



**Computer Engineering Department**

**Academic Year: 2021-2022**

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

<b>Name</b>	<b>Pratik Pujari</b>		
<b>UID no.</b>	<b>2020300054</b>	<b>Class:</b>	<b>Comps C Batch</b>
<b>Experiment No.</b>	<b>7</b>		

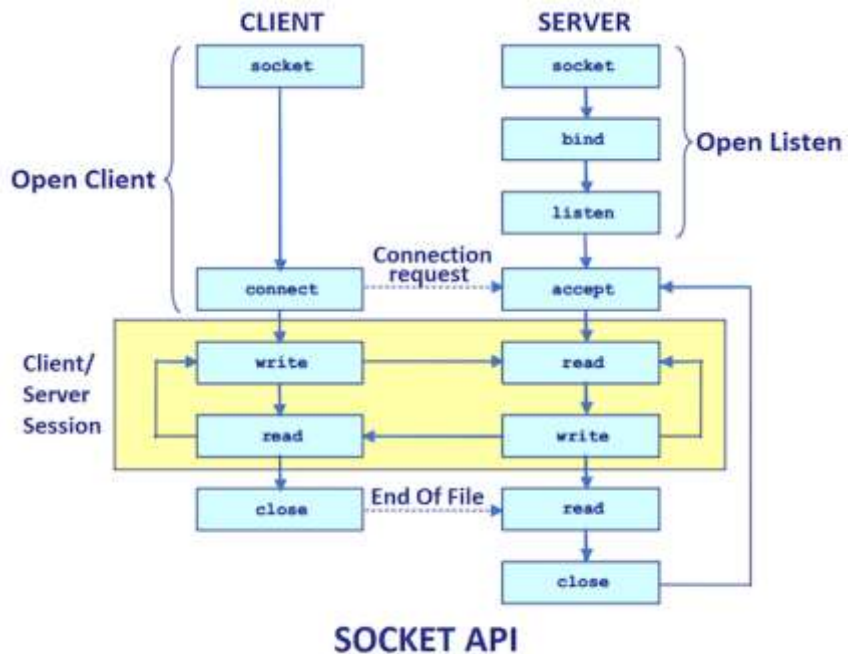
<b>AIM:</b>	To implement Socket Programming
<b>THEORY:</b>	<p><b>Java Socket Programming</b></p> <p>Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less.</p> <p>Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.</p> <p>The client in socket programming must know two information:</p> <ol style="list-style-type: none"><li>1. IP Address of Server, and</li><li>2. Port number.</li></ol> <p>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.</p>



## Computer Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech Sem.: 4 Course: CCN



### 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.



**Computer Engineering Department**

**Academic Year: 2021-2022**

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

	<ul style="list-style-type: none"><li>• <b>Socket creation:</b> <code>int sockfd = socket(domain, type, protocol)</code>  sockfd: socket descriptor, an integer (like a file-handle) domain: integer, specifies communication domain. We use AF_LOCAL as defined in the POSIX standard for communication between processes on the same host. For communicating between processes on different hosts connected by IPV4, we use AF_INET and AF_INET6 for processes connected by IPV6. type: communication type  SOCK_STREAM: TCP(reliable, connection oriented) SOCK_DGRAM: UDP(unreliable, connectionless) protocol: Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)</li><li>• <b>Setsockopt:</b> <code>int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);</code>  This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as "address already in use".</li><li>• <b>Bind:</b> <code>int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);</code>  After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.</li></ul>
--	---



**Computer Engineering Department**

**Academic Year: 2021-2022**

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

	<ul style="list-style-type: none"><li>• <b>Listen:</b> <code>int listen(int sockfd, int backlog);</code>  It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.</li><li>• <b>Accept:</b> <code>int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);</code>  It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.</li></ul> <p><b>Stages for Client</b></p> <ul style="list-style-type: none"><li>• <b>Socket connection:</b> Exactly same as that of server's socket creation</li><li>• <b>Connect:</b> <code>int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);</code>  The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server's address and port is specified in addr.</li></ul>
--	--



**Computer Engineering Department**

**Academic Year: 2021-2022**

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

Server Client	
<b>Code:</b>	<pre>import java.net.*; import java.io.*; public class chatserver {     public static void deleteString(String st){          for(int i=0;i&lt;=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");</pre>



**Computer Engineering Department**

**Academic Year: 2021-2022**

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

	<pre>System.out.print("Server : "); s=stdin.readLine(); cout.println(s);     }     ss.close();     sk.close();     cin.close();     cout.close();     stdin.close(); } }</pre>
--	--

<b>Client Server</b>	
<b>CALCULATION:</b>	<pre>import java.net.*; import java.io.*; public class chatclient {     public static void deleteString(String st){          for(int i=0;i&lt;=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!";</pre>



**Computer Engineering Department**

**Academic Year: 2021-2022**

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





	<pre>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(); }</pre>
--	--



**Computer Engineering Department**

**Academic Year: 2021-2022**

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

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





**Computer Engineering Department**

**Academic Year: 2021-2022**

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

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

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

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

```
Terminal
[03/19/22]seed@VM:~/.../Chat Servers$ 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.