Lab 3 Socket Programming

Implement the Selective Repeat protocol using Java UDP socket API. You need to submit two .java files, one for the sender and one for the receiver. On the sender, you could use the following codes to send a packet:

```
DatagramSocket senderSocket = new DatagramSocket(9877);
InetAddress IPAddress = InetAddress.getByName("localhost");
byte[] sendData = new byte[1024];
DatagramPacket sendPkt = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
senderSocket.send(sendPkt);
…
```

On the receiver, you could use the following codes to receive a packet:

```
DatagramSocket receiverSocket = new DatagramSocket(9876);
byte[] rcvData = new byte[1024];
DatagramPacket rcvPkt = new DatagramPacket(rcvData, rcvData.length);
receiverSocket.receive(rcvPkt);
InetAddress IPAddress = rcvPkt.getAddress();
int port = receivePacket.getPort();
…
```

Example output on the sender would look like this.

```
Enter the window's size on the sender: 4
Enter the maximum sequence number on the sender: 10
Select the packet(s) that will be dropped: 2
Send window's size and maximum seq. number to the receiver
Receive confirmation from the receiver
Packet 0 is sent, window [0*, 1, 2, 3]
Packet 1 is sent, window [0*, 1*, 2, 3]
Packet 2 is sent, window [0*, 1*, 2*, 3]
Packet 3 is sent, window [0*, 1*, 2*, 3*]
Ack 0 is received, window [1*, 2*, 3*, 4]
Packet 4 is sent, window [1*, 2*, 3*, 4*]
Ack 1 is received, window [2*, 3*, 4*, 5]
Packet 5 is sent, window [2*, 3*, 4*, 5*]
Ack 3 is received, window [2*, 3, 4*, 5*]
Ack 4 is received, window [2*, 3, 4, 5*]
Ack 5 is received, window [2*, 3, 4, 5]
Packet 2 times out, resend packet 2
Packet 2 is sent, window [2*, 3, 4, 5]
Ack 2 is received, window [6, 7, 8, 9]
Packet 6 is sent, window [6*, 7, 8, 9]
Packet 7 is sent, window [6*, 7*, 8, 9]
Packet 8 is sent, window [6*, 7*, 8*, 9]
Packet 9 is sent, window [6*, 7*, 8*, 9*]
Ack 6 is received, window [7*, 8*, 9*, -]
Ack 7 is received, window [8*, 9*, -, -]
Ack 8 is received, window [9*, -, -, -]
Ack 9 is received, window [-, -, -, -]
```

Example output on the receiver would look like this.

```
Packet 0 is received, send Ack0, window [1, 2, 3, 4]
Packet 1 is received, send Ack1, window [2, 3, 4, 5]
Packet 3 is received, send Ack3, window [2, 3#, 4, 5]
Packet 4 is received, send Ack4, window [2, 3#, 4#, 5]
Packet 5 is received, send Ack5, window [2, 3#, 4#, 5#]
Packet 2 is received, send Ack2, window [6, 7, 8, 9]
Packet 6 is received, send Ack6, window [7, 8, 9, -]
Packet 7 is received, send Ack7, window [8, 9, -, -]
Packet 8 is received, send Ack8, window [9, -, -, -]
Packet 9 is received, send Ack9, window [-, -, -, -]
```