senderPort);
            socket.send(sendPacket);
            System.out.println("Sent ACK for: " + receivedMessage);
```

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

Output :-



Result :-

Thus, the program for stop and wait protocol was implemented and executed successfully.