Lab Experiment 4

Simulate N-bit Sliding Window protocol, at data link layer.

Sender program:

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

class DataFrame implements Serializable {
    public char info;
    public int sequenceNumber;

    DataFrame() {
    }

    DataFrame(char data, int sequenceNumber) {
        this.info = data;
        this.sequenceNumber = sequenceNumber;
    }
}

class Acknowledgement implements Serializable {
    public boolean ack;
    public int sequenceNumber;

    Acknowledgement() {
    }

    Acknowledgement(boolean ack, int sequenceNumber) {
        this.ack = ack;
        this.sequenceNumber = sequenceNumber;
    }
}

public class Client {

    public static void main(String[] args) {
        try {
            System.out.println("Client Started");
            Socket socket = new Socket("localhost", 12345);

            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream objectInputStream = new
ObjectInputStream(socket.getInputStream());

            int noOfBits = 16;
            int windowSize = 5;
```

```
            DataFrame[] data = new DataFrame[noOfBits];
            int left_ptr = 0;
            int right_ptr = 0;

            while (left_ptr < noOfBits) {
                while (right_ptr < noOfBits && (right_ptr - left_ptr) <
windowSize) {
                    data[right_ptr] = new DataFrame((char) ('a' + right_ptr),
right_ptr);
                    objectOutputStream.writeObject(data[right_ptr]);
                    System.out.println("Sent: " + data[right_ptr].info);
                    right_ptr++;

                }
                Acknowledgement ACK = (Acknowledgement)
objectInputStream.readObject();
                if (ACK.ack) {
                    System.out.println("Acknowledgment received for Frame " +
ACK.sequenceNumber);
                    left_ptr++;
                } else {
                    System.out.println("NACK received for Frame " +
ACK.sequenceNumber + ". Retransmitting.");
                    int resendIndex = ACK.sequenceNumber;
                    objectOutputStream.writeObject(data[resendIndex]);
                    System.out.println("Sent (Retransmit): " +
data[resendIndex].info);
                }

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

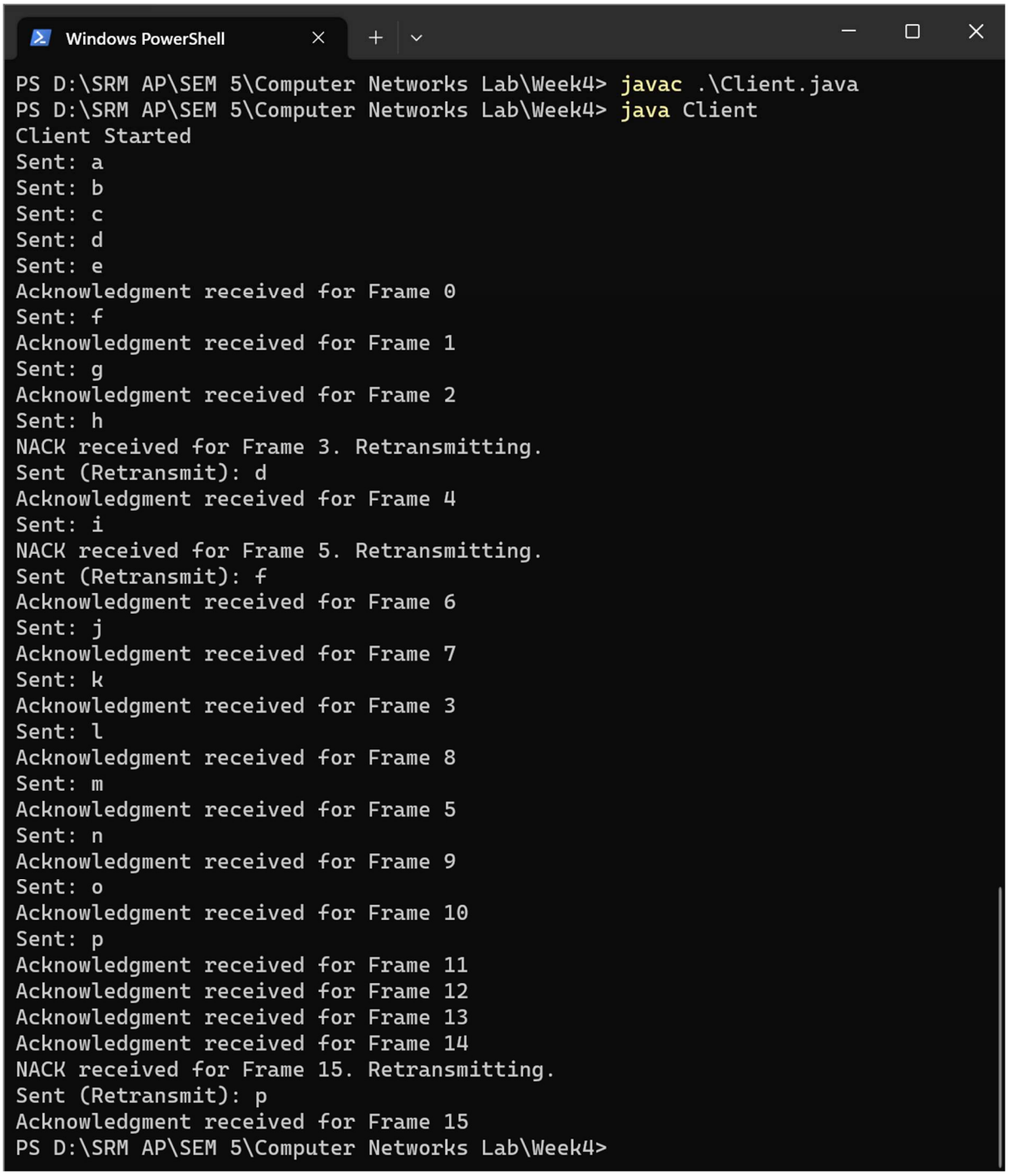
Receiver program:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
import java.util.Map;

class DataFrame implements Serializable {
    public char info;
```

```
    public int sequenceNumber;

    DataFrame() {
    }

    DataFrame(char data, int sequenceNumber) {
        this.info = data;
        this.sequenceNumber = sequenceNumber;
    }
}

class Acknowledgement implements Serializable {
    public boolean ack;
    public int sequenceNumber;

    Acknowledgement() {
    }

    Acknowledgement(boolean ack, int sequenceNumber) {
        this.ack = ack;
        this.sequenceNumber = sequenceNumber;
    }
}

public class Server {
    public static void main(String[] args) {
        try {
            System.out.println("Waiting for clients");
            ServerSocket ss = new ServerSocket(12345);
            Socket s = ss.accept();
            System.out.println("Client connected");

            ObjectInputStream objectInputStream = new
ObjectInputStream(s.getInputStream());
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(s.getOutputStream());

            Map<Integer, Boolean> explicitError = new HashMap<>();
            explicitError.put(3, true);
            explicitError.put(5, true);
            explicitError.put(15, true);
            int framesReceived = 0;
            while (framesReceived < 16) {
                DataFrame receivedFrame = (DataFrame)
objectInputStream.readObject();
                framesReceived++;
                if (explicitError.containsKey(receivedFrame.sequenceNumber)
                        && explicitError.get(receivedFrame.sequenceNumber)) {
```

```
                    Acknowledgement ACK = new Acknowledgement(false,
receivedFrame.sequenceNumber);
                    objectOutputStream.writeObject(ACK);
                    System.out.println("Sent NACK for " +
receivedFrame.sequenceNumber);
                    explicitError.put(receivedFrame.sequenceNumber, false);
                    framesReceived--;
                } else {
                    System.out.println("Received: " + receivedFrame.info);
                    Acknowledgement ACK = new Acknowledgement(true,
receivedFrame.sequenceNumber);
                    objectOutputStream.writeObject(ACK);
                    System.out.println("Sent Acknowledgement for " +
receivedFrame.sequenceNumber);
                }

            }
        } catch (

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

Sender Output:



Receiver Output: