```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.ArrayList;
import java.util.Collections;

public class MyReceiver {

	public static final double LOST_PACK_PROBABILITY = 0.1;

	public static void main(String[] args) {

		DatagramSocket socket = null;
		int portNumber = 0;
		if (args.length == 1) {

			portNumber = Integer.parseInt(args[0]);

		} else {
			System.out.println("Invalid Parameters argv[0] - PortNumber/n");
		}

		byte[] incomingData = new byte[1024];
		try {

			socket = new DatagramSocket(portNumber);
			System.out.println("Reciver Side is Ready to Accept Packets at PortNumber: " + portNumber
					+ "\n");

			DatagramPacket initialPacket = new DatagramPacket(incomingData, incomingData.length);
			socket.receive(initialPacket);
			byte[] data1 = initialPacket.getData();
			ByteArrayInputStream inInitial = new ByteArrayInputStream(data1);
			ObjectInputStream isInitial = new ObjectInputStream(inInitial);
			InitiateTransfer initiateTransfer = (InitiateTransfer) isInitial.readObject();
			System.out.println(
					"Initial configuration Recieved = " + initiateTransfer.toString() + "\n");

			int type = initiateTransfer.getType();
			InetAddress IPAddress = initialPacket.getAddress();
			int port = initialPacket.getPort();
			initiateTransfer.setType(100);

			ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
			ObjectOutputStream os = new ObjectOutputStream(outputStream);
			os.writeObject(initiateTransfer);

			byte[] replyByte = outputStream.toByteArray();
			DatagramPacket replyPacket = new DatagramPacket(replyByte, replyByte.length, IPAddress, port);
			socket.send(replyPacket);

			if (type == 0) {
				initiateTransfer.setType(0);
				gbnTransfer(socket, initiateTransfer);
			} else {
				initiateTransfer.setType(1);
				srtransfer(socket, initiateTransfer);
			}

		} catch (Exception e) {

			e.printStackTrace();
		}

	}

	private static void srtransfer(DatagramSocket socket, InitiateTransfer initiateTransfer)
			throws IOException, ClassNotFoundException {

		ArrayList<SegmentData> received = new ArrayList<>();
		boolean end = false;
		int waitingFor = 0;
		byte[] incomingData = new byte[1024];

		ArrayList<SegmentData> buffer = new ArrayList<>();

		while (!end) {

			DatagramPacket incomingPacket = new DatagramPacket(incomingData, incomingData.length);
			socket.receive(incomingPacket);
			InetAddress IPAddress = incomingPacket.getAddress();
			int port = incomingPacket.getPort();
			byte[] data = incomingPacket.getData();
			ByteArrayInputStream in = new ByteArrayInputStream(data);
			ObjectInputStream is = new ObjectInputStream(in);
			SegmentData segmentData = (SegmentData) is.readObject();

			char ch = segmentData.getPayLoad();
			int hashCode = ("" + ch).hashCode();
			boolean checkSum = (hashCode == segmentData.getCheckSum());

			if (segmentData.getSeqNum() == waitingFor && segmentData.isLast() && checkSum) {

				waitingFor++;
				received.add(segmentData);
				int value = sendData(segmentData, waitingFor, socket, IPAddress, port, false);
				if (value < waitingFor) {
					waitingFor = value;
					int length = received.size();
					System.out.println("Packet " + (waitingFor) + "");
					System.out.println("Packet Lost\n");
					received.remove(length - 1);
					end = false;

				} else {

					System.out.println("Last packet received\n");
					end = true;
				}

			}

			else if (segmentData.getSeqNum() == waitingFor && checkSum && buffer.size() > 0) {

				received.add(segmentData);
				waitingFor++;
				int value = sendData(segmentData, waitingFor, socket, IPAddress, port, false);
				if (value < waitingFor) {
					waitingFor = value;
					int length = received.size();
					System.out.println("----------Packet " + (waitingFor) + " lost in the Transmission\n");
					//System.out.println("!!!!!!!!!!!!!!!!!!!!!!Packet Lost!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n");
					received.remove(length - 1);

				} else {
					ArrayList<SegmentData> temp = new ArrayList<>();
					temp.addAll(buffer);
					int count = 0;
					for (int i = 0; i < temp.size(); i++) {
						if (!(waitingFor == temp.get(i).getSeqNum())) {

							break;

						} else {
							waitingFor++;
							count++;
							System.out.println(
									"Packet " + buffer.get(i).getSeqNum() + " delivered to Application From Buffer\n");
						}
					}
					buffer = new ArrayList<>();
					for (int j = 0; j < temp.size(); j++) {
						if (j < count) {
							continue;
						}
						buffer.add(temp.get(j));
					}
					if (waitingFor == initiateTransfer.getNumPackets()) {
						end = true;
					}

				}

			}

			else if (segmentData.getSeqNum() == waitingFor && checkSum && buffer.size() == 0) {
				received.add(segmentData);
				waitingFor++;
				int value = sendData(segmentData, waitingFor, socket, IPAddress, port, false);
				if (value < waitingFor) {
					waitingFor = value;
					int length = received.size();
					System.out.println("Packet " + (waitingFor) + " lost in the Transmission\n");
					//System.out.println("!!!!!!!!!!!!!!!!!!!!!!Packet Lost!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n");
					received.remove(length - 1);

				}

				else {

				}

			}

			else if (segmentData.getSeqNum() > waitingFor && checkSum) {

				sendData(segmentData, waitingFor, socket, IPAddress, port, true);
				System.out.println("Packet " + segmentData.getSeqNum()
						+ "Stored in Buffer\n");
				buffer.add(segmentData);
				Collections.sort(buffer);

			}

			else if (segmentData.getSeqNum() < waitingFor && checkSum) {

				sendData(segmentData, waitingFor, socket, IPAddress, port, true);
				System.out.println(
						"Packet Already Delivered  Sending Duplicate Ack\n");

			}

			else if (!checkSum) {
				System.out.println("Packet " + (segmentData.getSeqNum()) + " received");
				System.out.println("Checksum Error");
				System.out.println("Packet " + segmentData.getSeqNum() + " Discarded\n");
				segmentData.setSeqNum(-1000);
			}

			else {

				System.out.println("Packet " + segmentData.getSeqNum() + " Discarded\n");
				segmentData.setSeqNum(-1000);
			}

		}

	}

	public static int sendData(SegmentData segmentData, int waitingFor, DatagramSocket socket, InetAddress iPAddress,
			int port, boolean b) throws IOException {

		AckData ackData = new AckData();
		ackData.setAckNo(segmentData.getSeqNum() + 1);

		ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
		ObjectOutputStream os = new ObjectOutputStream(outputStream);
		os.writeObject(ackData);

		byte[] replyByte = outputStream.toByteArray();
		DatagramPacket replyPacket = new DatagramPacket(replyByte, replyByte.length, iPAddress, port);

		if ((Math.random() > LOST_PACK_PROBABILITY | b) && segmentData.getSeqNum() != -1000) {
			System.out.println("Packet " + (ackData.getAckNo()-1) + " received");
			String reply = "Sending Acknowledgment for Packet :" + (ackData.getAckNo() - 1)
```

```
                                        + "";
                        System.out.println(reply + "\n");
                        socket.send(replyPacket);
                } else if (segmentData.getSeqNum() != -1000 && !b) {
                        waitingFor--;
                }
                return waitingFor;
        }

        private static void gbnTransfer(DatagramSocket socket, InitiateTransfer initiateTransfer)
                        throws IOException, ClassNotFoundException {

                ArrayList<SegmentData> received = new ArrayList<>();
                boolean end = false;
                int waitingFor = 0;
                byte[] incomingData = new byte[1024];

                while (!end) {
                        DatagramPacket incomingPacket = new DatagramPacket(incomingData, incomingData.length);
                        socket.receive(incomingPacket);
                        byte[] data = incomingPacket.getData();
                        ByteArrayInputStream in = new ByteArrayInputStream(data);
                        ObjectInputStream is = new ObjectInputStream(in);
                        SegmentData segmentData = (SegmentData) is.readObject();
                        System.out.println("Packet Received  =  " + segmentData.getSeqNum() + "\n");

                        char ch = segmentData.getPayLoad();
                        int hashCode = ("" + ch).hashCode();
                        boolean checkSum = (hashCode == segmentData.getCheckSum());

                        if (!checkSum) {
                                System.out.println("Error Occured in the Data\n");
                        }

                        if (segmentData.getSeqNum() == waitingFor && segmentData.isLast() && checkSum) {

                                waitingFor++;
                                received.add(segmentData);
                                System.out.println("Last packet received\n");

                                end = true;

                        } else if (segmentData.getSeqNum() == waitingFor && checkSum) {
                                waitingFor++;
                                received.add(segmentData);
                                // System.out.println("Packed stored ");
                        }

                        else if (!checkSum) {
                                System.out.println("Checksum Error\n");
                                segmentData.setSeqNum(-1000);
                        }

                        else {
                                System.out.println("Packet discarded (not in order)\n");
                                segmentData.setSeqNum(-1000);
                        }

                        InetAddress IPAddress = incomingPacket.getAddress();
                        int port = incomingPacket.getPort();

                        AckData ackData = new AckData();
                        ackData.setAckNo(waitingFor);

                        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
                        ObjectOutputStream os = new ObjectOutputStream(outputStream);
                        os.writeObject(ackData);

                        byte[] replyByte = outputStream.toByteArray();
                        DatagramPacket replyPacket = new DatagramPacket(replyByte, replyByte.length, IPAddress, port);

                        if (Math.random() > LOST_PACK_PROBABILITY && segmentData.getSeqNum() != -1000) {
                                String reply = "Sending Acknowledgment Number :" + ackData.getAckNo()
                                                + "\n";
                                System.out.println(reply + "\n");
                                socket.send(replyPacket);
                        } else if (segmentData.getSeqNum() != -1000) {
                                int length = received.size();
                                System.out.println("Packet Lost\n");
                                received.remove(length - 1);
                                waitingFor--;
                                if (end) {
                                        end = false;
                                }

                        }

                }

        }

}
```