EXPERIMENT 5 – DAEMON PROGRAM

```
class adminThread extends Thread {
  adminThread() {
    setDaemon(false);
  }
  public void run() {
    boolean d = isDaemon();
    System.out.println("daemon = " + d);
  }
}

public class Tester {
  public static void main(String[] args) throws Exception {
    Thread thread = new adminThread();
    System.out.println("thread = " + thread.currentThread());
    thread.setDaemon(false);
    thread.start();
  }
}
```

EXPERIMENT 6 - IMPLEMENTING STOP & WAIT PROTOCOL

AIM

To implement the stop and wait protocol using java programming language.

PROCEDURE

SERVER

Step1: sequence ß 0

Step2: Accept new packet and assign sequence to it.

Step3: Send packet sequence with sequence number sequence.

Step4: Set timer for recently sent packets.

Step5: If error free acknowledgment from receiver and NextFrameExpected -> sequence then sequenceß NextFrameExpected.

Step6: If time out then go to step3.

Step7: Stop.

CLIENT

Step1: Start.

Step2: NextFrameExpectedß 0, repeat steps 3 forever.

Step3: If error-free frame received and sequence= NextFrameExpected, then pass packet to higher layer and NextFrameExpectedß NextFrameExpected+1(modulo 2).

Step4: Stop.

PROGRAM

SERVER

```
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",2005);
sequence=0;
out=new ObjectOutputStream(sender.getOutputStream());
out.flush();
 in=new ObjectInputStream(sender.getInputStream());
 str=(String)in.readObject();
 System.out.println("reciver     > "+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();
}}
```

CLIENT

```
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;
Receiver(){}
public void run(){
try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
reciever = new ServerSocket(2005,10);
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[]){
Receiver s=new Receiver();
while(true){
s.run();
}
}
}
```

RESULT:

Thus the program is executed successfully and the output is verified.