AIM
To implement Sliding Window Protocol and Stop and Wait Protocol using Java.

(a) SLIDING WINDOW PROTOCOL
ALGORITHM
Step 1:Start the program.
Step 2:Get the frame size from the user
Step 3:To create the frame based on the user request.
Step 4:To send frames to server from the client side.
Step 5:If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.
Step 6:Stop the program

PROGRAM
Sender:

```
import java.net.*;
import java.io.*;
import java.rmi.*;
public class slidsender
{
public static void main(String a[])throws Exception

{
ServerSocket ser=new ServerSocket(10);
Socket s=ser.accept();
DataInputStream in=new DataInputStream(System.in);
DataInputStream in1=new DataInputStream(s.getInputStream());
String sbuff[]=new String[8];
PrintStream p;
int sptr=0,sws=8,nf,ano,i;
String ch;
do
{
p=new PrintStream(s.getOutputStream());
System.out.print("Enter the no. of frames : ");
nf=Integer.parseInt(in.readLine());
p.println(nf);
if(nf<=sws-1)
{
System.out.println("Enter "+nf+" Messages to be send\n");
for(i=1;i<=nf;i++)
{
sbuff[sptr]=in.readLine();
p.println(sbuff[sptr]);
sptr=++sptr%8;
}
sws-=nf;
System.out.print("Acknowledgment received");
ano=Integer.parseInt(in1.readLine());
System.out.println(" for "+ano+" frames");
sws+=nf;
}
else
{
System.out.println("The no. of frames exceeds window size");
```

```
break;
}
System.out.print("\nDo you wants to send some more frames : ");
ch=in.readLine(); p.println(ch);
}
while(ch.equals("yes"));
s.close();
}
}
```

Receiver:

```
import java.net.*;
import java.io.*;
class slidreceiver
{
public static void main(String a[])throws Exception
{
Socket s=new Socket(InetAddress.getLocalHost(),10);
DataInputStream in=new DataInputStream(s.getInputStream());
PrintStream p=new PrintStream(s.getOutputStream());
int i=0,rptr=-1,nf,rws=8;
String rbuf[]=new String[8];
String ch; System.out.println();
do
{
nf=Integer.parseInt(in.readLine());
if(nf<=rws-1)
{
for(i=1;i<=nf;i++)
{
rptr=++rptr%8;
rbuf[rptr]=in.readLine();
System.out.println("The received Frame " +rptr+" is : "+rbuf[rptr]);
}
rws-=nf;
System.out.println("\nAcknowledgment sent\n");
p.println(rptr+1); rws+=nf; }
else
break;
ch=in.readLine();
}
while(ch.equals("yes"));
}
}
```

OUTPUT

Sender:

```
Enter the no. of frames : 4
Enter 4 Messages to be send
hi
how r u
i am fine
how is evryone
Acknowledgment received for 4 frames
Do you wants to send some more frames : no
```

Receiver:

The received Frame 0 is : hi
The received Frame 1 is : how r u
The received Frame 2 is : i am fine
The received Frame 3 is : how is everyone

(b)STOP AND WAIT PROTOCOL
ALGORITHM
Step 1: Start.
Step 2 : Invoke the classes ObjectOutputStream and ObjectInputStream with an object to get the input and to project the output between the sender and receiver
Step 3: Create the socket class for both sender and receiver
Step 4: Split the messages sent from the sender and write on the receiver using writeobj
Step 5: the packets are created by partitioning the messages
Step 6: Terminate the program
PROGRAM
Sender:

```
import java.io.*;
import java.net.*;
public class Sender{
Socket sender;
ObjectOutputStream out;
ObjectInputStream in;
String packet,ack,str, msg;
int n,i=0,sequence=0;
Sender(){}
public void run(){
 try{
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Waiting for Connection....");
 sender = new Socket("localhost",2004);
sequence=0;

out=new ObjectOutputStream(sender.getOutputStream());
out.flush();
 in=new ObjectInputStream(sender.getInputStream());
 str=(String)in.readObject();
 System.out.println("receiver     > "+str);
System.out.println("Enter the data to send....");
packet=br.readLine();
n=packet.length();
do{
try{
 if(i<n){
msg=String.valueOf(sequence);
msg=msg.concat(packet.substring(i,i+1));
}
else if(i==n){
 msg="end";out.writeObject(msg);break;
}
out.writeObject(msg);


sequence=(sequence==0)?1:0;
out.flush();
System.out.println("data sent>"+msg);
```

```
ack=(String)in.readObject();
System.out.println("waiting for ack.....\n\n");
if(ack.equals(String.valueOf(sequence))){
i++;
System.out.println("receiver   >  "+" packet recieved\n\n");
}
else{
System.out.println("Time out resending data....\n\n");
sequence=(sequence==0)?1:0;
}
}catch(Exception e){}
}while(i<n+1);
System.out.println("All data sent. exiting.");
}catch(Exception e){}
finally{
try{
in.close();
out.close();
sender.close();
}
catch(Exception e){}
}
}
public static void main(String args[]){
Sender s=new Sender();
s.run();
}
}
```

Receiver:

```
import java.io.*;
import java.net.*;
public class Receiver{
ServerSocket reciever;
Socket connection=null;
ObjectOutputStream out;
ObjectInputStream in;
String packet,ack,data="";
int i=0,sequence=0;
Reciever(){}
public void run(){
try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
reciever = new ServerSocket(2004,10);
System.out.println("waiting for connection...");
connection=reciever.accept();
sequence=0;
System.out.println("Connection established   :");
out=new ObjectOutputStream(connection.getOutputStream());
out.flush();
in=new ObjectInputStream(connection.getInputStream());
out.writeObject("connected    .");
do{
try{
packet=(String)in.readObject();
if(Integer.valueOf(packet.substring(0,1))==sequence){
```

```
data+=packet.substring(1);
sequence=(sequence==0)?1:0;
System.out.println("\n\nreceiver >"+packet);
}
else
{
System.out.println("\n\nreceiver>"+packet +"   duplicate data");
}
if(i<3){
out.writeObject(String.valueOf(sequence));i++;
}
else{
out.writeObject(String.valueOf((sequence+1)%2));
i=0;
}
}
catch(Exception e){}
}while(!packet.equals("end"));
System.out.println("Data recived="+data);
out.writeObject("connection ended    .");
}
catch(Exception e){}
finally{
try{
in.close();
out.close();
reciever.close();
}
catch(Exception e){}
}
}
public static void main(String args[]){
Reciever s=new Reciever();
while(true){
s.run();
}}}
```

SAMPLE OUTPUT

Sender:

Waiting for Connection....
reciver > connected .
Enter the data to send....
myname
data sent>0m
waiting for ack.....
receiver > packet recieved
data sent>1y
waiting for ack.....
receiver > packet received

data sent>0n

waiting for ack.....
receiver > packet recieved
data sent>1a
waiting for ack.....
Time out resending data....
data sent>1a
waiting for ack.....
receiver > packet recieved
data sent>0m
waiting for ack.....
receiver > packet recieved
data sent>1e
waiting for ack.....
receiver > packet recieved
All data sent. exiting.

Receiver:
waiting for connection...
Connection established :
receiver >0m
receiver >1y
receiver >0n
receiver >1a
receiver >1a duplicate data
receiver >0m
receiver >1e
Data recived=myname
waiting for connection...

RESULT
Thus the java program for implementing stop & wait and sliding window protocol is created and executed.