Experiment No.13

Title: Implementation Socket Programming Client -Server
Aim: Implementation of Socket Programming-Iterative Server Implementation.

Theory:

Both clients & servers can run in concurrent mode:

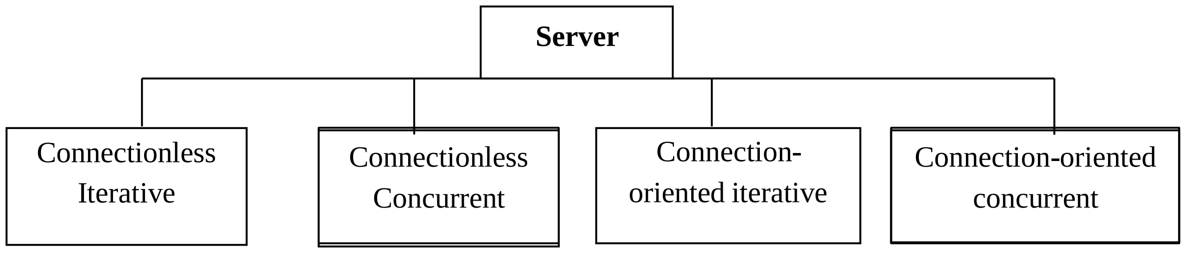
- **Concurrent in Clients** :-

Clients can run on a machine either iteratively on concurrently. Running clients iteratively means running them one by one, one client must start, run & terminate before the m/c can start another client. Most computers today, however, allow concurrent clients, i.e two or more clients can run at the same time.

- **Concurrency in servers** :-

An iterative server can process only one request at a time, it receive a request, processes it, & sends the response to the response to the requestor before it handles another request. A concurrent server, on the other hand, can process many requests at the same time & thus can share its time between many respects. The servers use either UDP, a connectionless transport layer protocol or TCP, a connection-oriented transport layer protocol & the service method.

Theoretically, we can have four types of servers: Connectionless iterative, Connectionless Concurrent, Connection-oriented iterative & connection-oriented concurrent.


Server
Connectionless Iterative
Connectionless Concurrent
Connection-oriented iterative
Connection-oriented concurrent

- **Connectionless Iterative Server** :-

The servers that use UDP are normally iterative, which as we have said, means that the server processes one request at a time. A server gets the request in a datagram from UDP, processes the request, & gives the response to UDP to send to the client.

The server plays no attention to the other datagram's. Theses datagram's are stored in a queue, waiting for service. They could all be from many clients. In either case they are processed one by one in order of arrival.

Statement: Implement Socket programming of client-sever.

Program:

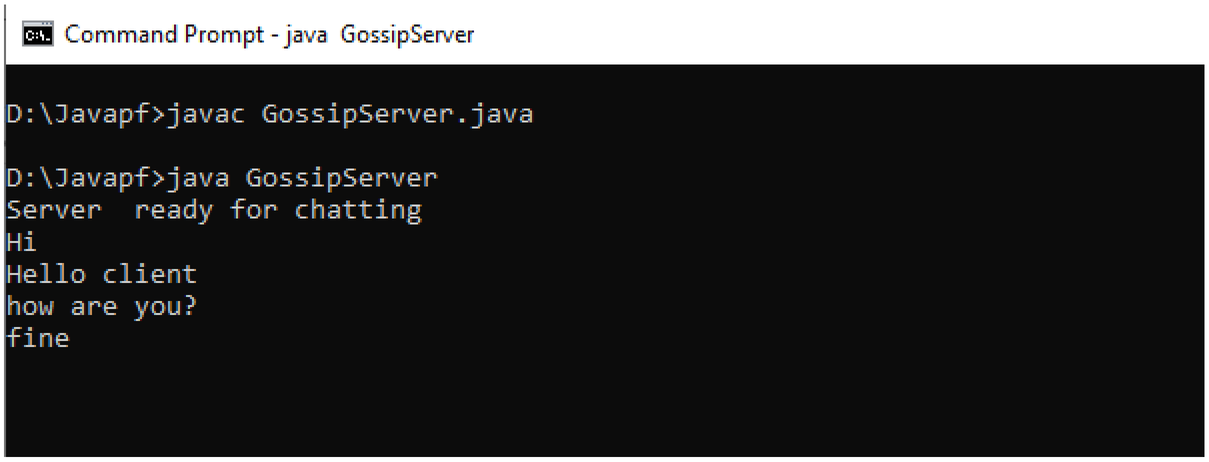
For Server:

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


 ServerSocket sersock = new ServerSocket(3000);
      System.out.println("Server  ready for chatting");
      Socket sock = sersock.accept( );
                              // reading from keyboard (keyRead object)
      BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
                                                               // sending to client (pwrite object)
      OutputStream ostream = sock.getOutputStream();
      PrintWriter pwrite = new PrintWriter(ostream, true);

                              // receiving from server ( receiveRead  object)
      InputStream istream = sock.getInputStream();
      BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

      String receiveMessage, sendMessage;
      while(true)
      {
        if((receiveMessage = receiveRead.readLine()) != null)
        {
           System.out.println(receiveMessage);
        }
        sendMessage = keyRead.readLine();
        pwrite.println(sendMessage);
        pwrite.flush();
      }
    }
}
```

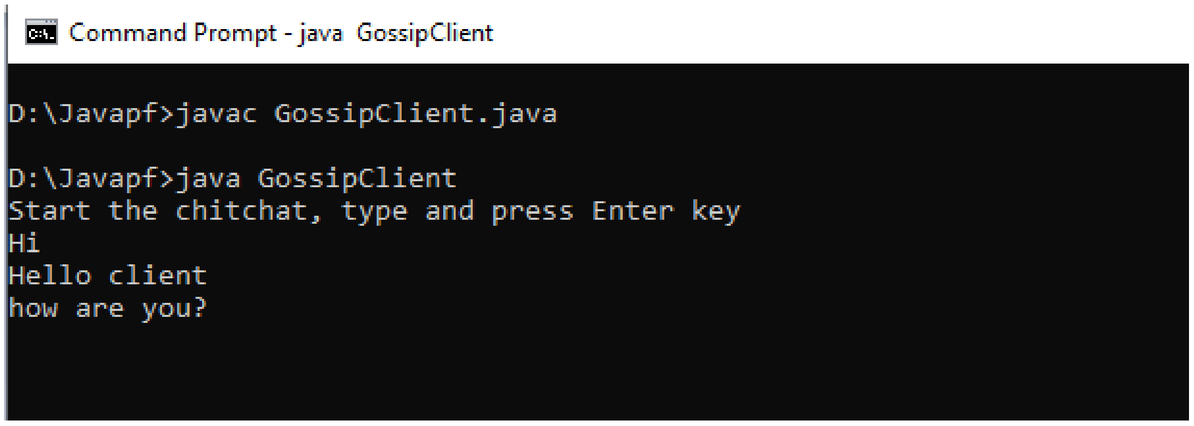
For Client:

```
        import java.io.*;
import java.net.*;
public class GossipClient
{
  public static void main(String[] args) throws Exception
  {
     Socket sock = new Socket("127.0.0.1", 3000);
                               // reading from keyboard (keyRead object)
     BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
                              // sending to client (pwrite object)
     OutputStream ostream = sock.getOutputStream();
     PrintWriter pwrite = new PrintWriter(ostream, true);

                              // receiving from server ( receiveRead  object)
     InputStream istream = sock.getInputStream();
     BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

     System.out.println("Start the chitchat, type and press Enter key");

     String receiveMessage, sendMessage;
     while(true)
     {
        sendMessage = keyRead.readLine();  // keyboard reading
        pwrite.println(sendMessage);       // sending to server
        pwrite.flush();                    // flush the data
        if((receiveMessage = receiveRead.readLine()) != null) //receive from server
        {
            System.out.println(receiveMessage); // displaying at DOS prompt
        }
      }
    }
}
```

Output:-


Command Prompt - java GossipServer

```
D:\Javapf>javac GossipServer.java

D:\Javapf>java GossipServer
Server  ready for chatting
Hi
Hello client
how are you?
fine
```


Command Prompt - java GossipClient

```
D:\Javapf>javac GossipClient.java

D:\Javapf>java GossipClient
Start the chitchat, type and press Enter key
Hi
Hello client
how are you?
```

Conclusion: Thus we have studied and implemented the socket programming.