Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

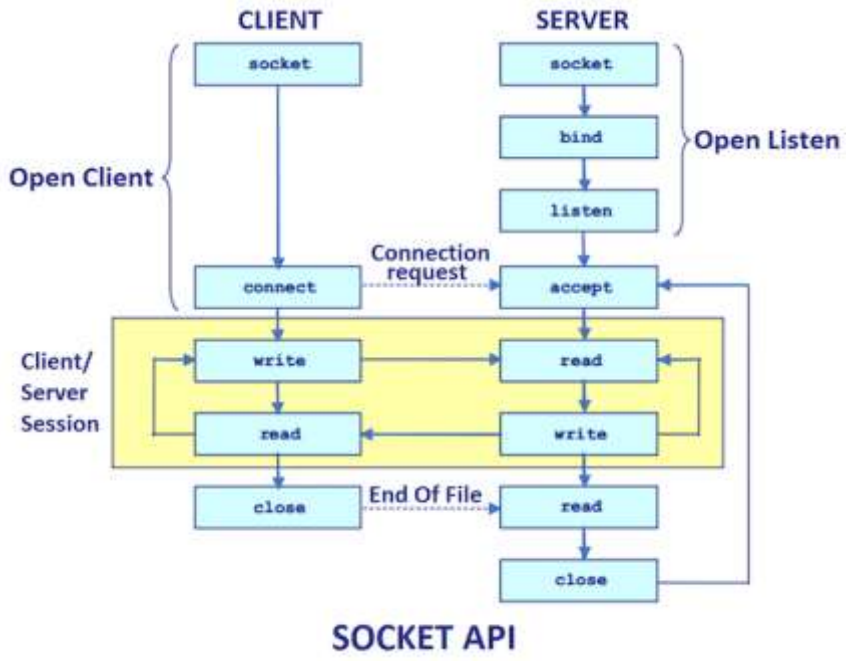
Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

Name	**Pratik Pujari**		
UID no.	**2020300054**	**Class:**	**Comps C Batch**
Experiment No.	5		

AIM:	To implement Socket Programming
THEORY:	**Java Socket Programming** Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less. Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming. The client in socket programming must know two information: 1. IP Address of Server, and 2. Port number. Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.



Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods

Method	Description
1) public InputStream getInputStream()	returns the InputStream attached with this socket.
2) public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Method	Description
1) public Socket accept()	returns the socket and establish a connection between server and client.
2) public synchronized void close()	closes the server socket.

Server Client	

Code:	import java.net.*;
	import java.io.*;
	public class chatserver {
	public static void deleteString(String st){
	
	for(int i=0;i<=st.length();i++)
	System.out.print('\b');
	
	}
	public static void main(String args[]) throws Exception {
	ServerSocket ss=new ServerSocket(2000);
	Socket sk=ss.accept();
	BufferedReader cin=new BufferedReader(new InputStreamReader(sk.getInputStream()));
	PrintStream cout=new PrintStream(sk.getOutputStream());
	BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
	String s,message;
	message="Awaiting for clients's reply!";
	System.out.println("CHAT SERVER started ");
	System.out.println("Connected to Port : 2000");
	System.out.println("Type End to leave Chat");
	while (true)
	{
	System.out.print(message);
	s=cin.readLine();
	if (s.equalsIgnoreCase("END"))
	{
	cout.println("BYE");
	break;
	}
	deleteString(message);
	System. out.print("\nClient : "+s+"\n");

```
            System.out.print("Server : ");
            s=stdin.readLine();
            cout.println(s);
                }
                ss.close();
                sk.close();
                cin.close();
                cout.close();
                stdin.close();
        }
}
```

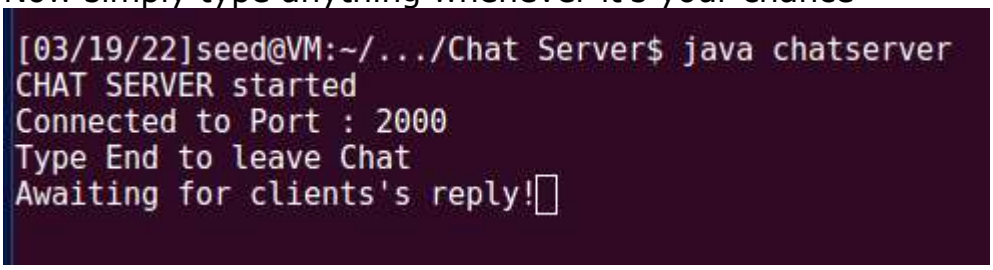
Client Server	
CALCULATION:	

```
import java.net.*;
import java.io.*;
public class  chatclient {
        public static void deleteString(String st){

                for(int i=0;i<=st.length();i++)
                        System.out.print('\b');

        }
        public static void main(String args[]) throws
Exception
        {
                Socket sk=new Socket("127.0.0.1",2000);
                BufferedReader sin=new BufferedReader(new
InputStreamReader(sk.getInputStream()));
                PrintStream sout=new
PrintStream(sk.getOutputStream());
                BufferedReader stdin=new BufferedReader(new
InputStreamReader(System.in));
                String s,message;
                message="Awaiting for server's reply!";
```

```
                System.out.println("CHAT CLIENT Connected");
                System.out.println("Connected: 2000");
                System.out.println("Type End to exit the chat");
                while (  true )
                {
                        System.out.print("Client : ");
                        s=stdin.readLine();
            sout.println(s);
            System.out.print(message);
            s=sin.readLine();
            deleteString(message);
            System.out.print("\nServer : "+s+"\n");
            if ( s.equalsIgnoreCase("END") )
            break;
        }
        sk.close();
        sin.close();
        sout.close();
        stdin.close();
    }
}
```

OUTPUT TABLE:

RESULT:

Screenshots	
OUTPUT:	Firstly, start the chatserver.java in order start a socket  Then initialize the chat client for the sock of port 2000 to make a request an accept it After a connection is made server side should look like this  Client side  Now simply type anything whenever it's your chance 

```
[03/19/22]seed@VM:~/.../Chat Server$ java chatserver
CHAT SERVER started
Connected to Port : 2000
Type End to leave Chat
Awaiting for clients's reply!
```

```
Client : Hello
Awaiting for server's reply!
Server : Hi
Client : How is the weather?
Awaiting for server's reply!
Server : It is quite nice over here!
Client :
```

```
Awaiting for clients's reply!
Client : Hello
Server : Hi
Awaiting for clients's reply!
Client : How is the weather?
Server : It is quite nice over here!
Awaiting for clients's reply!
```

```
Client : Hello
Awaiting for server's reply!
Server : Hi
Client : How is the weather?
Awaiting for server's reply!
Server : It is quite nice over here!
Client : How's the war going?
Awaiting for server's reply!
Server : It's ok.
Client :
```

```
Awaiting for clients's reply!
Client : Hello
Server : Hi
Awaiting for clients's reply!
Client : How is the weather?
Server : It is quite nice over here!
Awaiting for clients's reply!
Client : How's the war going?
Server : It's ok.
Awaiting for clients's reply!
```

```
Terminal
[03/19/22]seed@VM:~/.../Chat Server$ java chatclient
CHAT CLIENT Connected
Connected: 2000
Type End to exit the chat
Client : Hello
Awaiting for server's reply!
Server : Hi
Client : How is the weather?
Awaiting for server's reply!
Server : It is quite nice over here!
Client : How's the war going?
Awaiting for server's reply!
Server : It's ok.
Client : End
Awaiting for server's reply!
Server : BYE
Client : Bye
```

The chat ends with client says End or the server side says Bye

RESULT: Learnt about the socket programming and its features. Learnt on how to generate a socket using the Socket class. Used the function/methods of BufferedReader and PrintStream in order to get and send messages of users.