Experiement – 2

Program: 20 – Lines

```
import java.io.*;
import java.net.*;
public class Experiment_2 {
    public static void main(String[] args) {
        String website = "https://www.gstatic.com/webp/gallery/1.jpg";
        String fileName = "test_image.jpg";
        try (InputStream in = new URL(website).openStream();
             OutputStream out = new FileOutputStream(fileName)) {
            System.out.println("Downloading File From: " + website);
            byte[] buffer = new byte[2048];
            int bytesRead;
            while ((bytesRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
            System.out.println("Download complete.");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Experiment – 3A

Server Program: 17 – lines

```
import java.net.*;
import java.io.*;
public class EchoServer_3A{
   public static void main(String[] args) throws IOException {
      try (ServerSocket serverSocket = new ServerSocket(1234)) {
         System.out.println("server started. waiting for client...");
         try (Socket clientSocket = serverSocket.accept();
              PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
              BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()))) {
            System.out.println("client connected.");
            out.println("hello from server!");
            System.out.println("client says: " + in.readLine());
         }
      }
   }
}
```

Client Program: 15 – lines

```
import java.net.*;
import java.io.*;
public class EchoClient_3A{
   public static void main(String[] args) throws IOException {
      System.out.println("trying to connect...");
      try (Socket socket = new Socket("localhost", 1234);
          BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
          PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {
         System.out.println("connected to server.");
         System.out.println("server says: " + in.readLine());
         out.println("hello from client!");
      }
   }
}
```

Experiment – 3B

Server Program: 21 – Lines

```
import java.io.*;
import java.net.*;
class ChatServer_3B{
   public static void main(String[] args) throws Exception {
      ServerSocket ss = new ServerSocket(4000);
      System.out.println("Server started");
      try (Socket s = ss.accept();
          BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
          PrintWriter out = new PrintWriter(s.getOutputStream(), true);
          BufferedReader kb = new BufferedReader(new InputStreamReader(System.in))) {
         System.out.println("client connected");
         String msg;
         while (!(msg = in.readLine()).equals("end")) {
            System.out.println("Server: " + msg);
            System.out.print("Message to Client: ");
            out.println(kb.readLine());
         }
         System.out.println("Client Disconnected");
      }
   }
}
```

Client Program: 20 – Lines

```
import java.io.*;
import java.net.*;
class ChatClient_3B{
    public static void main(String[] args) throws Exception {
        try (Socket s = new Socket(InetAddress.getLocalHost(), 4000);
            BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
            PrintWriter out = new PrintWriter(s.getOutputStream(), true);
            BufferedReader kb = new BufferedReader(new InputStreamReader(System.in))) {
           System.out.println("Type \"end\" to quit");
           String msg;
           while (true) {
              System.out.print("Message to Server: ");
              msg = kb.readLine();
              out.println(msg);
              if (msg.equals("end")) break;
              System.out.println("Client: " + in.readLine());
           }
        }
    }
}
```

Experiment – 4

Server Program: 10 – lines

```
import socket
dns_records = {"annauniv.edu": "103.70.60.38", "google.com": "142.250.182.206"}
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('localhost', 53535))
print("DNS Server is running...")
while True:
    data, addr = s.recvfrom(1024)
    domain = data.decode()
    print("received query for:", domain)
    ip = dns_records.get(domain, "Domain not found")
    s.sendto(ip.encode(), addr)
```

Client Program: 8 – Lines

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
domain = input("enter domain name: ")
s.sendto(domain.encode(), ('localhost', 53535))
data, _ = s.recvfrom(1024)
print("IP address:", data.decode())
s.close()
```

Experiment – 5

Wireshark Capturing Network Packets

Experiment – 6

Server Program: 10 – Lines

```
import socket
arp = {"192.168.1.2": "AA:BB:CC:DD:EE:01", "192.168.1.3": "AA:BB:CC:DD:EE:02"}
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('localhost', 54545))
print("ARP Server is running...")
while True:
    d,a = s.recvfrom(1024)
    print("received query for", d.decode())
    s.sendto(arp.get(d.decode(), "MAC not found").encode(), a)
```

Client Program: 6 – Lines

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
c.sendto(input("Enter IP address: ").encode(), ('localhost', 54545))
print("MAC Address:", c.recvfrom(1024)[0].decode())
c.close()
```

Experiment – 6B

Server Program: 9 – Lines

```
import socket
rarp = {"AA:BB:CC:DD:EE:01": "192.168.1.2", "AA:BB:CC:DD:EE:02": "192.168.1.3"}
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('localhost', 55555))
print("RARP Server is running...")
while True:
    d,a = s.recvfrom(1024)
    print("received query for", d.decode())
    s.sendto(rarp.get(d.decode(), "IP not found").encode(), a)
```

Client Program: 5 – Lines

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
c.sendto(input("Enter MAC address: ").encode(), ('localhost', 55555))
print("IP Address:", c.recvfrom(1024)[0].decode())
c.close()
```

Experiment – 7

Program: 15 – lines

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp start
$ns run
```

Experiment – 8

Program: 21 – Lines

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp start
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n1 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr start
$ns run
```

Experiment – 9A

Program: 8 – Lines

```
set ns [new Simulator]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]


$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n0 1Mb 10ms DropTail


$ns run
```

Experiment – 9B

Program: 8 – Lines

```
set ns [new Simulator]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]


$ns duplex-link $n0 $n1 1Mb 5ms DropTail

$ns duplex-link $n1 $n2 1Mb 5ms DropTail

$ns duplex-link $n2 $n0 1Mb 5ms DropTail


$ns run
```

Experiment – 10

Server Program: 15 – lines

```
import socket, zlib
s = socket.socket(); s.bind(('localhost',9999)); s.listen(1)
print("crc server is running...")
conn, addr = s.accept()
print("connected to:", addr)
data = conn.recv(1024)
msg, crc = data[:-4], data[-4:]
calc_crc = zlib.crc32(msg).to_bytes(4, 'big')
if calc_crc == crc:
    print("no error detected. Data is correct.")
else:
    print("error detected in data.")
conn.close()
s.close()
```

Client program: 10 – Lines

```
import socket, zlib
s = socket.socket(); s.connect(('localhost',9999))
data = input("enter binary data: ").encode()
poly = input("enter generator polynomial: ")  # input accepted but unused in this simple example
crc = zlib.crc32(data).to_bytes(4, 'big')
codeword = data + crc
print("received codeword:", codeword)
s.send(codeword)
s.close()
```