

Ex. No : 2b	Implementation of the Sliding Window Protocol
Date :	

Aim :-

To implement and execute sliding window protocol using java.

Algorithm :-**Sender :**

Step 1 : Start the program.

Step 2 : Implement .net and other necessary packages.

Step 3 : Declare objects for input/output and socket to receive the frame and send acknowledgment.

Step 4 : Wait for the client to establish connection.

Step 5 : Receive the frame one by one from the socket and return the frame number as acknowledgment within the thread sleep time.

Step 6 : Send acknowledgment for each receiving frame and continue the process until acknowledgment for all frames are sent.

Step 7 : Receive the acknowledgment of last frame.

Step 8 : Stop the program.

Receiver :

Step 1 : Start the program.

Step 2 : Import .net and other necessary packages.

Step 3 : Create objects for server socket, socket to send the frames and display the server address when the server is connected.

Step 4 : Get the number of frames to be sent, from the user.

Step 5 : Send the first frame to server using the socket.

Step 6 : When one frame is sent, wait for the acknowledgment from the receiver for the previous frame.

Programs :-**Sender.java**

```
import java.util.LinkedList;
import java.util.Queue;
public class Sender {
    private int windowSize;
```

```
private Queue<Integer> window;
private int totalPackets;
private int nextSeqNum;
public Sender(int windowSize, int totalPackets) {
    this.windowSize = windowSize;
    this.totalPackets = totalPackets;
    this.window = new LinkedList<>();
    this.nextSeqNum = 0;
}
public void startTransmission(Receiver receiver) {
    while (nextSeqNum < totalPackets || !window.isEmpty()) {
        while (window.size() < windowSize && nextSeqNum < totalPackets) {
            System.out.println("Sender: Sending packet with sequence number " + nextSeqNum);
            window.add(nextSeqNum);
            receiver.receivePacket(nextSeqNum);
            nextSeqNum++;
        }
        receiver.acknowledge(window);
        window.clear();
    }
}
```

Receiver1.java

```
import java.util.Queue;
public class Receiver {
    public void receivePacket(int packet) {
        System.out.println("Receiver: Received packet with sequence number " + packet);
    }
    public void acknowledge(Queue<Integer> window) {
        for (Integer seqNum : window) {
            System.out.println("Receiver: Acknowledging packet with sequence number " + seqNum);
        }
    }
}
```

Main.java

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("717823P141");

        System.out.print("Enter the window size: ");

        int windowSize = scanner.nextInt();

        System.out.print("Enter the total number of packets: ");

        int totalPackets = scanner.nextInt();

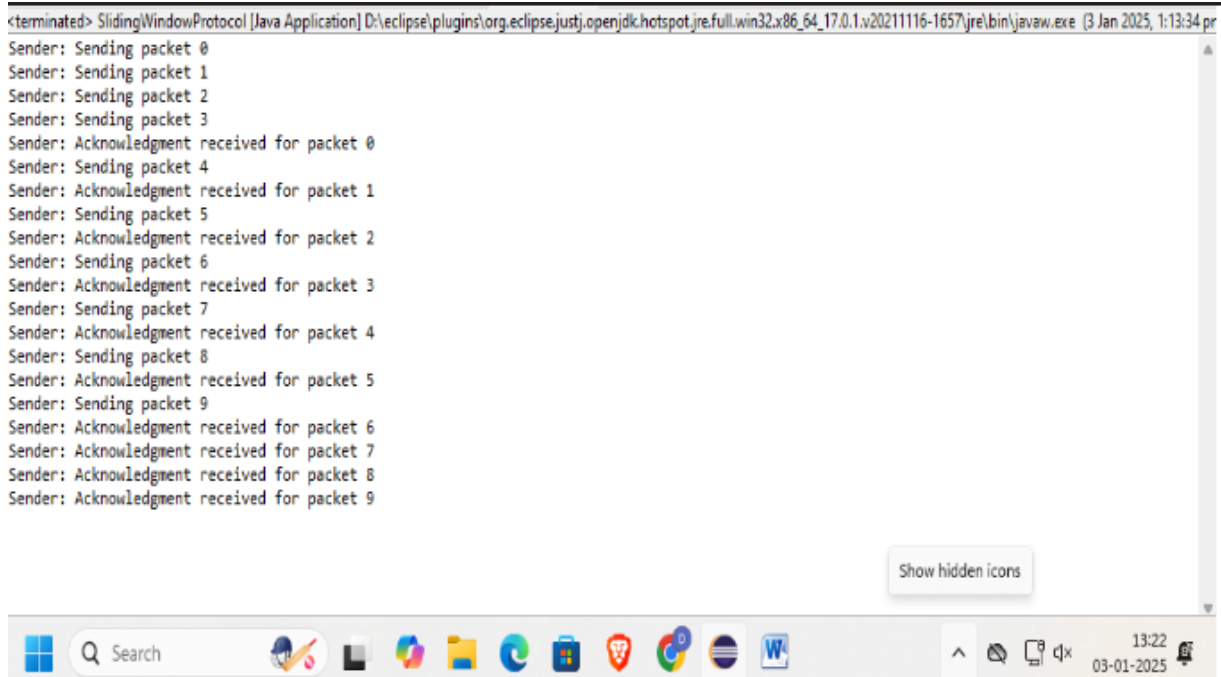
        Sender sender = new Sender1(windowSize, totalPackets);

        Receiver receiver = new Receiver1();

        sender.startTransmission(receiver);

        scanner.close();

    }}
}
```

Output :-

```
<terminated> SlidingWindowProtocol [Java Application] D:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (3 Jan 2025, 1:13:34 pm)
Sender: Sending packet 0
Sender: Sending packet 1
Sender: Sending packet 2
Sender: Sending packet 3
Sender: Acknowledgment received for packet 0
Sender: Sending packet 4
Sender: Acknowledgment received for packet 1
Sender: Sending packet 5
Sender: Acknowledgment received for packet 2
Sender: Sending packet 6
Sender: Acknowledgment received for packet 3
Sender: Sending packet 7
Sender: Acknowledgment received for packet 4
Sender: Sending packet 8
Sender: Acknowledgment received for packet 5
Sender: Sending packet 9
Sender: Acknowledgment received for packet 6
Sender: Acknowledgment received for packet 7
Sender: Acknowledgment received for packet 8
Sender: Acknowledgment received for packet 9
```

Result :-

Thus, the program for sliding window protocol was implemented and executed successfully.