IMPLEMENTATION OF GO BACK N PROTOCOL USING JAVA AND COMPARISON OF GO BACK N WITH SELECTIVE REPEAT

Report submitted to SASTRA Deemed to be

University as the requirement of the course

CSE302: COMPUTER NETWORKS

Submitted by

CHAKKA SAI VENKAT

Reg.No:124160012

B.Tech Electronics and Communication Engineering (Cyber Physical Systems)

November 2022

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

THANJAVUR

TAMILNADU, INDIA -613401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

THANJAVUR, TAMILNADU – 613401

Bonafide Certificate

This is to certify that the report titled "**IMPLEMENTATION OF GO BACK N PROTOCOL USING JAVA AND COMPARISON OF GO BACK N WITH SELECTIVE REPEAT**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech is a bonafide record of the work done by Shri/Ms. **CHAKKA SAI VENKAT (Reg No : 124160012,ECE CPS)** during the academic year 2022 ,in the School of Electrical and Electronic Engineering.

Project Based Work Viva voce held on

Examiner 1 Examiner 2

ACKNOWLEDGEMENTS

First of all, I express my gratitude to Prof. Dr S Vaidhyasubramaniam, Vice Chancellor, SASTRA Deemed to be University, who provided all facilities and constant encouragement during my study. I extend my sincere thanks to Prof. R Chandramouli, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project. It is my privilege to express our sincerest thanks to my project coordinator, Dr. Rajesh Sir who motivated me during the project.

I owe a debt of most profound gratitude to my mentor Dr. Rajesh A sir (ECE/SEEE) for her valuable inputs, able guidance, encouragement, wholehearted cooperation, and constructive criticism throughout my project on the topic “Implementation of Go Back N protocol using JAVA GUI”.

I take this opportunity to thank all my lecturers who have directly or indirectly helped my project.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

GBN	Go Back-N (Transport Layer Protocol)
SR	Selective Repeat (Transport Layer Protocol)
PKT	Packet (Used to mention the transport layer segments)
RTT	Round Trip Time (Time taken by the packet to send and receive the acknowledgement)
ACK	Positive Acknowledgement (A packet sent by the receiver upon receiving a valid packet)
NACK	Negative Acknowledgement(A packet sent by the receiver upon receiving a corrupted packet)
RDT	Reliable Data Transfer

ABSTRACT

The programmer has to provide particular method to transmit data over the internet, to the receiver, which may or may not have a direct link from client to host. While ensuring this reliability, one needs to be ensured that the different rates at which sender/receiver process data should not be a flow control and reliability must be achieved with minimum possible usage of bandwidth. These are the main responsibilities of Transport Layer.

It must be remembered that, underlying network layer (IP), is unreliable, and thus a transferred packet, has a good probability to be dropped/lost, by an intermediate router, or may be overly delayed, in reaching receiver. Thus, a feedback mechanism must be used, between the sender and the receiver. It is ensured that acknowledgment not receiving with in the time is considered as Packets got lost in the network. So, it retransmits again, which makes the channel bandwidth more.

We deal with packets with positive acknowledgement as well as negative acknowledgement. The above must be done efficiently with minimal wastage of resources.

REFERENCES

1. Computer_Networking_A_Top-Down_Approach -6th Edition-KUROSE | ROSS.

2. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4737114

3.JAVA The Complete Reference-11th Edition – Herbert Schildt.

4. https://www.javatpoint.com/java-swing

5. StackOverflow

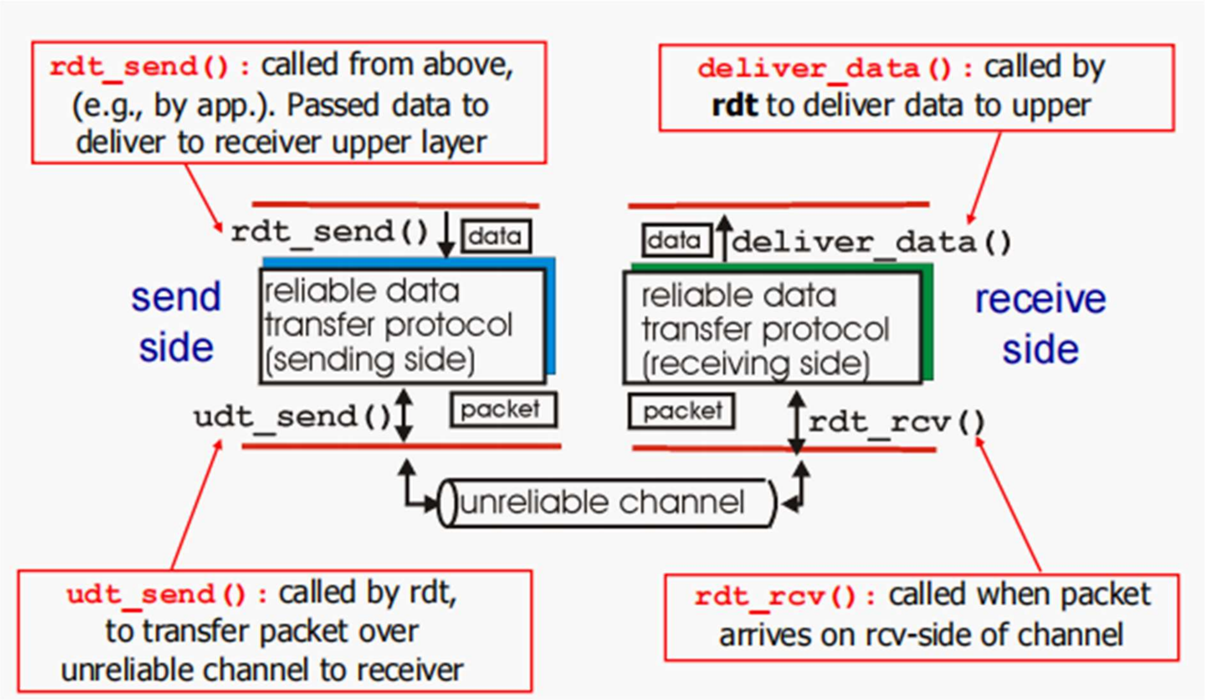
INDEX

CHAPTER 1

INTRODUCTION

1. Reliable Data Transfer

- **CHECK SUM** plays an important role in the reliable data Transfer. To know whether the content in the packets being lost. In network point of view the information in the packet is being corrupted means there is flipping of 1 s and 0 s.
- This problem not only occurs in the Transport Layer but also in the Link Layer. This is the responsibility of the Reliable Data protocol to implement this service abstraction.
- This process can be achieved by the layer below the reliable transfer is unreliable.



rdt_send() : called from above, (e.g., by app.). Passed data to deliver to receiver upper layer
deliver_data() : called by rdt to deliver data to upper
rdt_send()
data
data
deliver_data()
send side
reliable data transfer protocol (sending side)
reliable data transfer protocol (receiving side)
receive side
udt_send()
packet
packet
rdt_rcv()
unreliable channel
udt_send() : called by rdt, to transfer packet over unreliable channel to receiver
rdt_rcv() : called when packet arrives on rcv-side of channel

FIG 1.1: Reliable Data Transfer

2. STOP AND WAIT PROTOCOL

- In the beginning of the discussion, to know about the pipelined protocols we should know about the functioning of the simple protocol “STOP AND WAIT PROTOCOL”. This is the correct protocol, but in the present scenario of high-speed networks this may not work due to its low efficiency.
- STOP AND WAIT protocol basically sends only one packet from the sender to the receiver side at a time. After getting the acknowledgment for that packet again it will send the next packet.
- For sending these packets the sender follows timer for each packet and the sequence number. If a packet is lost in the middle of the process, the sender will notify the timer and within the time the acknowledgment hasn’t received from receiver means it again sends the packet.
- While in the case of packet is sent to the receiver and acknowledgement is also sent by receiver but the ACK is lost in the middle of the internet, again the sender will send the packet with different sequence number to avoid the duplicate packets at the receiver side.
- The receiver sends both the ACK and NACK as a part of transmission. This ACK is sent when the packet is received to the receiver without any bit errors. It sends NACK (negative Acknowledgement) when the packet contains bit errors.
- Sending ACK and NACK makes the programming much complex. So sending the ACK for the last correct packet makes the programming or algorithm part much simpler.
- To overcome all these drawbacks and to achieve good efficiency of the transmission of packets, a new topic is introduced called **PipeLining.**
- PipeLining Operation here refers to the sending the multiple packets at a time from sender to receiver.
- Popular Techniques of this PipeLining process are
    1. Go- Back N
    2. Selective Repeat
- RTT stands for Round Trip Time. RTT is time for a sender to send the packet and time to analyse the packet (with bit errors or without bit errors) and to send the Acknowledgement.

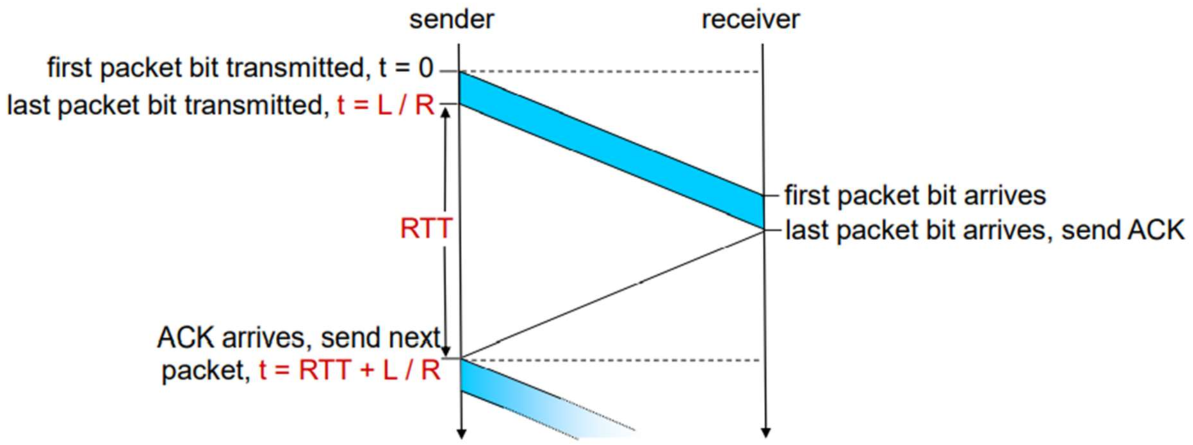

sender
receiver
first packet bit transmitted, t = 0
last packet bit transmitted, t = L / R
RTT
first packet bit arrives
last packet bit arrives, send ACK
ACK arrives, send next packet, t = RTT + L / R

FIG 1.2: Stop and Wait Protocol

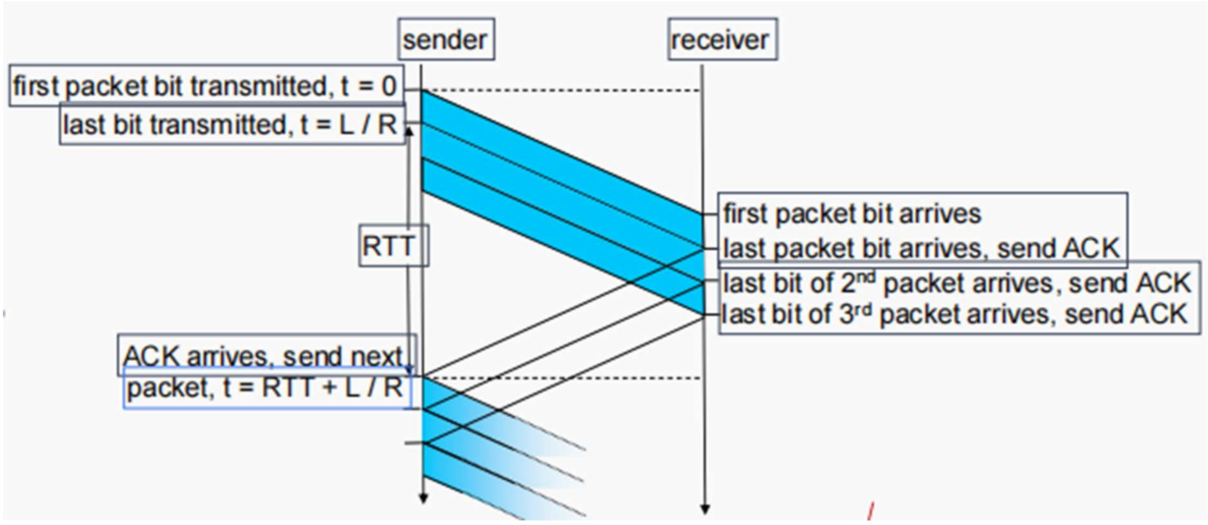

sender
receiver
first packet bit transmitted, t = 0
last bit transmitted, t = L / R
RTT
first packet bit arrives
last packet bit arrives, send ACK
last bit of 2nd packet arrives, send ACK
last bit of 3rd packet arrives, send ACK
ACK arrives, send next
packet, t = RTT + L / R

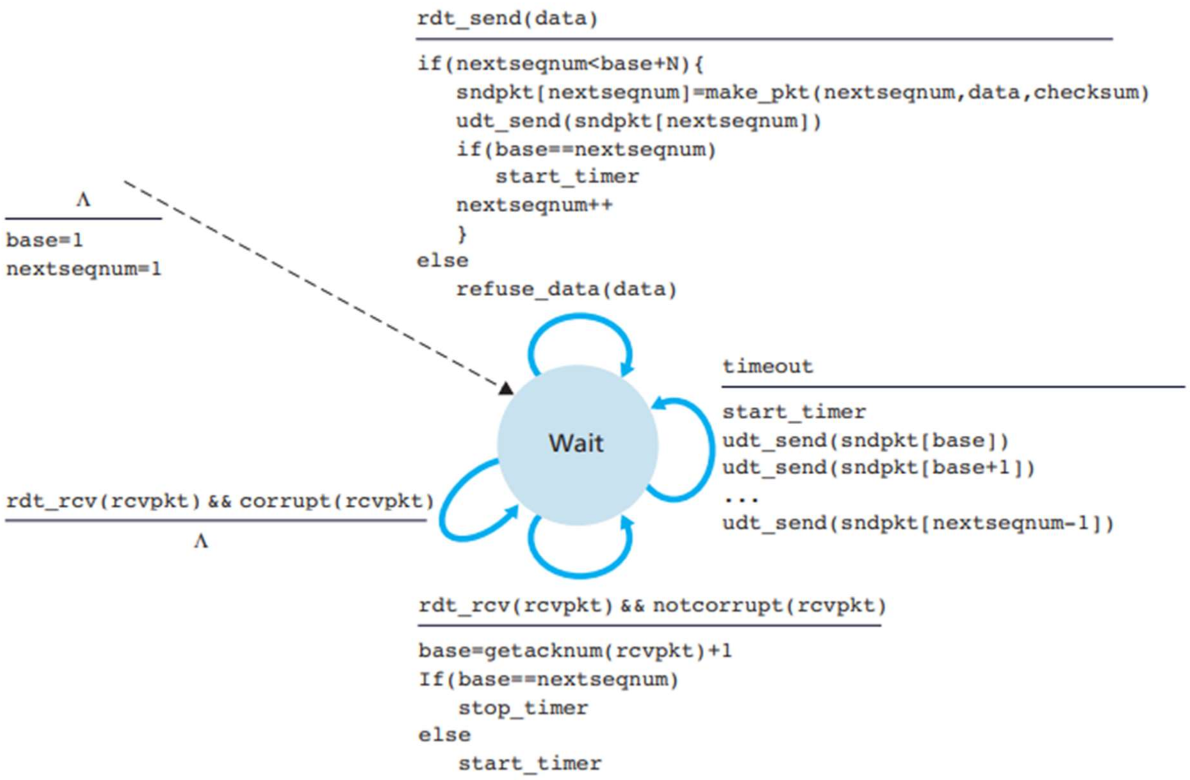
FIG 1.3 : Pipelined Operation

CHAPTER 2

GO-BACK-N PROTOCOL

- Sender in the Go-Back N is allowed to send the N packets at a time without any acknowledgement. That is sender can have N unacknowledged packets in the pipeline.
- Receiver only sends cumulative acknowledgement. It does not acknowledge the packets if there is gap in the middle.
- Sender has timer for oldest unacknowledged packets. Before the timer expires, the sender should receive the acknowledgement from the receiver regarding that the packets are received or not.
- If sender doesn’t receive any ack it should retransmit the packets again from where it didn’t get the Acknowledgement.

WORKING:



```
rdt_send(data)
_______________
if(nextseqnum<base+N){
   sndpkt[nextseqnum]=make_pkt(nextseqnum,data,checksum)
   udt_send(sndpkt[nextseqnum])
   if(base==nextseqnum)
      start_timer
   nextseqnum++
   }
else
   refuse_data(data)
```

```
Λ
_____________
base=1
nextseqnum=1
```

```
timeout
_______________
start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
...
udt_send(sndpkt[nextseqnum-1])
```

```
rdt_rcv(rcvpkt) && corrupt(rcvpkt)
__________________________________
                Λ
```

```
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
_____________________________________
base=getacknum(rcvpkt)+1
If(base==nextseqnum)
   stop_timer
else
   start_timer
```

FIG 2.1: FSM DIAGRAM FOR GO-BACK-N SENDER

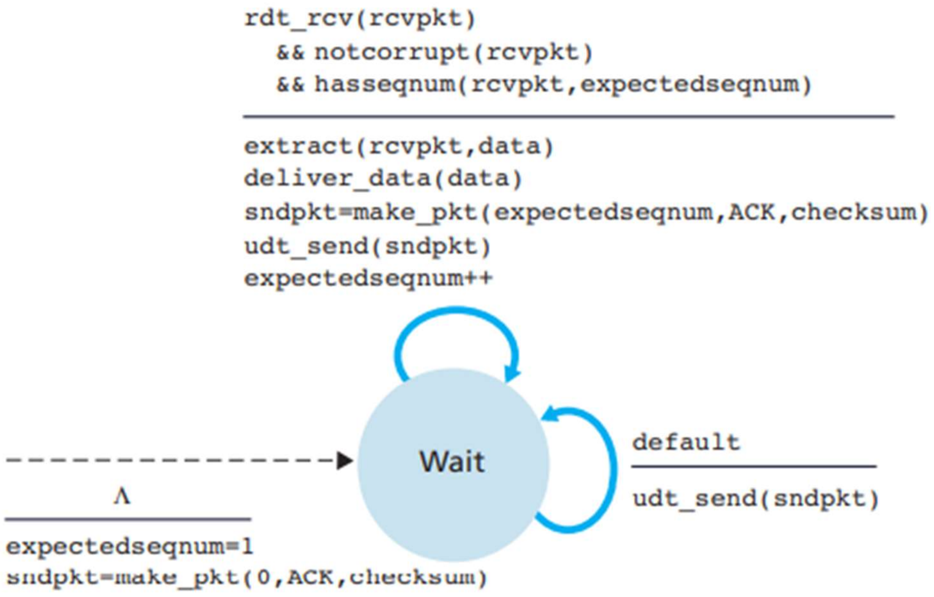


FIG 2.2: FSM DIAGRAM FOR GO-BACK-N RECEIVER

EXPLAINATION:

In the sender side as the data is sent, if the base+N value is grater than the nextSeqNo then create the packets of same size with parameters nextSeqNo , data and checksum. Now send the packet along the unreliable channel. If base value is equal to nextSeqNo, start the timer and increment the nextSeqNo. If the base+N value is less than or equal to nextSeqNo means just reject the data.

If the timeout condition is occurred, start the timer, and send all the packets from the packets which did not received acknowledgment to all the packets till the transmitted.

If the packet is received successfully and the packets do not have errors in the data, then stop the timer or else start the timer. If the packets haven't received or the packets had errors means retransmit the packets from the base to the last sent packets through the unreliable channel.

In the case of the Receiver side of GBN , if the packet is received successfully without any errors and with expectedSeqNo ,extract the packet and send it to the next layer with expectedseqNo , acknowledgement and checksum . Increment the expectedSeqNum and the process continues like this.

- Base is the starting packet index of the window and nextseqNum is the index of the next packet's index that need to be transmitted.
- Range [0, base-1]: packets already sent, and acknowledged by receiver.
- Range [base, nextseqNum-1]: packets are sent ,but packets yet to acknowledge.
- Range [nextseqNum, base+N-1] : sequence numbers that are sent immediately with packet ,as and when data is given by upper layer.
- Range [base+N,till last_seqNum] : these packets can't be used until a unacknowledged packet in the current sender window is acknowledged.

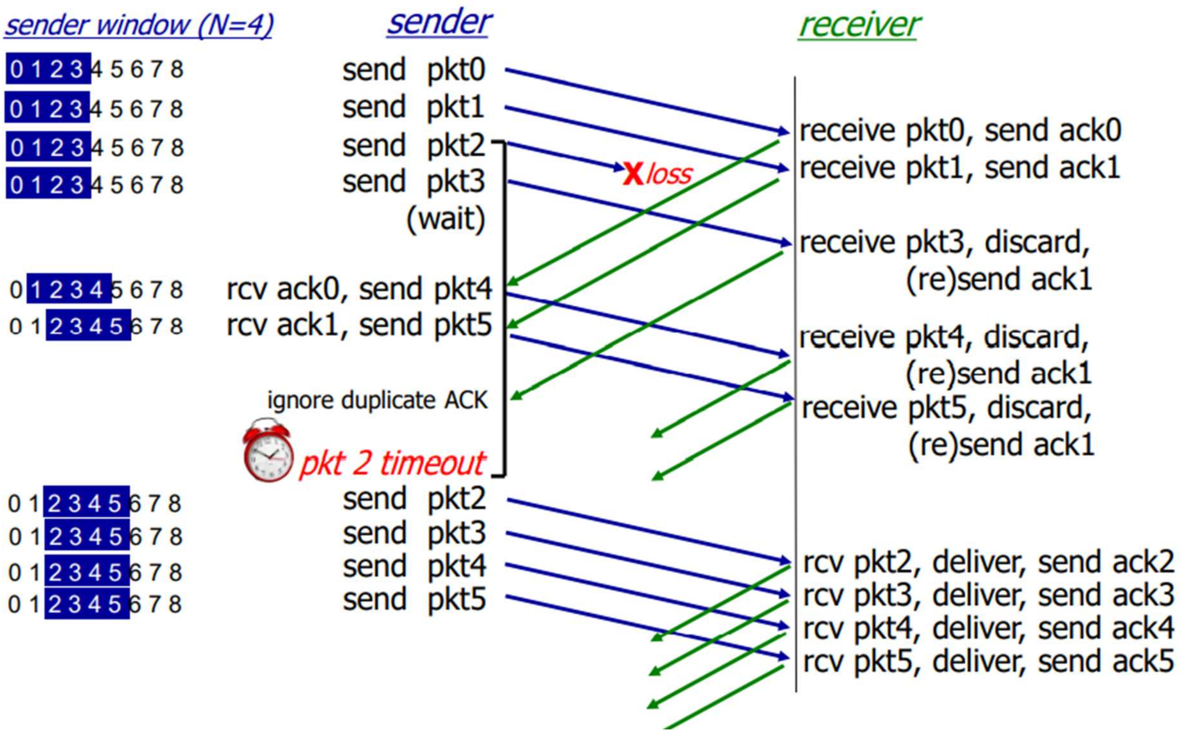
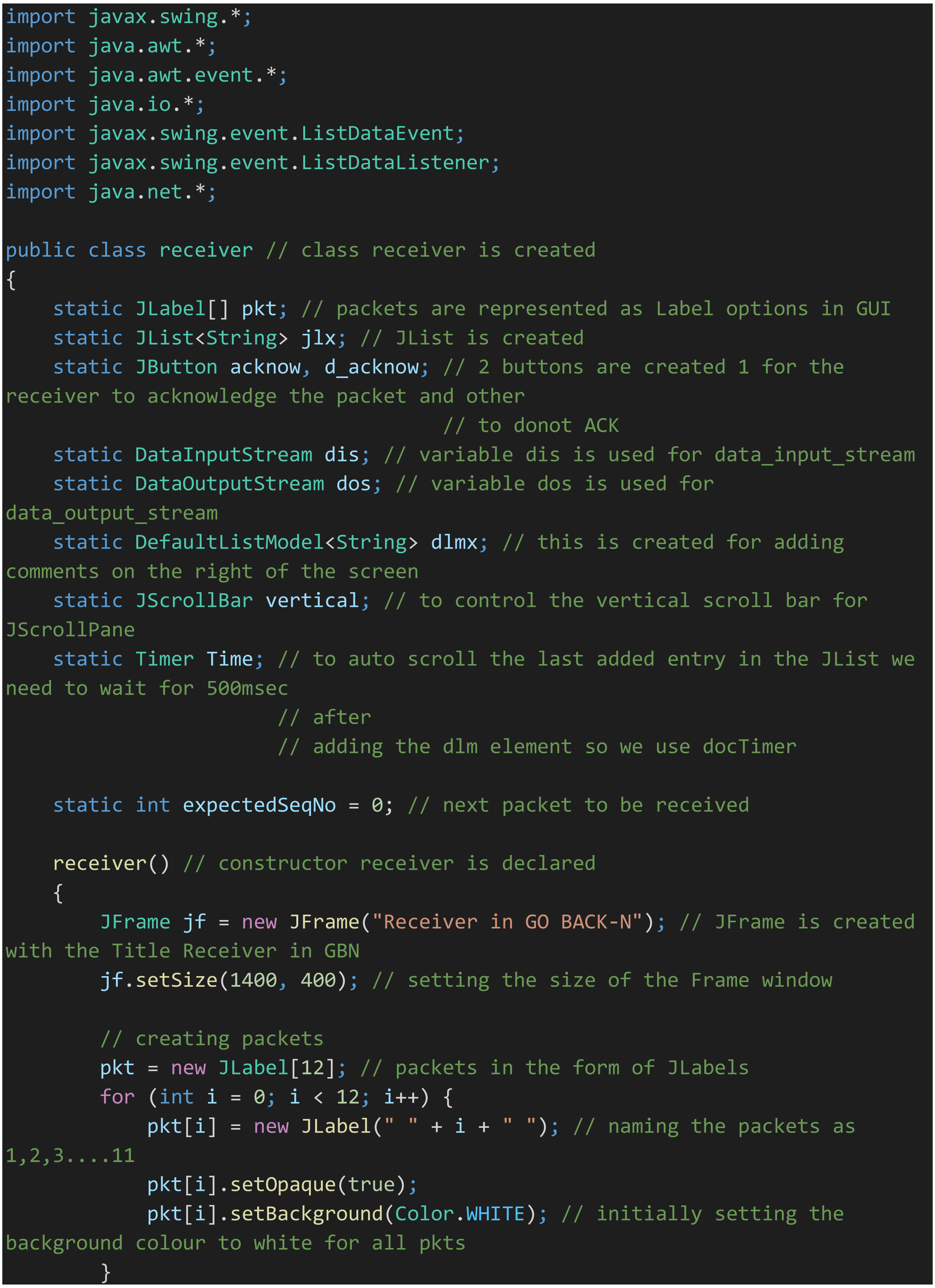


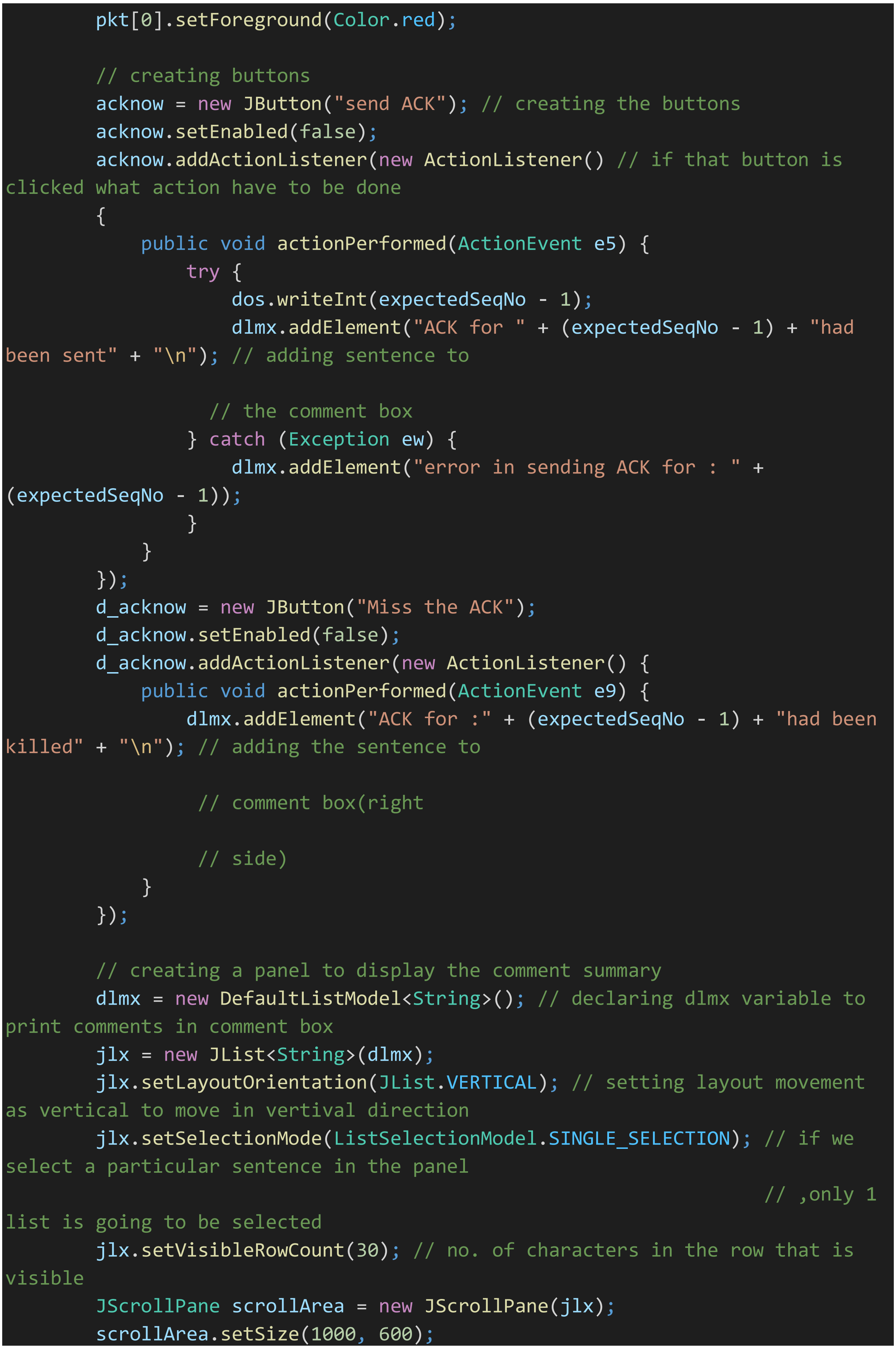
Fig 2.3: Go Back N protocol

CHAPTER 3
SOURCE CODE FOR IMPLEMENTATION OF GBN PROTOCOL

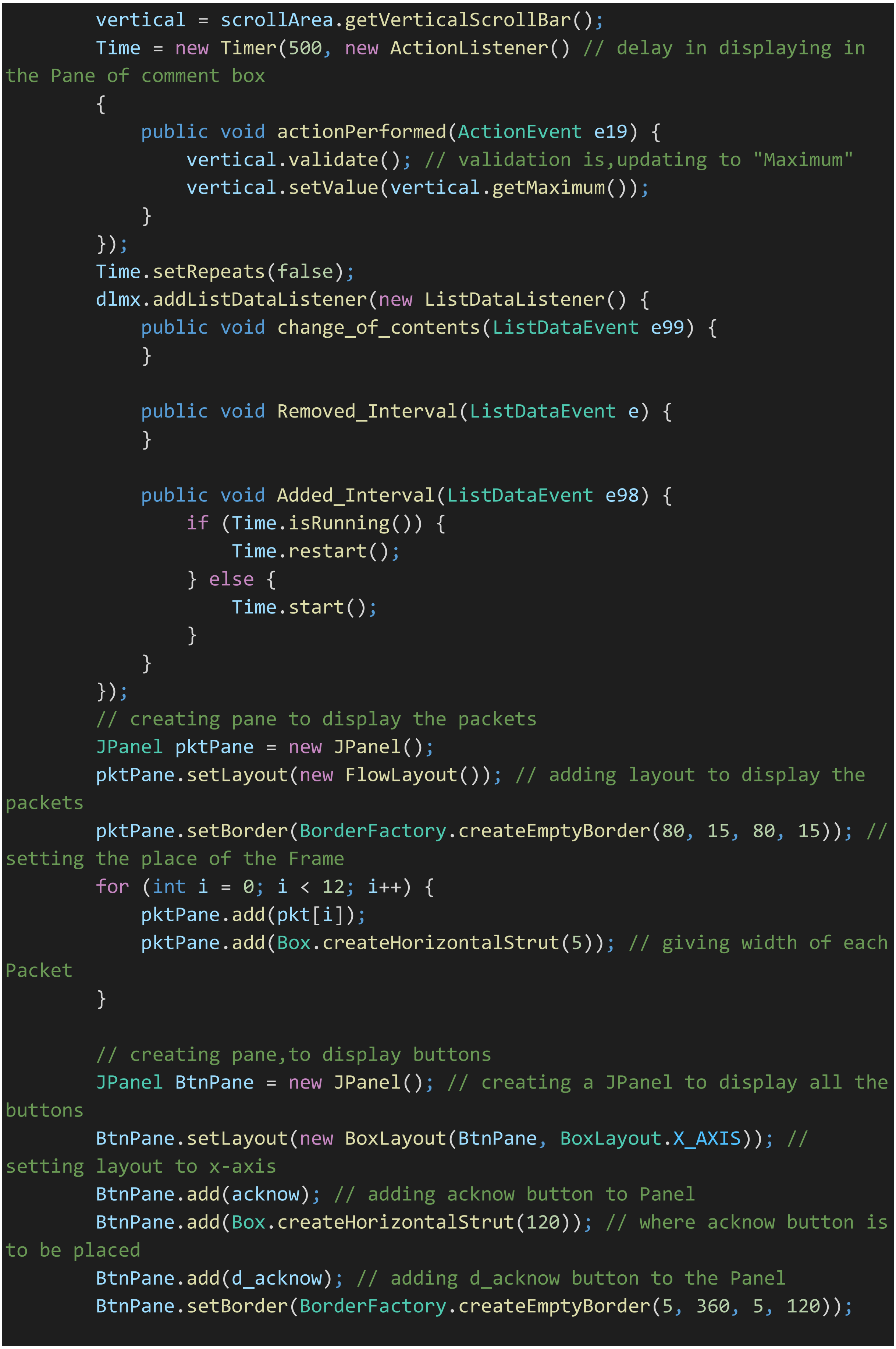
Receiver Side:



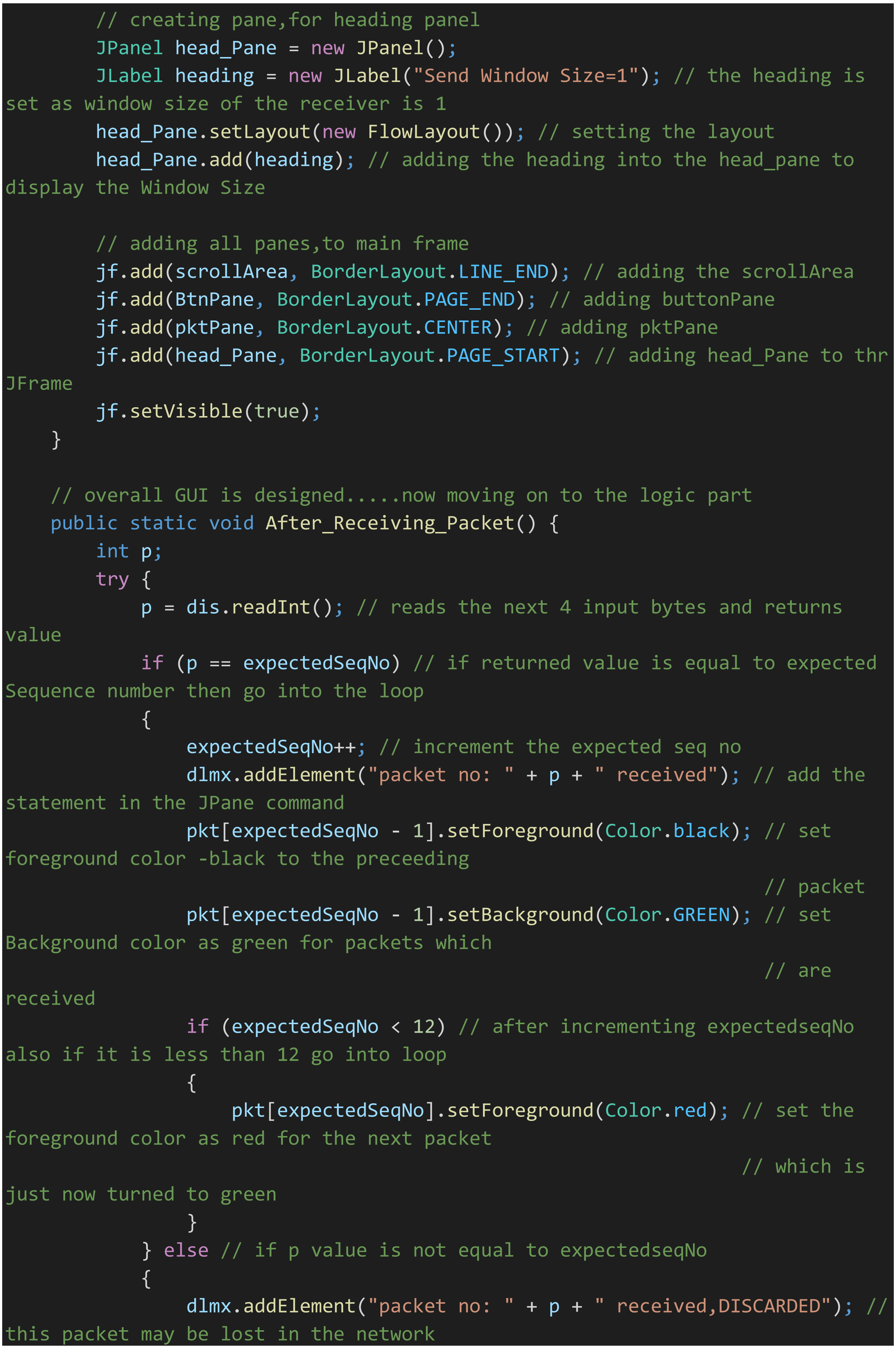
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.event.ListDataEvent;
import javax.swing.event.ListDataListener;
import java.net.*;

public class receiver // class receiver is created
{
    static JLabel[] pkt; // packets are represented as Label options in GUI
    static JList<String> jlx; // JList is created
    static JButton acknow, d_acknow; // 2 buttons are created 1 for the
receiver to acknowledge the packet and other
                                     // to donot ACK
    static DataInputStream dis; // variable dis is used for data_input_stream
    static DataOutputStream dos; // variable dos is used for
data_output_stream
    static DefaultListModel<String> dlmx; // this is created for adding
comments on the right of the screen
    static JScrollBar vertical; // to control the vertical scroll bar for
JScrollPane
    static Timer Time; // to auto scroll the last added entry in the JList we
need to wait for 500msec
                       // after
                       // adding the dlm element so we use docTimer

    static int expectedSeqNo = 0; // next packet to be received

    receiver() // constructor receiver is declared
    {
        JFrame jf = new JFrame("Receiver in GO BACK-N"); // JFrame is created
with the Title Receiver in GBN
        jf.setSize(1400, 400); // setting the size of the Frame window

        // creating packets
        pkt = new JLabel[12]; // packets in the form of JLabels
        for (int i = 0; i < 12; i++) {
            pkt[i] = new JLabel(" " + i + " "); // naming the packets as
1,2,3....11
            pkt[i].setOpaque(true);
            pkt[i].setBackground(Color.WHITE); // initially setting the
background colour to white for all pkts
        }
```



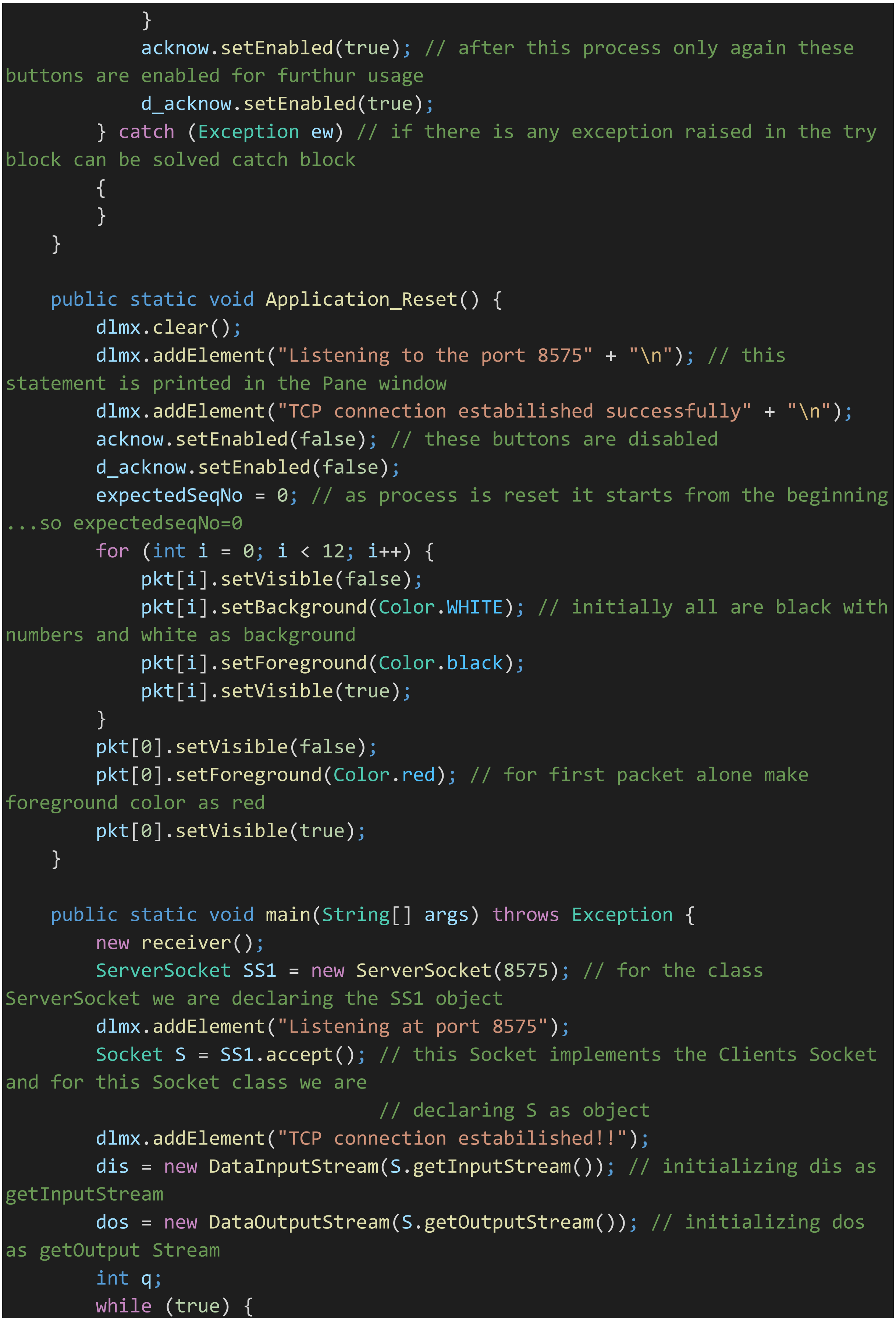
```
        pkt[0].setForeground(Color.red);

        // creating buttons
        acknow = new JButton("send ACK"); // creating the buttons
        acknow.setEnabled(false);
        acknow.addActionListener(new ActionListener() // if that button is
clicked what action have to be done
        {
            public void actionPerformed(ActionEvent e5) {
                try {
                    dos.writeInt(expectedSeqNo - 1);
                    dlmx.addElement("ACK for " + (expectedSeqNo - 1) + "had
been sent" + "\n"); // adding sentence to

                  // the comment box
                } catch (Exception ew) {
                    dlmx.addElement("error in sending ACK for : " +
(expectedSeqNo - 1));
                }
            }
        });
        d_acknow = new JButton("Miss the ACK");
        d_acknow.setEnabled(false);
        d_acknow.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e9) {
                dlmx.addElement("ACK for :" + (expectedSeqNo - 1) + "had been
killed" + "\n"); // adding the sentence to

                 // comment box(right

                 // side)
            }
        });

        // creating a panel to display the comment summary
        dlmx = new DefaultListModel<String>(); // declaring dlmx variable to
print comments in comment box
        jlx = new JList<String>(dlmx);
        jlx.setLayoutOrientation(JList.VERTICAL); // setting layout movement
as vertical to move in vertival direction
        jlx.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // if we
select a particular sentence in the panel
                                                                   // ,only 1
list is going to be selected
        jlx.setVisibleRowCount(30); // no. of characters in the row that is
visible
        JScrollPane scrollArea = new JScrollPane(jlx);
        scrollArea.setSize(1000, 600);
```



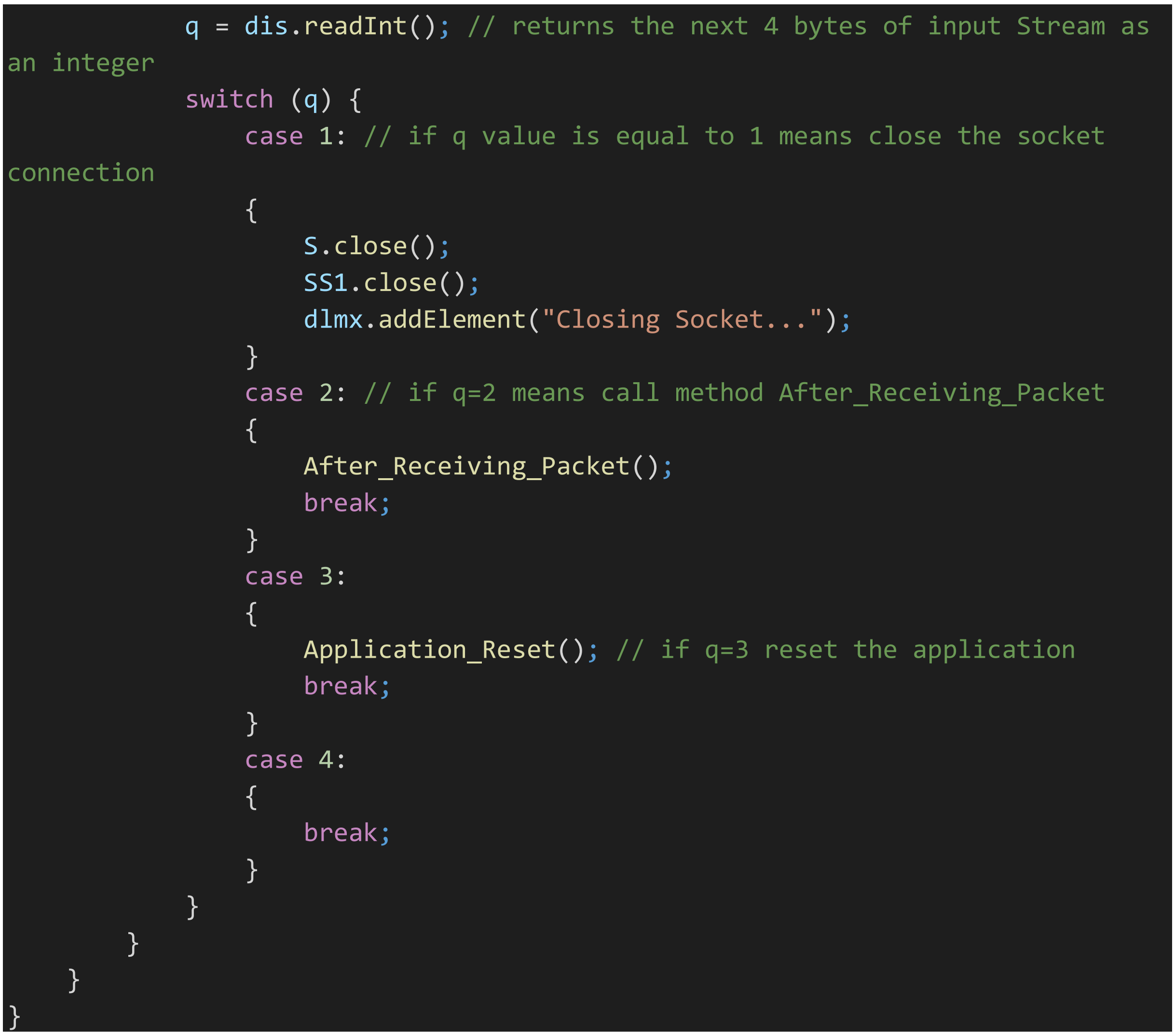
```
        vertical = scrollArea.getVerticalScrollBar();
        Time = new Timer(500, new ActionListener() // delay in displaying in
the Pane of comment box
        {
            public void actionPerformed(ActionEvent e19) {
                vertical.validate(); // validation is,updating to "Maximum"
                vertical.setValue(vertical.getMaximum());
            }
        });
        Time.setRepeats(false);
        dlmx.addListDataListener(new ListDataListener() {
            public void change_of_contents(ListDataEvent e99) {
            }

            public void Removed_Interval(ListDataEvent e) {
            }

            public void Added_Interval(ListDataEvent e98) {
                if (Time.isRunning()) {
                    Time.restart();
                } else {
                    Time.start();
                }
            }
        });
        // creating pane to display the packets
        JPanel pktPane = new JPanel();
        pktPane.setLayout(new FlowLayout()); // adding layout to display the
packets
        pktPane.setBorder(BorderFactory.createEmptyBorder(80, 15, 80, 15)); //
setting the place of the Frame
        for (int i = 0; i < 12; i++) {
            pktPane.add(pkt[i]);
            pktPane.add(Box.createHorizontalStrut(5)); // giving width of each
Packet
        }

        // creating pane,to display buttons
        JPanel BtnPane = new JPanel(); // creating a JPanel to display all the
buttons
        BtnPane.setLayout(new BoxLayout(BtnPane, BoxLayout.X_AXIS)); //
setting layout to x-axis
        BtnPane.add(acknow); // adding acknow button to Panel
        BtnPane.add(Box.createHorizontalStrut(120)); // where acknow button is
to be placed
        BtnPane.add(d_acknow); // adding d_acknow button to the Panel
        BtnPane.setBorder(BorderFactory.createEmptyBorder(5, 360, 5, 120));
```



```
        // creating pane,for heading panel
        JPanel head_Pane = new JPanel();
        JLabel heading = new JLabel("Send Window Size=1"); // the heading is
set as window size of the receiver is 1
        head_Pane.setLayout(new FlowLayout()); // setting the layout
        head_Pane.add(heading); // adding the heading into the head_pane to
display the Window Size

        // adding all panes,to main frame
        jf.add(scrollArea, BorderLayout.LINE_END); // adding the scrollArea
        jf.add(BtnPane, BorderLayout.PAGE_END); // adding buttonPane
        jf.add(pktPane, BorderLayout.CENTER); // adding pktPane
        jf.add(head_Pane, BorderLayout.PAGE_START); // adding head_Pane to thr
JFrame
        jf.setVisible(true);
    }

    // overall GUI is designed.....now moving on to the logic part
    public static void After_Receiving_Packet() {
        int p;
        try {
            p = dis.readInt(); // reads the next 4 input bytes and returns
value
            if (p == expectedSeqNo) // if returned value is equal to expected
Sequence number then go into the loop
            {
                expectedSeqNo++; // increment the expected seq no
                dlmx.addElement("packet no: " + p + " received"); // add the
statement in the JPane command
                pkt[expectedSeqNo - 1].setForeground(Color.black); // set
foreground color -black to the preceeding
                                                                   // packet
                pkt[expectedSeqNo - 1].setBackground(Color.GREEN); // set
Background color as green for packets which
                                                                   // are
received
                if (expectedSeqNo < 12) // after incrementing expectedseqNo
also if it is less than 12 go into loop
                {
                    pkt[expectedSeqNo].setForeground(Color.red); // set the
foreground color as red for the next packet
                                                                 // which is
just now turned to green
                }
            } else // if p value is not equal to expectedseqNo
            {
                dlmx.addElement("packet no: " + p + " received,DISCARDED"); //
this packet may be lost in the network
```



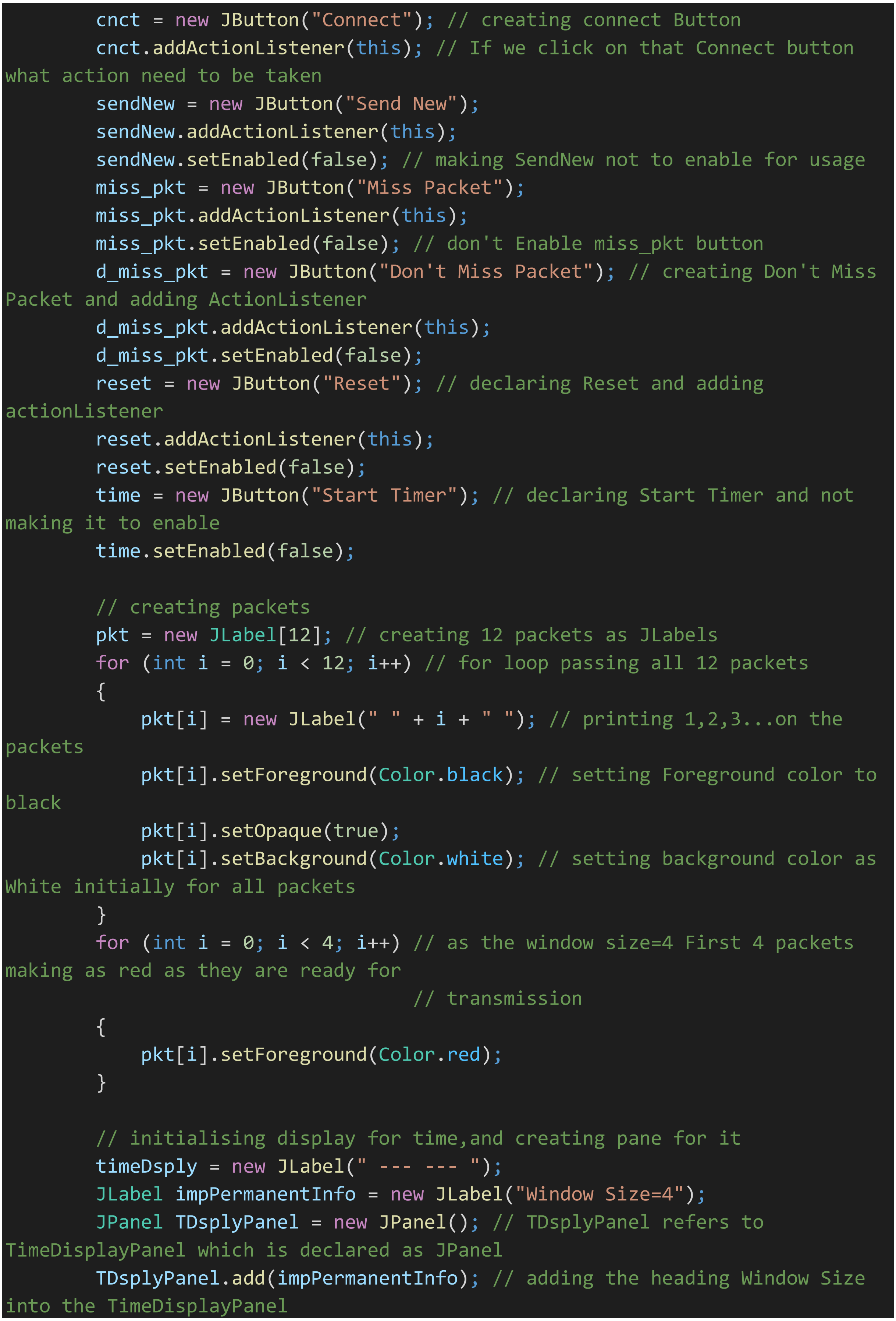
```
            }
            acknow.setEnabled(true); // after this process only again these
buttons are enabled for furthur usage
            d_acknow.setEnabled(true);
        } catch (Exception ew) // if there is any exception raised in the try
block can be solved catch block
        {
        }
    }

    public static void Application_Reset() {
        dlmx.clear();
        dlmx.addElement("Listening to the port 8575" + "\n"); // this
statement is printed in the Pane window
        dlmx.addElement("TCP connection estabilished successfully" + "\n");
        acknow.setEnabled(false); // these buttons are disabled
        d_acknow.setEnabled(false);
        expectedSeqNo = 0; // as process is reset it starts from the beginning
...so expectedseqNo=0
        for (int i = 0; i < 12; i++) {
            pkt[i].setVisible(false);
            pkt[i].setBackground(Color.WHITE); // initially all are black with
numbers and white as background
            pkt[i].setForeground(Color.black);
            pkt[i].setVisible(true);
        }
        pkt[0].setVisible(false);
        pkt[0].setForeground(Color.red); // for first packet alone make
foreground color as red
        pkt[0].setVisible(true);
    }

    public static void main(String[] args) throws Exception {
        new receiver();
        ServerSocket SS1 = new ServerSocket(8575); // for the class
ServerSocket we are declaring the SS1 object
        dlmx.addElement("Listening at port 8575");
        Socket S = SS1.accept(); // this Socket implements the Clients Socket
and for this Socket class we are
                                 // declaring S as object
        dlmx.addElement("TCP connection estabilished!!");
        dis = new DataInputStream(S.getInputStream()); // initializing dis as
getInputStream
        dos = new DataOutputStream(S.getOutputStream()); // initializing dos
as getOutput Stream
        int q;
        while (true) {
```



```
            q = dis.readInt(); // returns the next 4 bytes of input Stream as
an integer
            switch (q) {
                case 1: // if q value is equal to 1 means close the socket
connection
                {
                    S.close();
                    SS1.close();
                    dlmx.addElement("Closing Socket...");
                }
                case 2: // if q=2 means call method After_Receiving_Packet
                {
                    After_Receiving_Packet();
                    break;
                }
                case 3:
                {
                    Application_Reset(); // if q=3 reset the application
                    break;
                }
                case 4:
                {
                    break;
                }
            }
        }
    }
}
```

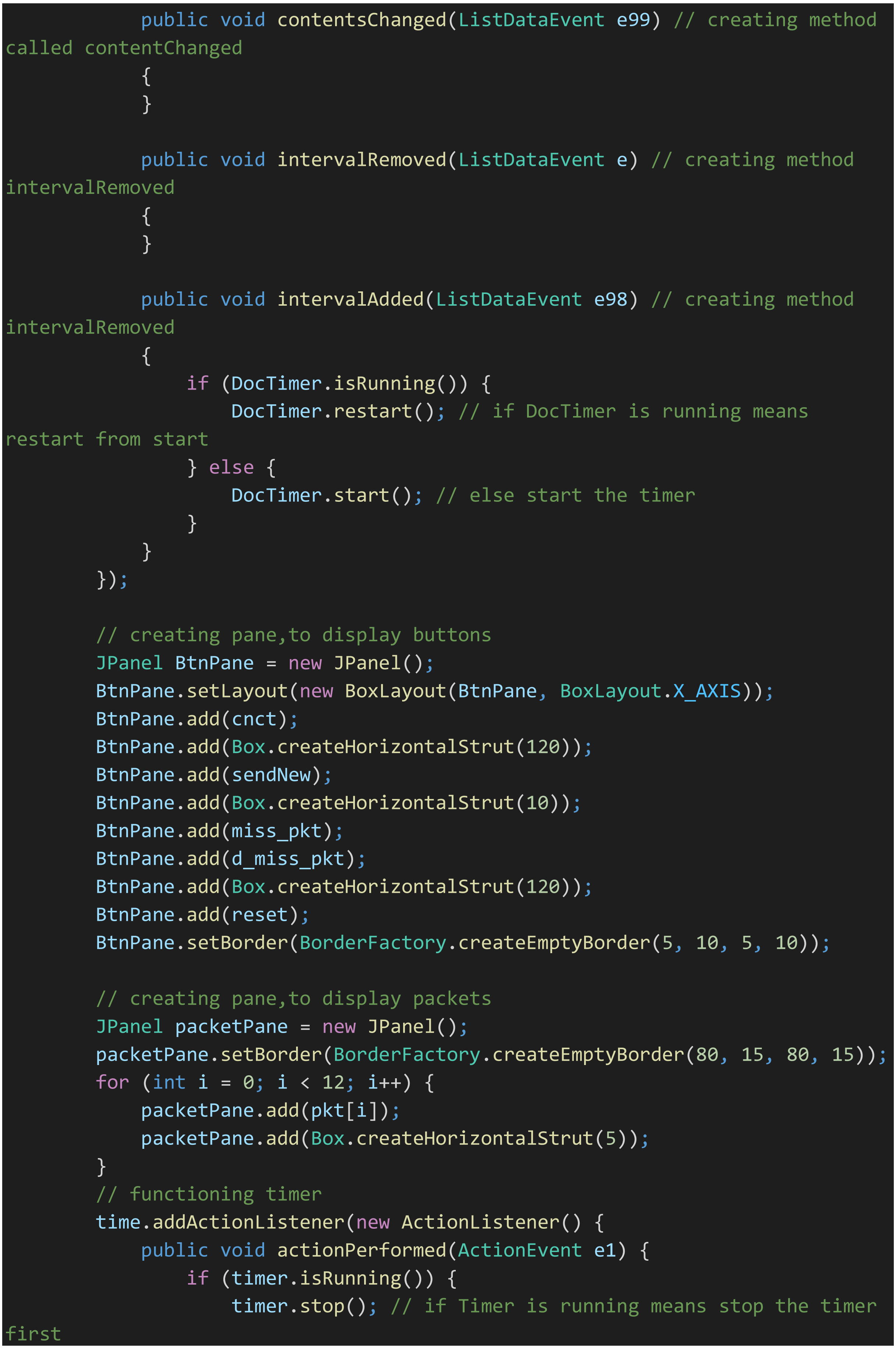
Sender Side :

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.event.ListDataEvent;
import javax.swing.event.ListDataListener;
import java.time.Duration;
import java.time.LocalDateTime;

public class sender implements ActionListener // class sender is created and adds a ActionListener
{
    static JButton cnct, reset, time, sendNew, miss_pkt, d_miss_pkt; // creating JButtons for all
    static JLabel[] pkt; // declaring pkt as JLabel
    static JList<String> jl1; // jl1 as JList for displaying in the JArea
    static DefaultListModel<String> dlmx; // this dlmx is used to add the strings in the JArea
    static JScrollBar vertical; // Activating Scroll Bar to move in the vertical
    static Timer DocTimer; // the delay time required for filling the JPane in the right side
    Socket S; // S is the object created for the class Socket
    static Timer timer;
    static JLabel timeDsply; // creating timeDisplay as JLabel
    LocalDateTime startTime;
    Duration duration = Duration.ofSeconds(12);
    static int base = 0; // base represents the first number packet of the window
    static int nextSeqNo = 0; // nextseqNo is the next packet that need to be sent
    static JLabel seqNoLabel, baseLabel;
    static DataInputStream dis; // declaring dis as static DataInputStream variable
    static DataOutputStream dos; // declaring dos as static DataOutputStream variable

    sender() // constructor declaration
    {
        JFrame jf = new JFrame("Sender in GBN"); // creating JFrame and title of the Frame
        jf.setSize(1300, 300); // setting Size of the JFrame
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // if we click on close button it should close

        // creating butons
```

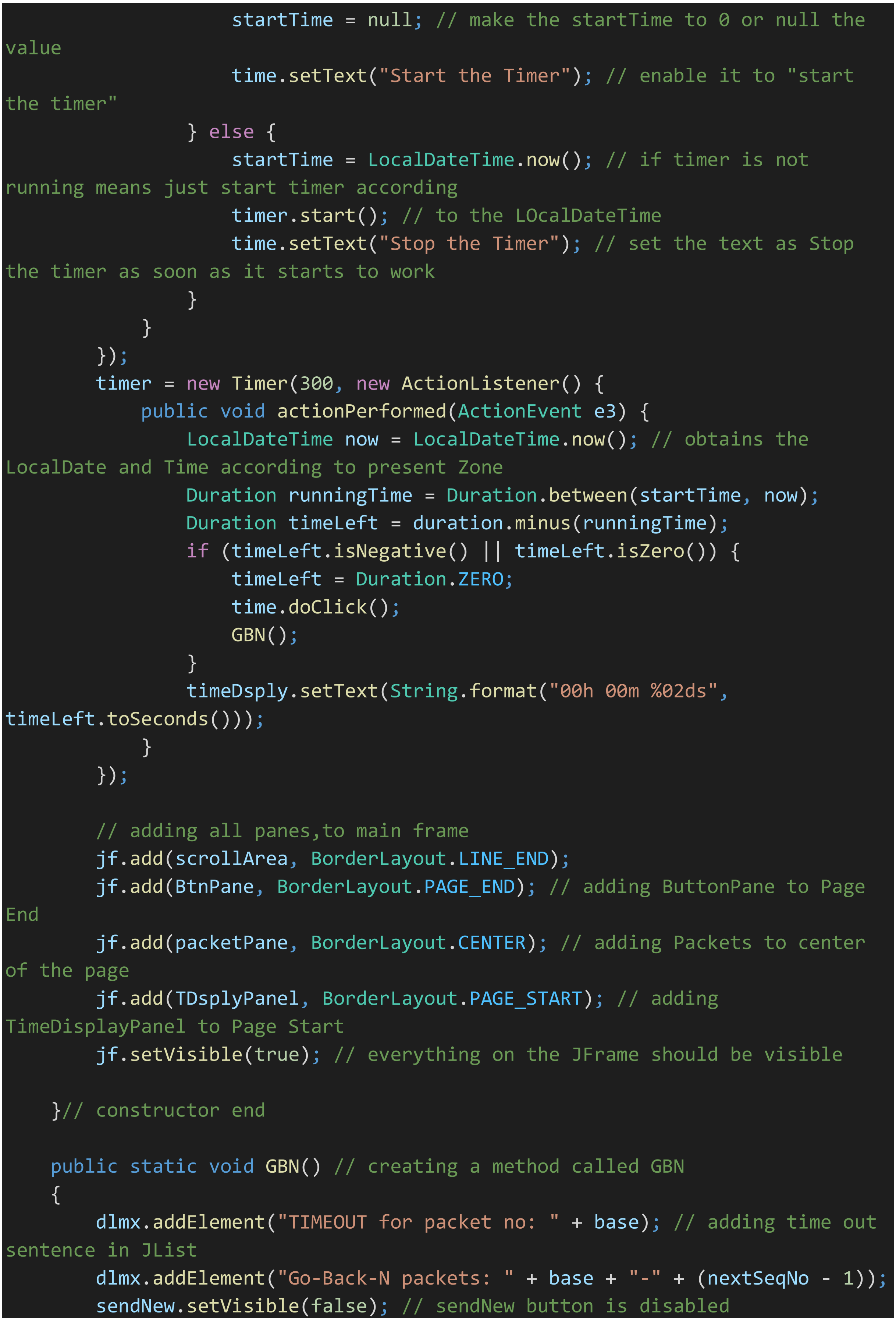


```
        cnct = new JButton("Connect"); // creating connect Button
        cnct.addActionListener(this); // If we click on that Connect button
what action need to be taken
        sendNew = new JButton("Send New");
        sendNew.addActionListener(this);
        sendNew.setEnabled(false); // making SendNew not to enable for usage
        miss_pkt = new JButton("Miss Packet");
        miss_pkt.addActionListener(this);
        miss_pkt.setEnabled(false); // don't Enable miss_pkt button
        d_miss_pkt = new JButton("Don't Miss Packet"); // creating Don't Miss
Packet and adding ActionListener
        d_miss_pkt.addActionListener(this);
        d_miss_pkt.setEnabled(false);
        reset = new JButton("Reset"); // declaring Reset and adding
actionListener
        reset.addActionListener(this);
        reset.setEnabled(false);
        time = new JButton("Start Timer"); // declaring Start Timer and not
making it to enable
        time.setEnabled(false);

        // creating packets
        pkt = new JLabel[12]; // creating 12 packets as JLabels
        for (int i = 0; i < 12; i++) // for loop passing all 12 packets
        {
            pkt[i] = new JLabel(" " + i + " "); // printing 1,2,3...on the
packets
            pkt[i].setForeground(Color.black); // setting Foreground color to
black
            pkt[i].setOpaque(true);
            pkt[i].setBackground(Color.white); // setting background color as
White initially for all packets
        }
        for (int i = 0; i < 4; i++) // as the window size=4 First 4 packets
making as red as they are ready for
                                    // transmission
        {
            pkt[i].setForeground(Color.red);
        }

        // initialising display for time,and creating pane for it
        timeDsply = new JLabel(" --- --- ");
        JLabel impPermanentInfo = new JLabel("Window Size=4");
        JPanel TDsplyPanel = new JPanel(); // TDsplyPanel refers to
TimeDisplayPanel which is declared as JPanel
        TDsplyPanel.add(impPermanentInfo); // adding the heading Window Size
into the TimeDisplayPanel
```

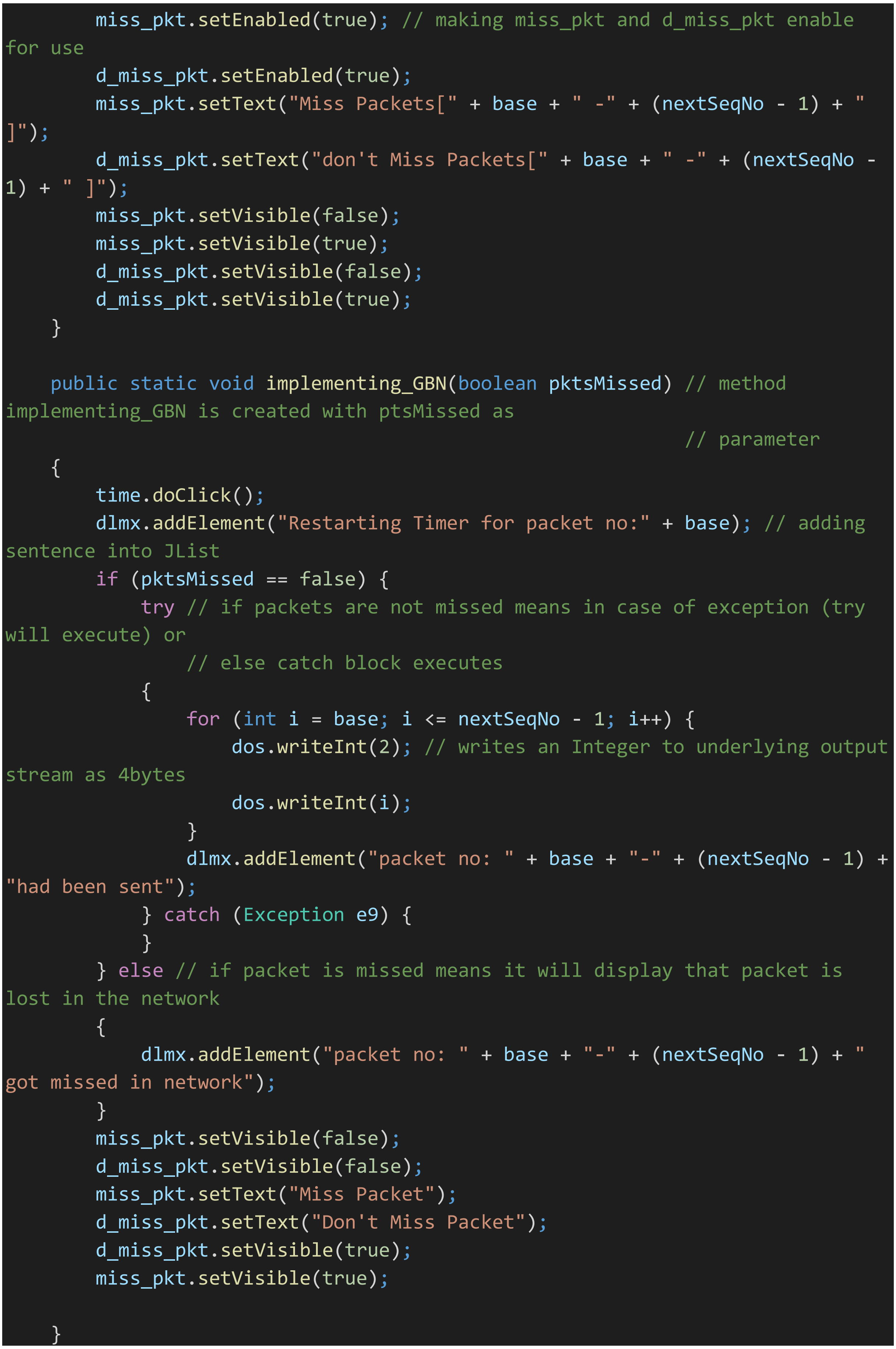
```
        TDsplyPanel.add(Box.createHorizontalStrut(120)); // created an
invisible fixed width of length 120
        TDsplyPanel.add(time); // adding time to timeDisplayPanel
        TDsplyPanel.add(timeDsply);
        TDsplyPanel.add(Box.createHorizontalStrut(10));

        seqNoLabel = new JLabel("next Sequence no:" + nextSeqNo); // creating
JLabel for seqNoLabel
        baseLabel = new JLabel("base :" + base); // creating jLabel for
BaseLabel
        TDsplyPanel.add(Box.createHorizontalStrut(30)); // creating invisible
horizontal width of length 30
        TDsplyPanel.add(seqNoLabel);
        TDsplyPanel.add(Box.createHorizontalStrut(10)); // creating invisible
horizontal width of length 10
        TDsplyPanel.add(baseLabel);
        baseLabel.setOpaque(true);
        baseLabel.setBackground(Color.white); // setting baseLabel as White
Background
        seqNoLabel.setOpaque(true);
        seqNoLabel.setBackground(Color.white); // similarly setting seqNoLabel
as White Background

        // creating pane,to display summary
        dlmx = new DefaultListModel<String>(); // dlmx is like a Array_List
which adds string type into the Area Display
        jl1 = new JList<String>(dlmx); // creating List
        jl1.setVisibleRowCount(20); // Maximum No.of letters going to be
filled in a line
        jl1.setLayoutOrientation(JList.VERTICAL);
        jl1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // if u
click on a list it will just select only one
                                                                   //
List_String at a time
        JScrollPane scrollArea = new JScrollPane(jl1);
        scrollArea.setSize(1000, 300);
        vertical = scrollArea.getVerticalScrollBar();
        DocTimer = new Timer(500, new ActionListener() // delay for strings
noting into JArea Box is 500msec
        {
            public void actionPerformed(ActionEvent e1) {
                vertical.validate();
                vertical.setValue(vertical.getMaximum());
            }
        });
        DocTimer.setRepeats(false);
        dlmx.addListDataListener(new ListDataListener() {
```



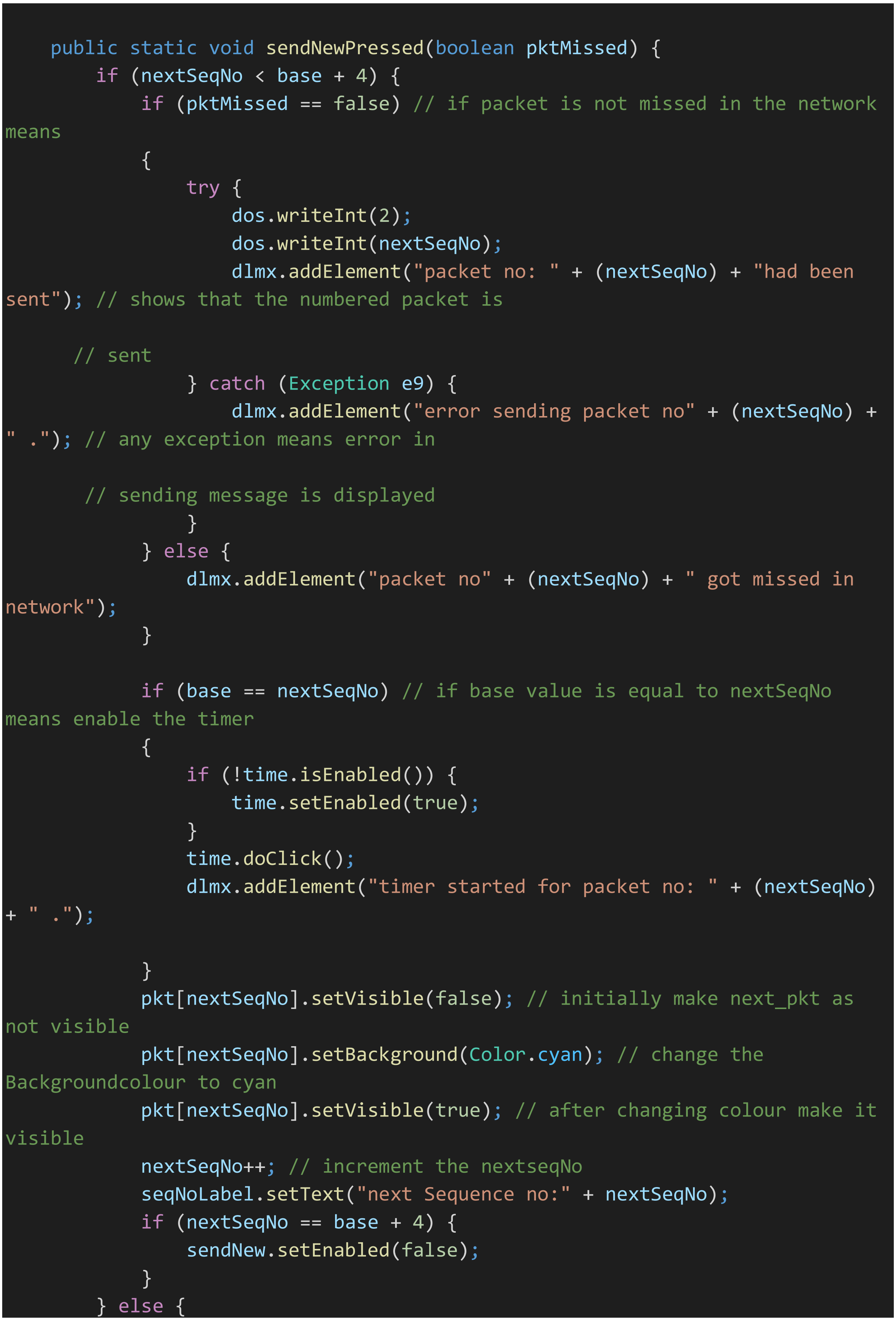
```
            public void contentsChanged(ListDataEvent e99) // creating method
called contentChanged
            {
            }

            public void intervalRemoved(ListDataEvent e) // creating method
intervalRemoved
            {
            }

            public void intervalAdded(ListDataEvent e98) // creating method
intervalRemoved
            {
                if (DocTimer.isRunning()) {
                    DocTimer.restart(); // if DocTimer is running means
restart from start
                } else {
                    DocTimer.start(); // else start the timer
                }
            }
        });

        // creating pane,to display buttons
        JPanel BtnPane = new JPanel();
        BtnPane.setLayout(new BoxLayout(BtnPane, BoxLayout.X_AXIS));
        BtnPane.add(cnct);
        BtnPane.add(Box.createHorizontalStrut(120));
        BtnPane.add(sendNew);
        BtnPane.add(Box.createHorizontalStrut(10));
        BtnPane.add(miss_pkt);
        BtnPane.add(d_miss_pkt);
        BtnPane.add(Box.createHorizontalStrut(120));
        BtnPane.add(reset);
        BtnPane.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));

        // creating pane,to display packets
        JPanel packetPane = new JPanel();
        packetPane.setBorder(BorderFactory.createEmptyBorder(80, 15, 80, 15));
        for (int i = 0; i < 12; i++) {
            packetPane.add(pkt[i]);
            packetPane.add(Box.createHorizontalStrut(5));
        }
        // functioning timer
        time.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e1) {
                if (timer.isRunning()) {
                    timer.stop(); // if Timer is running means stop the timer
first
```



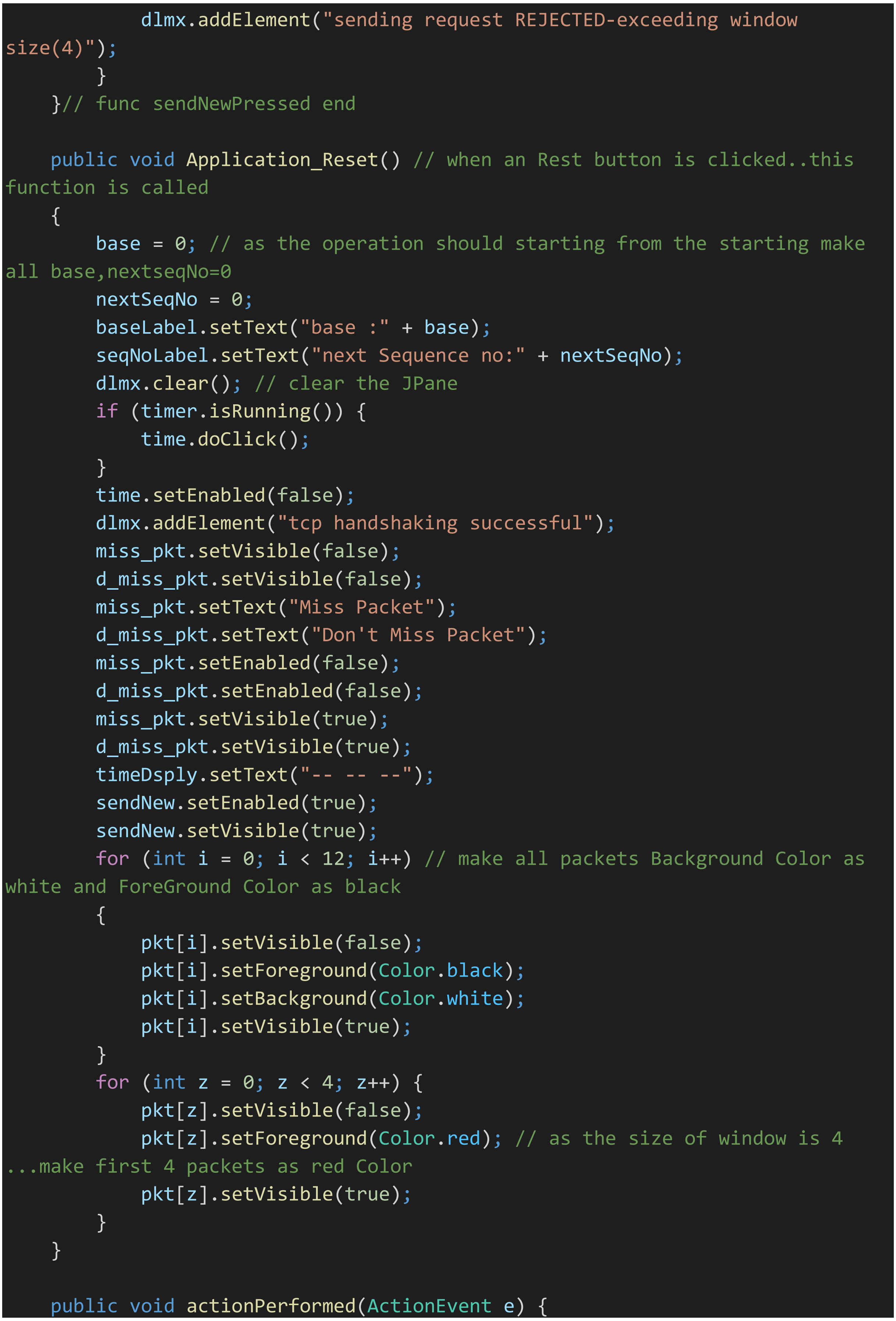
```
                    startTime = null; // make the startTime to 0 or null the
value
                    time.setText("Start the Timer"); // enable it to "start
the timer"
                } else {
                    startTime = LocalDateTime.now(); // if timer is not
running means just start timer according
                    timer.start(); // to the LOcalDateTime
                    time.setText("Stop the Timer"); // set the text as Stop
the timer as soon as it starts to work
                }
            }
        });
        timer = new Timer(300, new ActionListener() {
            public void actionPerformed(ActionEvent e3) {
                LocalDateTime now = LocalDateTime.now(); // obtains the
LocalDate and Time according to present Zone
                Duration runningTime = Duration.between(startTime, now);
                Duration timeLeft = duration.minus(runningTime);
                if (timeLeft.isNegative() || timeLeft.isZero()) {
                    timeLeft = Duration.ZERO;
                    time.doClick();
                    GBN();
                }
                timeDsply.setText(String.format("00h 00m %02ds",
timeLeft.toSeconds()));
            }
        });

        // adding all panes,to main frame
        jf.add(scrollArea, BorderLayout.LINE_END);
        jf.add(BtnPane, BorderLayout.PAGE_END); // adding ButtonPane to Page
End
        jf.add(packetPane, BorderLayout.CENTER); // adding Packets to center
of the page
        jf.add(TDsplyPanel, BorderLayout.PAGE_START); // adding
TimeDisplayPanel to Page Start
        jf.setVisible(true); // everything on the JFrame should be visible

    }// constructor end

    public static void GBN() // creating a method called GBN
    {
        dlmx.addElement("TIMEOUT for packet no: " + base); // adding time out
sentence in JList
        dlmx.addElement("Go-Back-N packets: " + base + "-" + (nextSeqNo - 1));
        sendNew.setVisible(false); // sendNew button is disabled
```



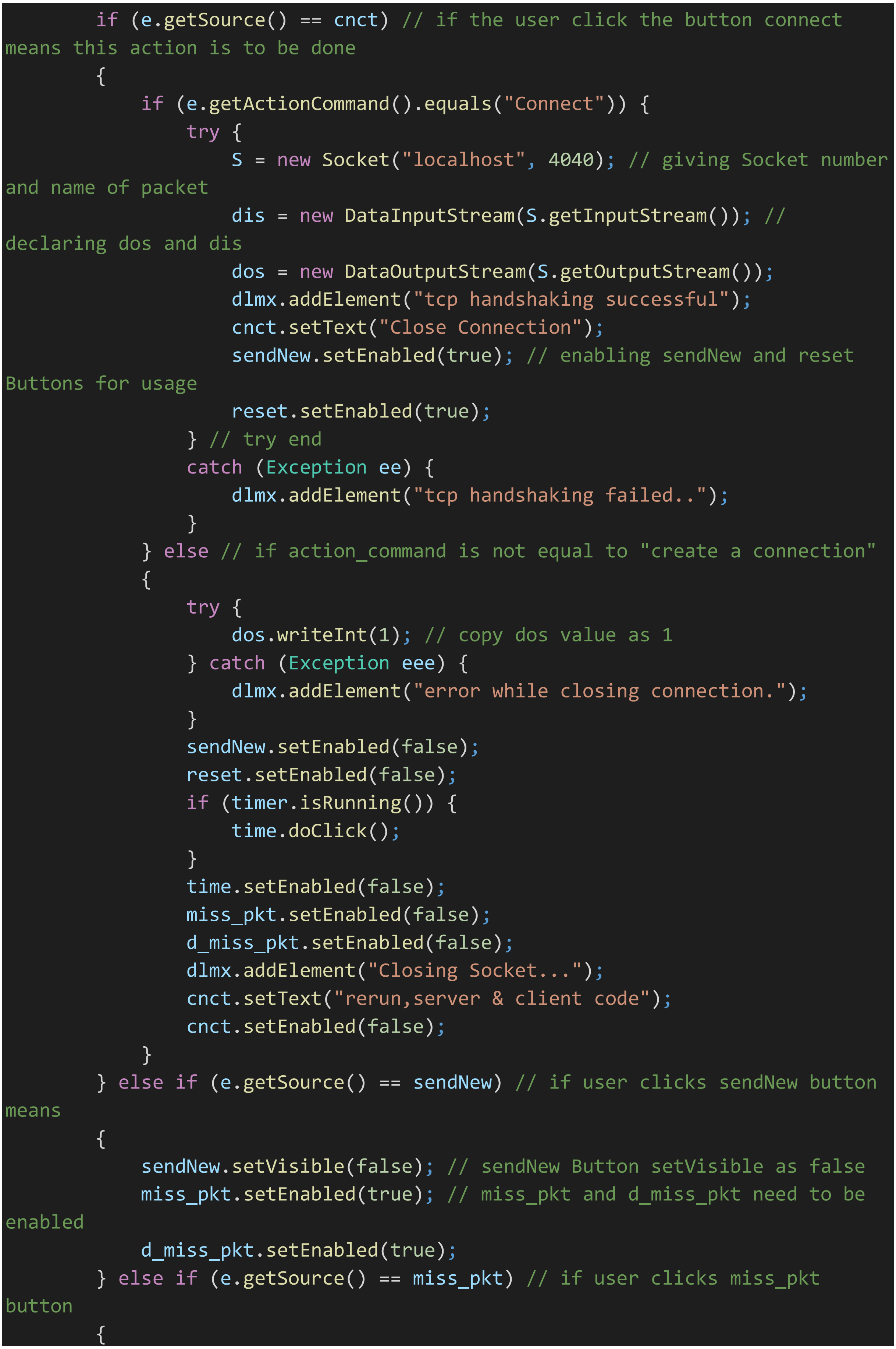
```
        miss_pkt.setEnabled(true); // making miss_pkt and d_miss_pkt enable
for use
        d_miss_pkt.setEnabled(true);
        miss_pkt.setText("Miss Packets[" + base + " -" + (nextSeqNo - 1) + "
]");
        d_miss_pkt.setText("don't Miss Packets[" + base + " -" + (nextSeqNo -
1) + " ]");
        miss_pkt.setVisible(false);
        miss_pkt.setVisible(true);
        d_miss_pkt.setVisible(false);
        d_miss_pkt.setVisible(true);
    }

    public static void implementing_GBN(boolean pktsMissed) // method
implementing_GBN is created with ptsMissed as
                                                            // parameter
    {
        time.doClick();
        dlmx.addElement("Restarting Timer for packet no:" + base); // adding
sentence into JList
        if (pktsMissed == false) {
            try // if packets are not missed means in case of exception (try
will execute) or
                // else catch block executes
            {
                for (int i = base; i <= nextSeqNo - 1; i++) {
                    dos.writeInt(2); // writes an Integer to underlying output
stream as 4bytes
                    dos.writeInt(i);
                }
                dlmx.addElement("packet no: " + base + "-" + (nextSeqNo - 1) +
"had been sent");
            } catch (Exception e9) {
            }
        } else // if packet is missed means it will display that packet is
lost in the network
        {
            dlmx.addElement("packet no: " + base + "-" + (nextSeqNo - 1) + "
got missed in network");
        }
        miss_pkt.setVisible(false);
        d_miss_pkt.setVisible(false);
        miss_pkt.setText("Miss Packet");
        d_miss_pkt.setText("Don't Miss Packet");
        d_miss_pkt.setVisible(true);
        miss_pkt.setVisible(true);

    }
```



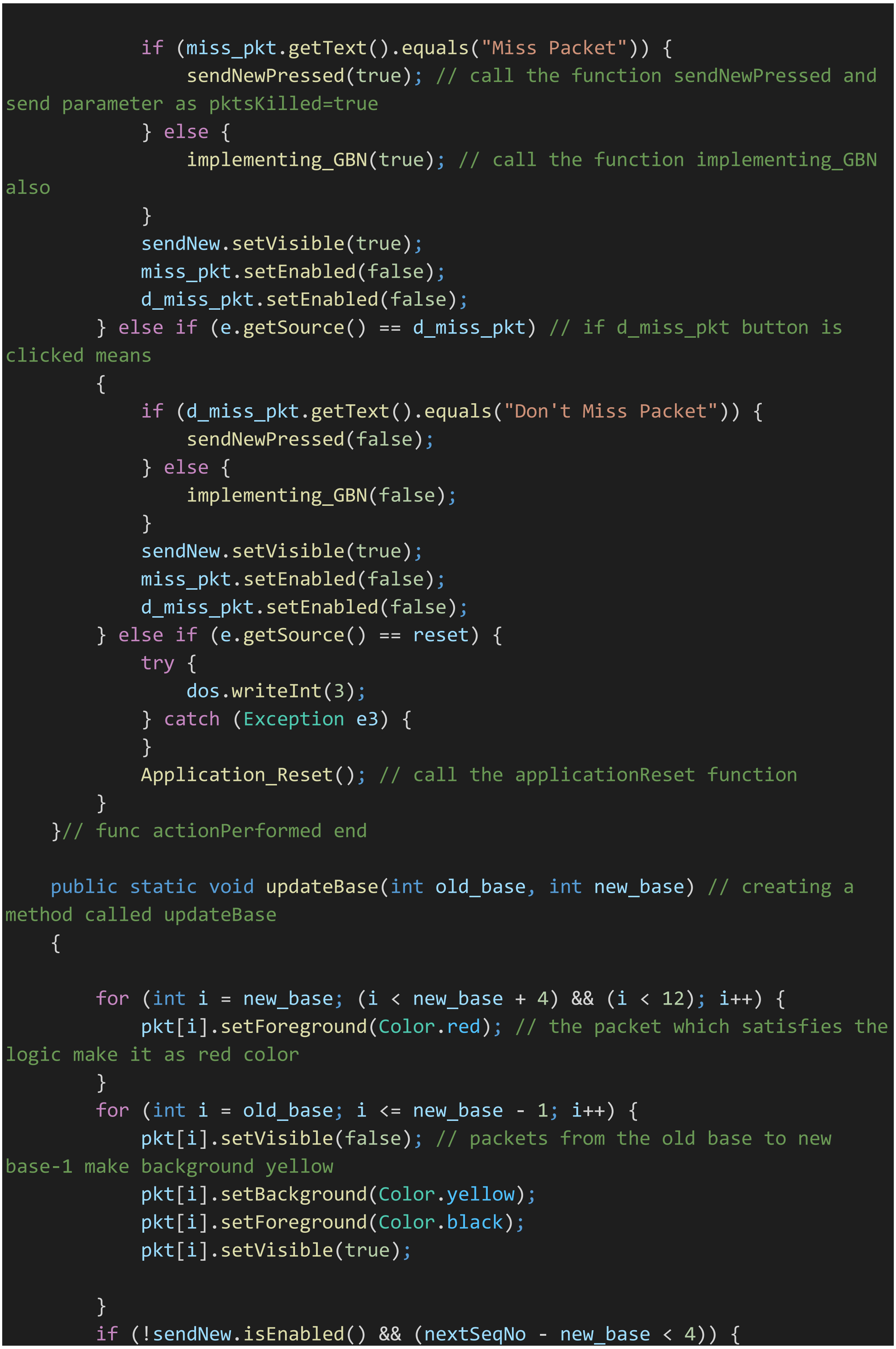
```
    public static void sendNewPressed(boolean pktMissed) {
        if (nextSeqNo < base + 4) {
            if (pktMissed == false) // if packet is not missed in the network
means
            {
                try {
                    dos.writeInt(2);
                    dos.writeInt(nextSeqNo);
                    dlmx.addElement("packet no: " + (nextSeqNo) + "had been
sent"); // shows that the numbered packet is

      // sent
                } catch (Exception e9) {
                    dlmx.addElement("error sending packet no" + (nextSeqNo) +
" ."); // any exception means error in

       // sending message is displayed
                }
            } else {
                dlmx.addElement("packet no" + (nextSeqNo) + " got missed in
network");
            }

            if (base == nextSeqNo) // if base value is equal to nextSeqNo
means enable the timer
            {
                if (!time.isEnabled()) {
                    time.setEnabled(true);
                }
                time.doClick();
                dlmx.addElement("timer started for packet no: " + (nextSeqNo)
+ " .");

            }
            pkt[nextSeqNo].setVisible(false); // initially make next_pkt as
not visible
            pkt[nextSeqNo].setBackground(Color.cyan); // change the
Backgroundcolour to cyan
            pkt[nextSeqNo].setVisible(true); // after changing colour make it
visible
            nextSeqNo++; // increment the nextseqNo
            seqNoLabel.setText("next Sequence no:" + nextSeqNo);
            if (nextSeqNo == base + 4) {
                sendNew.setEnabled(false);
            }
        } else {
```



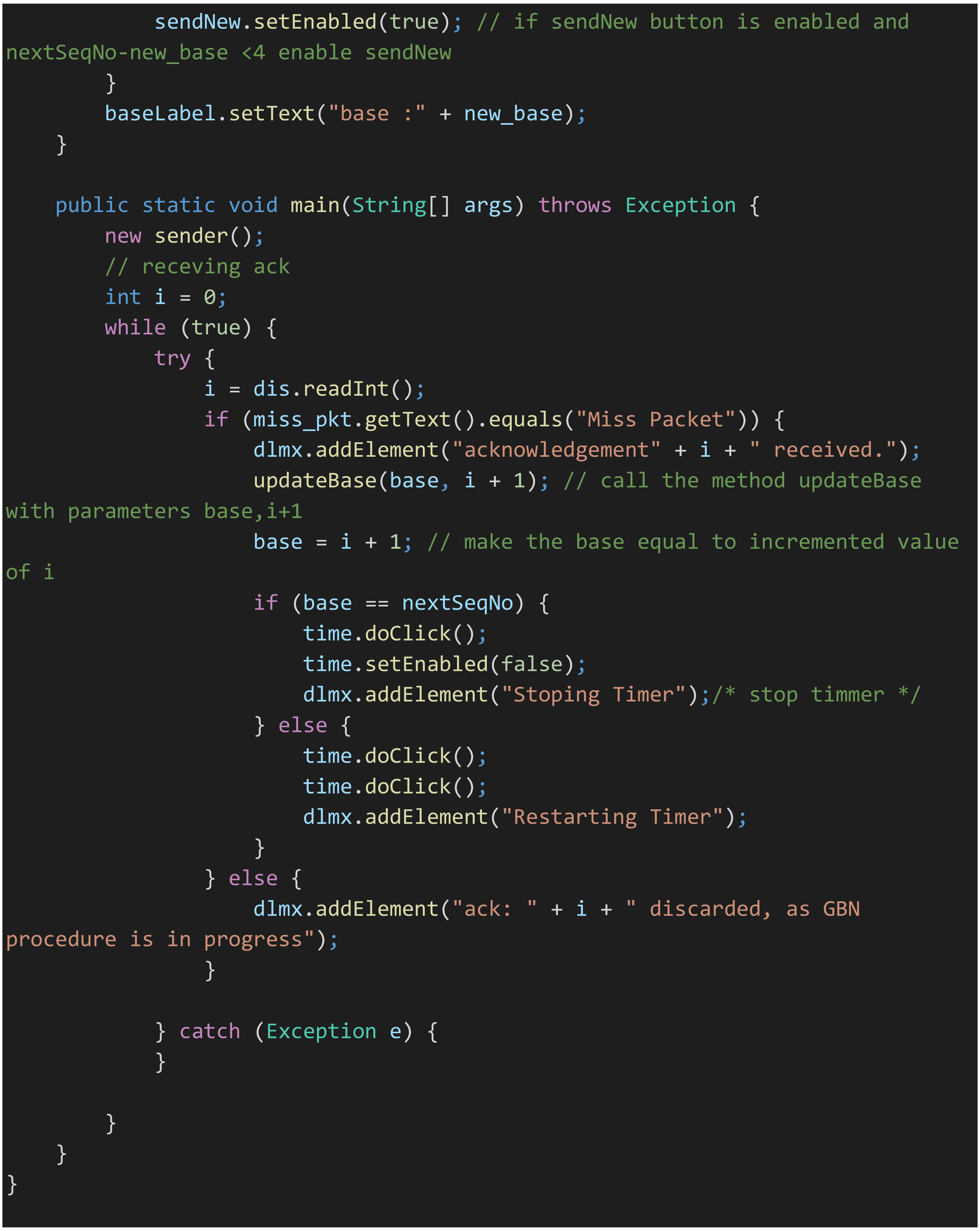
```
            dlmx.addElement("sending request REJECTED-exceeding window
size(4)");
        }
    }// func sendNewPressed end

    public void Application_Reset() // when an Rest button is clicked..this
function is called
    {
        base = 0; // as the operation should starting from the starting make
all base,nextseqNo=0
        nextSeqNo = 0;
        baseLabel.setText("base :" + base);
        seqNoLabel.setText("next Sequence no:" + nextSeqNo);
        dlmx.clear(); // clear the JPane
        if (timer.isRunning()) {
            time.doClick();
        }
        time.setEnabled(false);
        dlmx.addElement("tcp handshaking successful");
        miss_pkt.setVisible(false);
        d_miss_pkt.setVisible(false);
        miss_pkt.setText("Miss Packet");
        d_miss_pkt.setText("Don't Miss Packet");
        miss_pkt.setEnabled(false);
        d_miss_pkt.setEnabled(false);
        miss_pkt.setVisible(true);
        d_miss_pkt.setVisible(true);
        timeDsply.setText("-- -- --");
        sendNew.setEnabled(true);
        sendNew.setVisible(true);
        for (int i = 0; i < 12; i++) // make all packets Background Color as
white and ForeGround Color as black
        {
            pkt[i].setVisible(false);
            pkt[i].setForeground(Color.black);
            pkt[i].setBackground(Color.white);
            pkt[i].setVisible(true);
        }
        for (int z = 0; z < 4; z++) {
            pkt[z].setVisible(false);
            pkt[z].setForeground(Color.red); // as the size of window is 4
...make first 4 packets as red Color
            pkt[z].setVisible(true);
        }
    }

    public void actionPerformed(ActionEvent e) {
```



```
        if (e.getSource() == cnct) // if the user click the button connect
means this action is to be done
        {
            if (e.getActionCommand().equals("Connect")) {
                try {
                    S = new Socket("localhost", 4040); // giving Socket number
and name of packet
                    dis = new DataInputStream(S.getInputStream()); //
declaring dos and dis
                    dos = new DataOutputStream(S.getOutputStream());
                    dlmx.addElement("tcp handshaking successful");
                    cnct.setText("Close Connection");
                    sendNew.setEnabled(true); // enabling sendNew and reset
Buttons for usage
                    reset.setEnabled(true);
                } // try end
                catch (Exception ee) {
                    dlmx.addElement("tcp handshaking failed..");
                }
            } else // if action_command is not equal to "create a connection"
            {
                try {
                    dos.writeInt(1); // copy dos value as 1
                } catch (Exception eee) {
                    dlmx.addElement("error while closing connection.");
                }
                sendNew.setEnabled(false);
                reset.setEnabled(false);
                if (timer.isRunning()) {
                    time.doClick();
                }
                time.setEnabled(false);
                miss_pkt.setEnabled(false);
                d_miss_pkt.setEnabled(false);
                dlmx.addElement("Closing Socket...");
                cnct.setText("rerun,server & client code");
                cnct.setEnabled(false);
            }
        } else if (e.getSource() == sendNew) // if user clicks sendNew button
means
        {
            sendNew.setVisible(false); // sendNew Button setVisible as false
            miss_pkt.setEnabled(true); // miss_pkt and d_miss_pkt need to be
enabled
            d_miss_pkt.setEnabled(true);
        } else if (e.getSource() == miss_pkt) // if user clicks miss_pkt
button
        {
```



```
            if (miss_pkt.getText().equals("Miss Packet")) {
                sendNewPressed(true); // call the function sendNewPressed and send parameter as pktsKilled=true
            } else {
                implementing_GBN(true); // call the function implementing_GBN also
            }
            sendNew.setVisible(true);
            miss_pkt.setEnabled(false);
            d_miss_pkt.setEnabled(false);
        } else if (e.getSource() == d_miss_pkt) // if d_miss_pkt button is clicked means
        {
            if (d_miss_pkt.getText().equals("Don't Miss Packet")) {
                sendNewPressed(false);
            } else {
                implementing_GBN(false);
            }
            sendNew.setVisible(true);
            miss_pkt.setEnabled(false);
            d_miss_pkt.setEnabled(false);
        } else if (e.getSource() == reset) {
            try {
                dos.writeInt(3);
            } catch (Exception e3) {
            }
            Application_Reset(); // call the applicationReset function
        }
    }// func actionPerformed end

    public static void updateBase(int old_base, int new_base) // creating a method called updateBase
    {

        for (int i = new_base; (i < new_base + 4) && (i < 12); i++) {
            pkt[i].setForeground(Color.red); // the packet which satisfies the logic make it as red color
        }
        for (int i = old_base; i <= new_base - 1; i++) {
            pkt[i].setVisible(false); // packets from the old base to new base-1 make background yellow
            pkt[i].setBackground(Color.yellow);
            pkt[i].setForeground(Color.black);
            pkt[i].setVisible(true);

        }
        if (!sendNew.isEnabled() && (nextSeqNo - new_base < 4)) {
```



```
            sendNew.setEnabled(true); // if sendNew button is enabled and
nextSeqNo-new_base <4 enable sendNew
        }
        baseLabel.setText("base :" + new_base);
    }

    public static void main(String[] args) throws Exception {
        new sender();
        // receving ack
        int i = 0;
        while (true) {
            try {
                i = dis.readInt();
                if (miss_pkt.getText().equals("Miss Packet")) {
                    dlmx.addElement("acknowledgement" + i + " received.");
                    updateBase(base, i + 1); // call the method updateBase
with parameters base,i+1
                    base = i + 1; // make the base equal to incremented value
of i
                    if (base == nextSeqNo) {
                        time.doClick();
                        time.setEnabled(false);
                        dlmx.addElement("Stoping Timer");/* stop timmer */
                    } else {
                        time.doClick();
                        time.doClick();
                        dlmx.addElement("Restarting Timer");
                    }
                } else {
                    dlmx.addElement("ack: " + i + " discarded, as GBN
procedure is in progress");
                }

            } catch (Exception e) {
            }

        }
    }
}
```

CHAPTER 4

METHODS INCLUDED IN THE SOURCE CODE

Methods Used in the Sender Side of Go Back N

- **Implenting GBN() :** In this method , it restarts the timer for every packet if a new packet is sent. It has Boolean parameter as pkts_Missed .If it is true means it will show in JList as Packet got missed in the network . If pkts_Missed is made as false means it will show that packet holding that particular seqNo is being sent from the sender.
- **SendNewPressed()** : If the user clicks the button **SendNew** , this method is called. In this method it will check whether the seqNo of that packet and base are exceeding the size of the window. It will also has parameter as Boolean pkts_Missed. If the packet is not exceeding the window Size , it will update in the Display Box called JList . If base value is equal to nextseqNo means enable the time and update the same thing.
- **Application_Reset() :** If the user clicks on the **Reset** Button the main function will call this method . In this method all the base value, nextseqNo value are made to zero and clear all the JList part(Display Box) . If timer is running on the back side means , it stops the timer also. Disable all the buttons except the Connect Button. Initially make all the packets with Background colour as white and ForeGround colour as Black and as the size of the window is 4 , make first 4 packets to BackGround colour as red.
- **Update the Base()** : As soon as the packet is sent and the acknowledgement is received successfully , update the base with the old base in the right most part of the JFrame. Change the colour of the packets as Yellow as they have been already acknowledged.

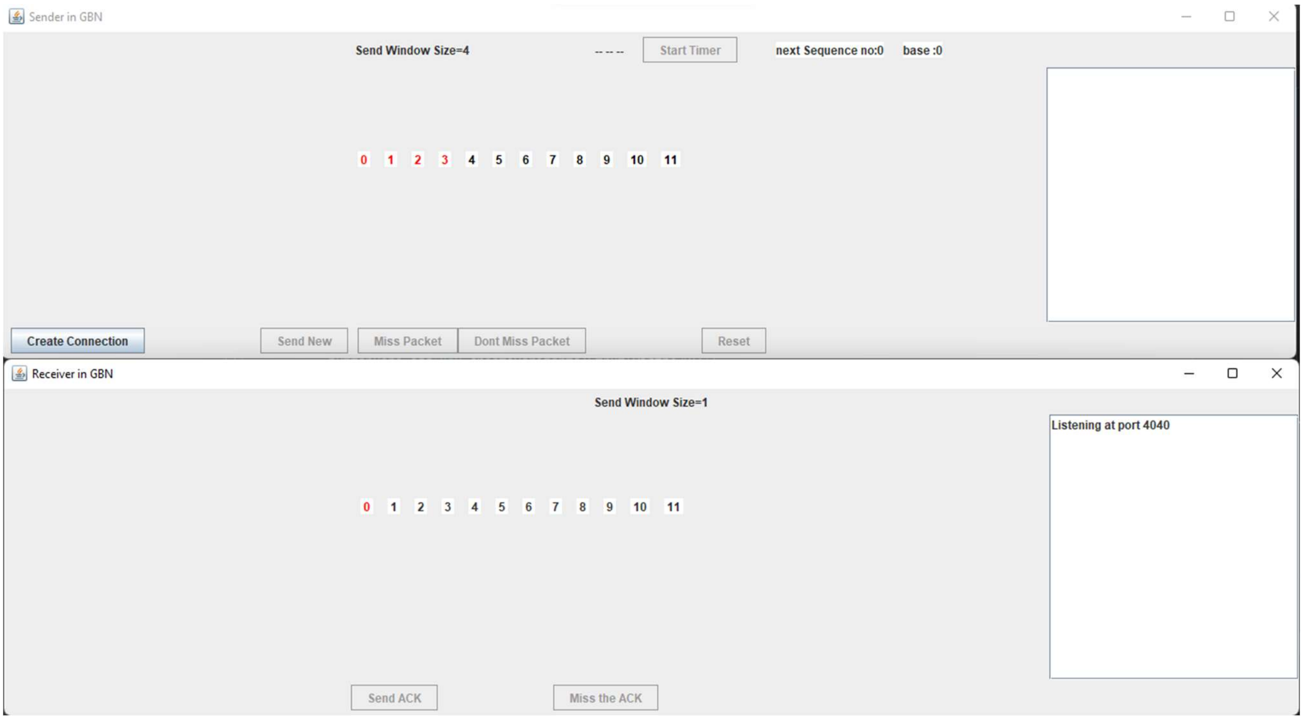
Methods Used in the Receiver Side of Go Back N

- **After_Receiving_Packet() :** In this method as soon as the packet is received, it updates the JList as specific packet with so and so seqNo is received. It changes the colour of that packet to BackGround colour as Green and ForeGround colour as black only. Enable the acknow and d_acknow buttons for sending the acknowledgement to the lastly received packet . If packet is missed in the network means update the JList as Packet is discarded.
- **Reset_Application() :** As soon as the user clicks the reset button in the sender window , necessary changes have to be done . Make the packets with ForeGround colour as Black and BackGround colour as white. Disable the acknow and d_acknow buttons , as there is no need of necessary to use.

CHAPTER 5

SNAPSHOTS

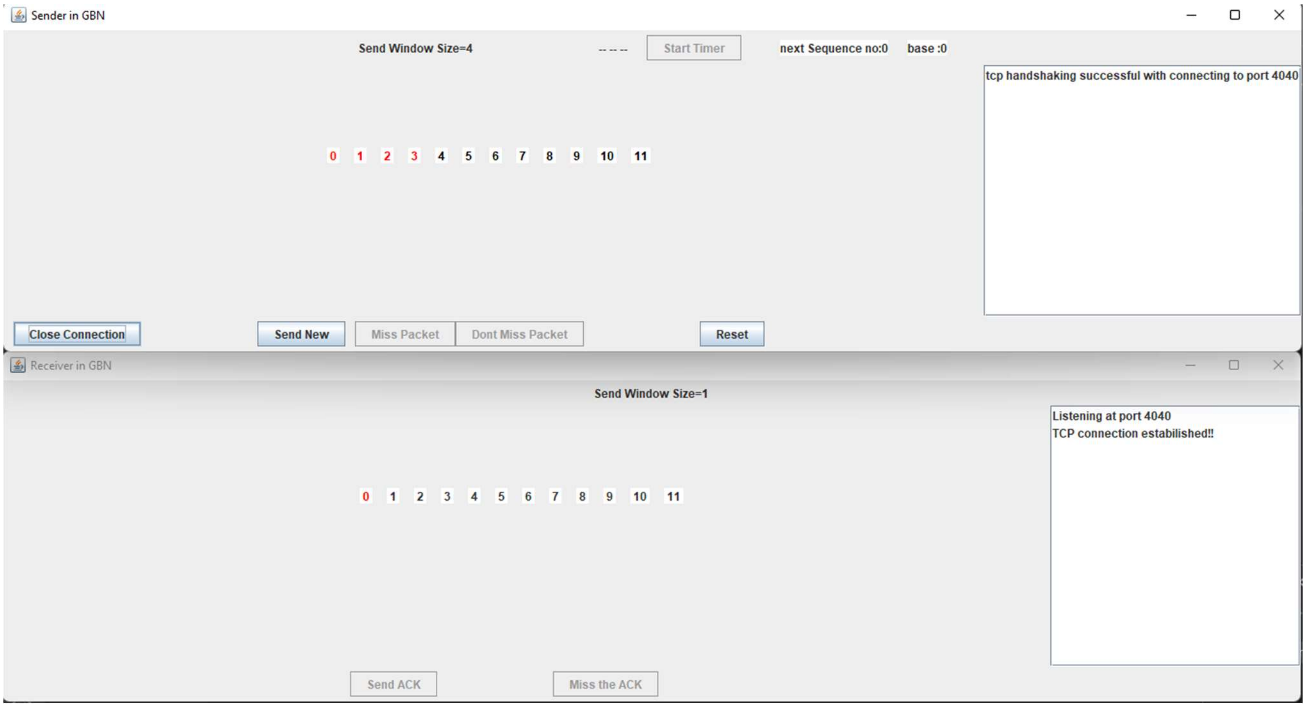
- After the execution of the source code the interface looks like this:



```
Listening at port 4040
```

Fig 5.1: Interface of the sender and receiver after execution

- After creating Connection, it will show "TCP SHANDSHAKING SUCCESSFUL".

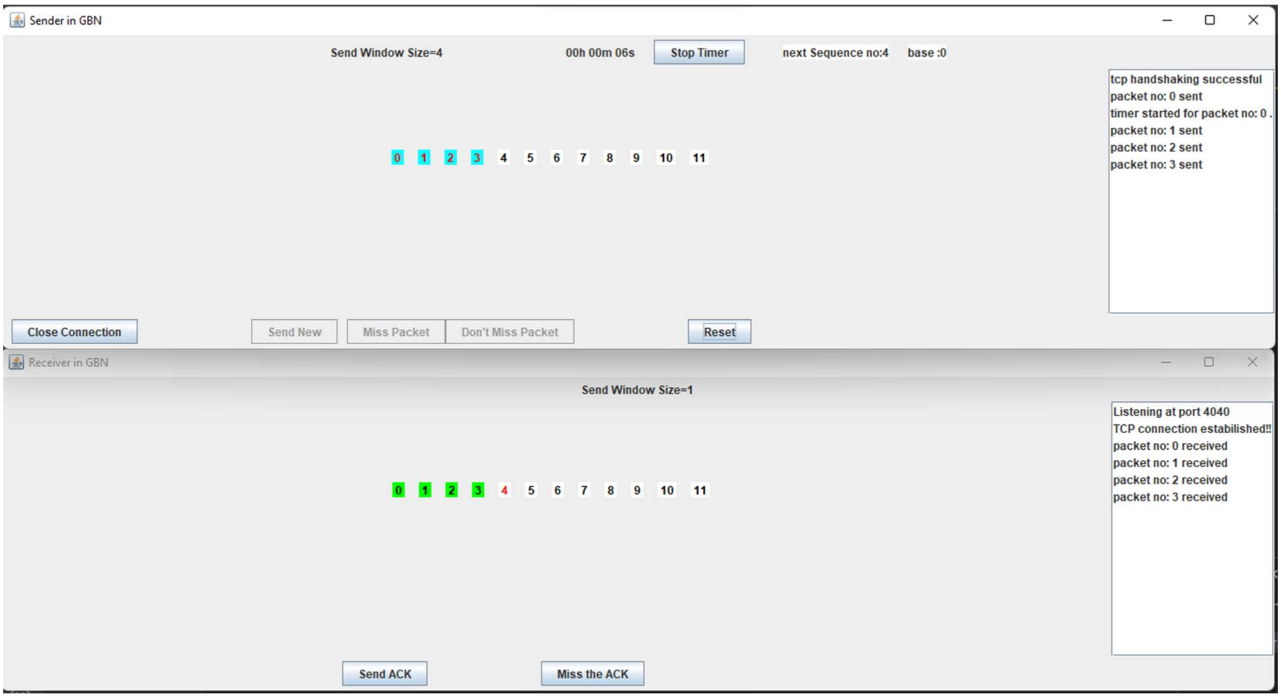


```
tcp handshaking successful with connecting to port 4040
```

```
Listening at port 4040
TCP connection estabilished!!
```

Fig 5.2: After clicking on Create Connection Button

- After the connection with sender, sendNew option is enabled. If we click on that button and send the first 4 packets from 0 to 3



Send Window Size=4 00h 00m 06s next Sequence no:4 base :0

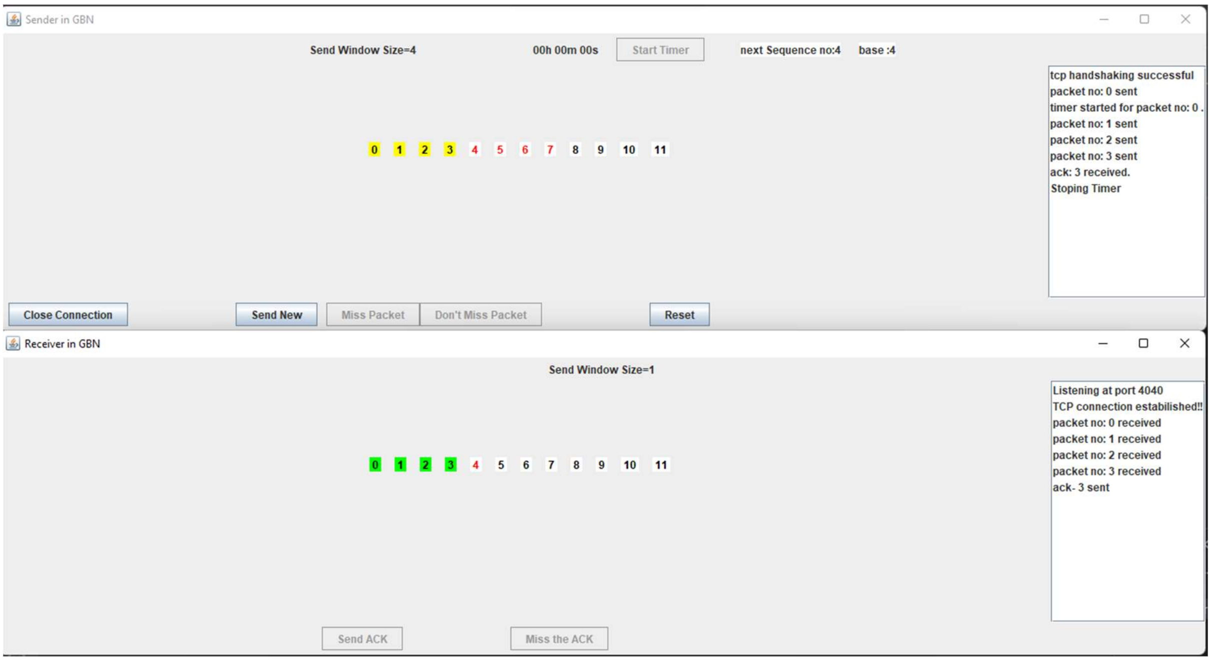
```
tcp handshaking successful
packet no: 0 sent
timer started for packet no: 0 .
packet no: 1 sent
packet no: 2 sent
packet no: 3 sent
```

Send Window Size=1

```
Listening at port 4040
TCP connection estabilished!!
packet no: 0 received
packet no: 1 received
packet no: 2 received
packet no: 3 received
```

Fig 5.3: Sending first 4 packets without missing the packets

- If we click on the “Send Ack” option it will send the Acknowledgment without missing ACK in the network**.**



Send Window Size=4 00h 00m 00s next Sequence no:4 base :4

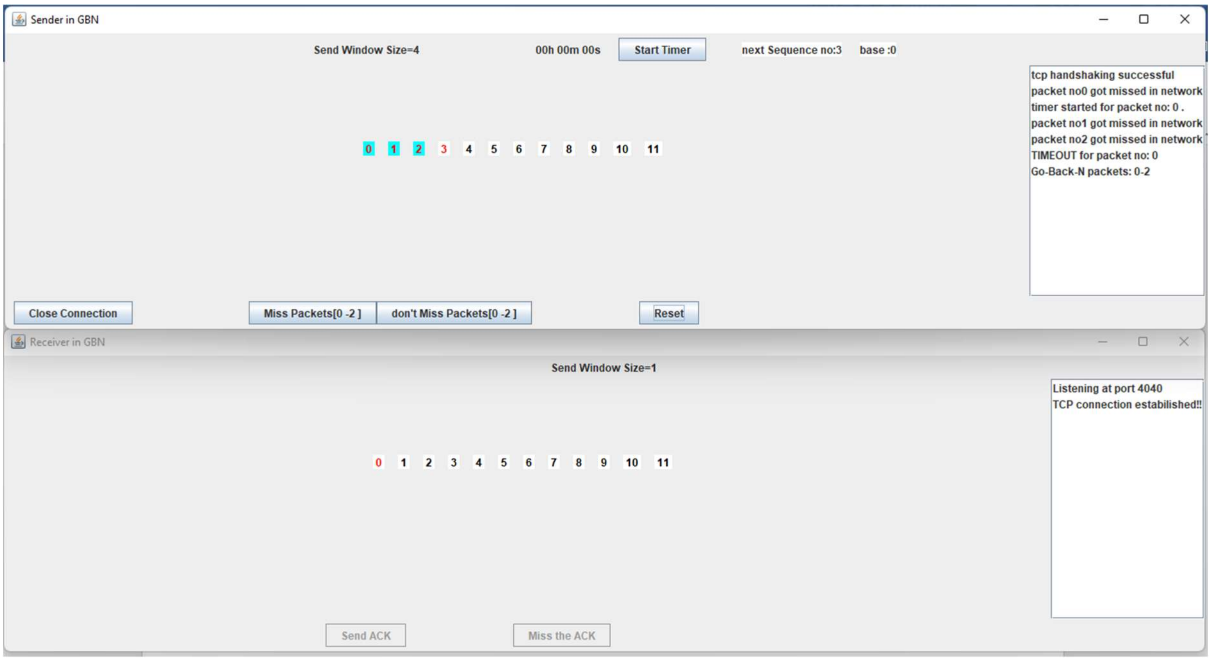
```
tcp handshaking successful
packet no: 0 sent
timer started for packet no: 0 .
packet no: 1 sent
packet no: 2 sent
packet no: 3 sent
ack: 3 received.
Stoping Timer
```

Send Window Size=1

```
Listening at port 4040
TCP connection estabilished!!
packet no: 0 received
packet no: 1 received
packet no: 2 received
packet no: 3 received
ack- 3 sent
```

Fig 5.4: Receiver sends ACK without missing in the network

- If the packets are going to be missed in the network



Send Window Size=4 00h 00m 00s next Sequence no:3 base :0

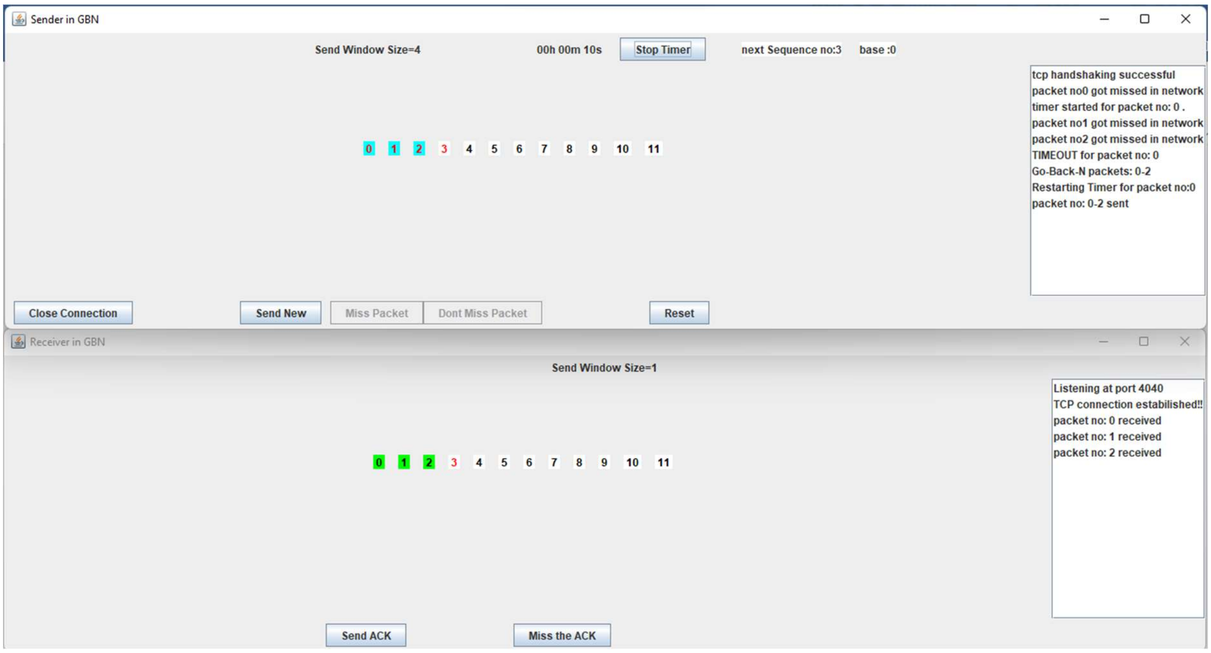
```
tcp handshaking successful
packet no0 got missed in network
timer started for packet no: 0 .
packet no1 got missed in network
packet no2 got missed in network
TIMEOUT for packet no: 0
Go-Back-N packets: 0-2
```

Send Window Size=1

```
Listening at port 4040
TCP connection estabilished!!
```

Fig 5.5: Packets are being lost in the network

- If we click on “Don’t Miss packet[0-2]” ,now the packets will be resent from 0-2 and receiver sends the ACK.



Send Window Size=4 00h 00m 10s next Sequence no:3 base :0

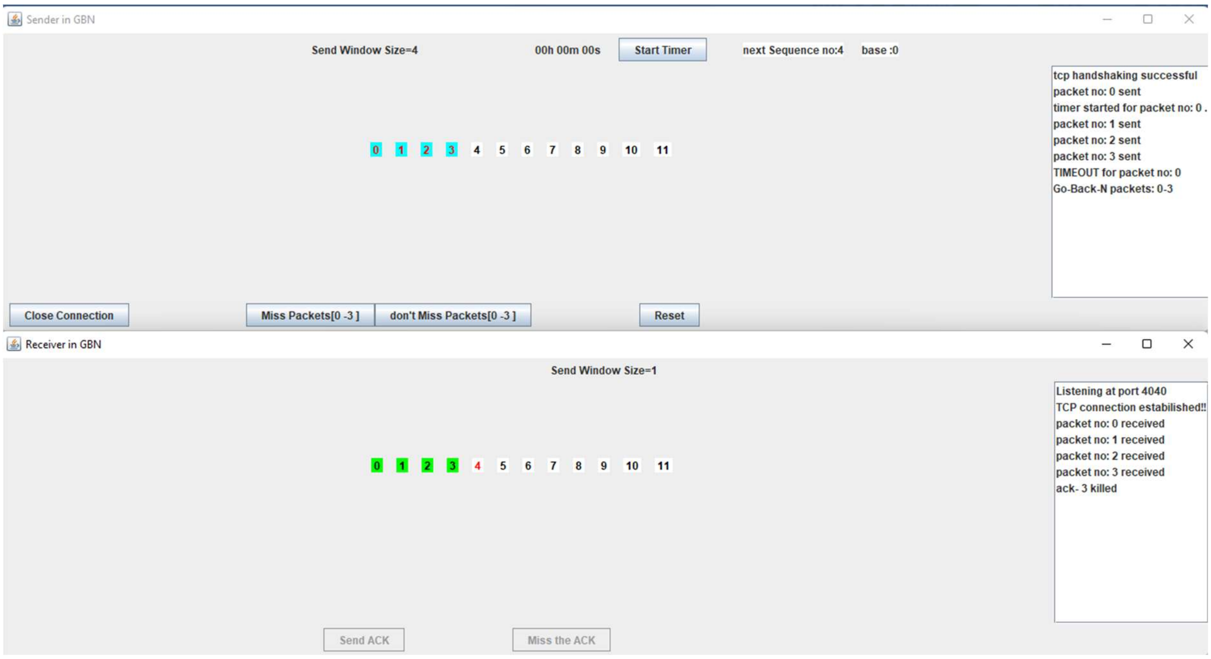
```
tcp handshaking successful
packet no0 got missed in network
timer started for packet no: 0 .
packet no1 got missed in network
packet no2 got missed in network
TIMEOUT for packet no: 0
Go-Back-N packets: 0-2
Restarting Timer for packet no:0
packet no: 0-2 sent
```

Send Window Size=1

```
Listening at port 4040
TCP connection estabilished!!
packet no: 0 received
packet no: 1 received
packet no: 2 received
```

Fig 5.6: If sender correctly retransmits the packets which are lost

- If receiver clicks on “Miss the ACK” means sender waits until the time 15sec and sender thinks that those packets may be lost in the internet. So again, it will retransmit.

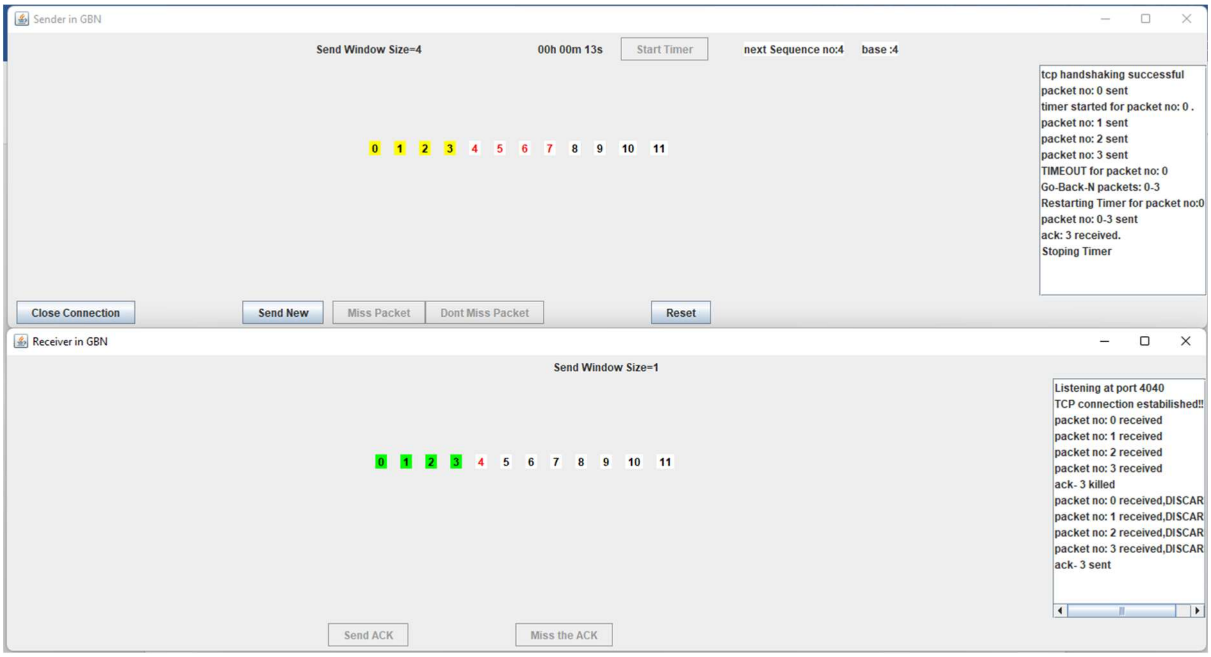


```
tcp handshaking successful
packet no: 0 sent
timer started for packet no: 0 .
packet no: 1 sent
packet no: 2 sent
packet no: 3 sent
TIMEOUT for packet no: 0
Go-Back-N packets: 0-3
```

```
Listening at port 4040
TCP connection estabilished!!
packet no: 0 received
packet no: 1 received
packet no: 2 received
packet no: 3 received
ack- 3 killed
```

Fig 5.7: If receiver missed the ACK in the network

- Finally again if the receiver sends ACK means the packets changes the colour, that means packets are being transmitted and receiver has already been acknowledged.



```
tcp handshaking successful
packet no: 0 sent
timer started for packet no: 0 .
packet no: 1 sent
packet no: 2 sent
packet no: 3 sent
TIMEOUT for packet no: 0
Go-Back-N packets: 0-3
Restarting Timer for packet no:0
packet no: 0-3 sent
ack: 3 received.
Stoping Timer
```

```
Listening at port 4040
TCP connection estabilished!!
packet no: 0 received
packet no: 1 received
packet no: 2 received
packet no: 3 received
ack- 3 killed
packet no: 0 received,DISCAR
packet no: 1 received,DISCAR
packet no: 2 received,DISCAR
packet no: 3 received,DISCAR
ack- 3 sent
```

Fig 5.8: If the receiver sends the ACK without missing in the network

CHAPTER 6

COMPARISON OF GO BACK N AND SELECTIVE REPEAT PROTOCOLS

GO BACK N PROTOCOL	**SELECTIVE REPEAT PROTOCOL**
1.GBN protocol is a sliding window protocol in which sender sends N packets at a time without receiving the acknowledgment.	**1.**Selective Repeat Protocol is also another type of protocol in which sends sends N packets at a time (N refers top size of the window).
2.GBN will not acknowledge if there is a gap in the middle of the packets. It asks sender to retransmit the packets again from where is packet is lost to the nextSeqNum-1 packet	**2**. SR sends acknowledgement for each and every packet. So that if a particular packet is lost , it easily identifies the packet and retransmit that packet alone
3.Sender only has timer for the base of that packet, that is it maintains timer only for 1st packet of that window.	**3.**Sender has timer for each packet which makes the sender side as well as receiver side a complex system.
4.The Bandwidth of the unreliable channel increases as huge number of packets are retransmitted again and again.	**4.** The Bandwidth of unreliable channel is very less when compared to the GBN, because it only asks the sender to retransmit only that packet alone.
5.GBN doesn't have buffer space to store the packets. So it doesn't acknowledge if there is a gap. It asks again to retransmit.	**5.** SR has buffer to store the packets if there is a gap in the middle of the packets. As sender retransmit, receiver keeps packets in order and transmit data to the next layer.
6.Time taken to transmit the packets from sender to receiver takes more time and several retransmissions.	**6.** Time taken to transmit the data packets from the sender to receiver takes less time when compared to the GBN as less retransmissions.
7. Efficiency of GBN is N/(1+2*a)	**7.** Efficiency of SR is N/(1+2*a)
8. Receiver window size is 1 in GBN	**8.** Receiver window size is N in the SR
9. In the GBN, neither at sender nor at receiver need sorting	**9.** In SR, receiver side needs sorting of the frames
10. Out of order packets are not accepted. Simply it discards at receiver	**10.** Out of order packets are accepted in SR protocol.

CONCLUSION AND FUTURE PLANS

CONCLUSION:

- In this work we analysed the transmission of packets from sender to receiver in present scenario and implemented the same transmission of GBN protocol for simple understanding using JAVA GUI environment.
- It is assumed that the packets are received to the receiver side in the same order from the sender side which made the coding part much simpler to implement.
- IN GBN ARQ protocol, considering that the packets can be transmitted continuously without receiving acknowledgement for previous packets, the system is categorized into different cases: sender sends packets without missing in the network and receiver sends ACK without missing in the network and another case is sender sends with missing packets and last case is receiver mis the ACK in the network.
- Above all cases have been implemented using the source code in GBN.

FUTURE PLANS:

- Throughout the implementation of the process, we assumed that packets are being receiving the same order of sender side. But some packets may receive very fast but some packets may be taking time to arrive at the receiver.
- So to reorder the packets according to the Sequence number makes complex coding part which may include high logic and more libraries to implement.
- This is achieved by adding additional field in the packet header. That is done by the network Layer.
- Even we can include number of times the packet being retransmitted, and bandwidth of unreliable channel can also been observed while in the process of compilation.
- These are about the Future Plans of implementation of Go Back N protocol.