

**U18CO018**  
**Shubham Shekhaliya**  
**Sub: CN**  
**Lab: Assignment 2**  
**Topic : Socket Programming**

- Perform Simple Socket Programming

Java Socket programming is used for communication between the applications running on different JRE(Client and Server).

Socket and ServerSocket classes are used for connection-oriented socket programming.

Client in socket programming must know two information:

1. IP Address of Server
2. Port number

## Example:

### Server.Java

```
import java.net.*;
import java.io.*;

public class Server {
    private Socket      socket = null;
    private ServerSocket server = null;
    private DataInputStream in  = null;
    public Server(int port) {
        try{
            server = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("Waiting for a client...");

            socket = server.accept();
            System.out.println("Client accepted");
            in = new DataInputStream(new BufferedInputStream(socket.getInputStream()));

            String line = "";
            while (!line.equals("Over")) {
                try{
                    line = in.readUTF();
                    System.out.println(line);

                }
                catch(IOException i) {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");

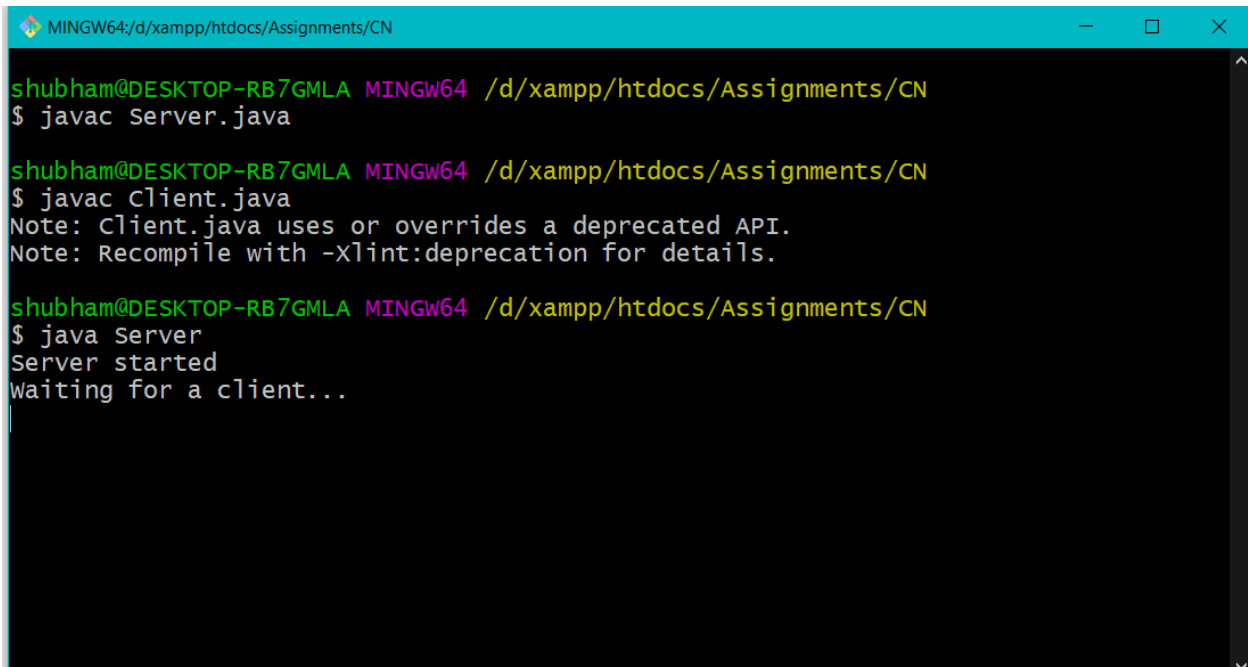
            socket.close();
            in.close();
        }
        catch(IOException i) {
            System.out.println(i);
        }
    }
    public static void main(String args[]) {
        Server server = new Server(5000);
    }
}
```

## Client.java

```
import java.net.*;
import java.io.*;

public class Client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
    public Client(String address, int port) {
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");
            input = new DataInputStream(System.in);
            out = new DataOutputStream(socket.getOutputStream());
        }
        catch(UnknownHostException u) {
            System.out.println(u);
        }
        catch(IOException i) {
            System.out.println(i);
        }
        String line = "";
        while (!line.equals("Over")) {
            try{
                line = input.readLine();
                out.writeUTF(line);
            }
            catch(IOException i) {
                System.out.println(i);
            }
        }
        try{
            input.close();
            out.close();
            socket.close();
        }
        catch(IOException i) {
            System.out.println(i);
        }
    }
    public static void main(String args[]) {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```

While Compiling Server.java and at a run time It will show.



```
MINGW64:/d/xampp/htdocs/Assignments/CN
shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ javac Server.java

shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ javac Client.java
Note: Client.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

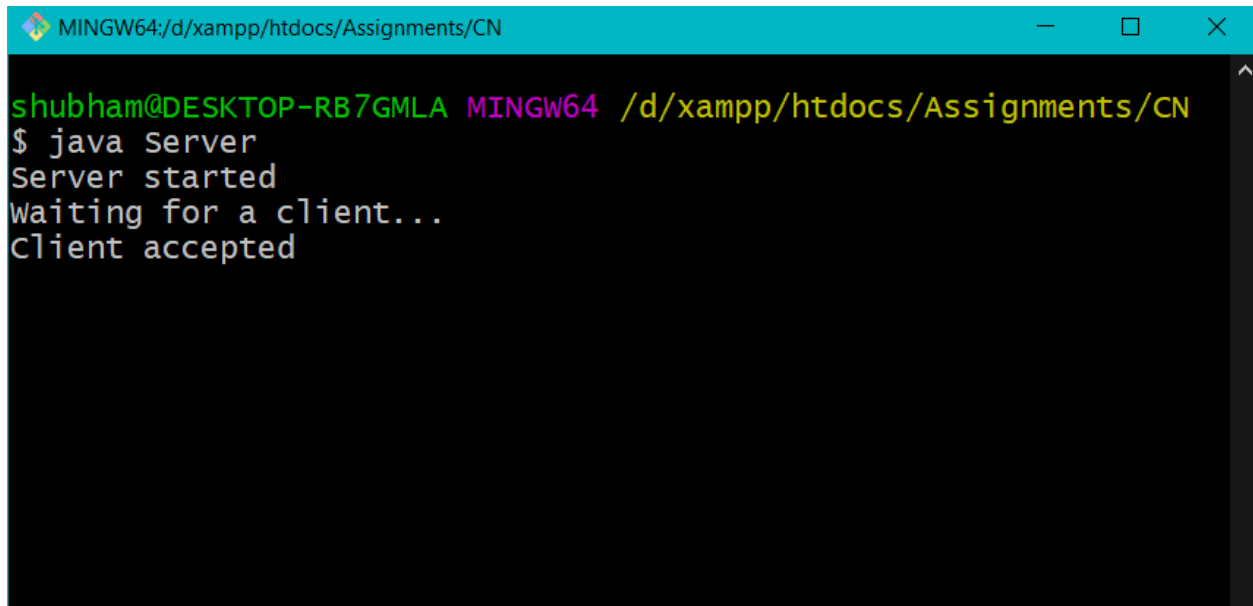
shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ java Server
Server started
waiting for a client...
```

While Compiling Server.java and at a run time It will show.



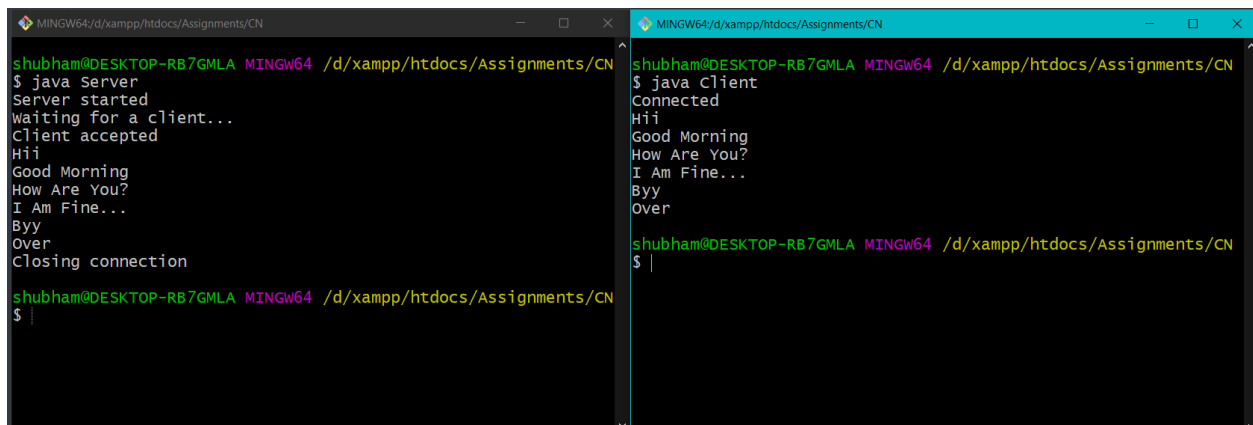
```
MINGW64:/d/xampp/htdocs/Assignments/CN
shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ java Client
Connected
```

At a Time of Connection Sever Also Show Us Client Connected Message

A terminal window titled 'MINGW64:/d/xampp/htdocs/Assignments/CN' with a blue header bar. The prompt is 'shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN'. The user enters '\$ java Server'. The output is 'Server started', 'Waiting for a client...', and 'Client accepted'.

```
shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ java Server
Server started
Waiting for a client...
Client accepted
```

And Whatever message will send through Client Server will receive and Show the message.

Two terminal windows side-by-side. The left window shows the server's output: 'Client accepted', 'Hii', 'Good Morning', 'How Are You?', 'I Am Fine...', 'Byy', 'Over', and 'Closing connection'. The right window shows the client's output: 'Connected', 'Hii', 'Good Morning', 'How Are You?', 'I Am Fine...', 'Byy', and 'Over'. Both windows have the same prompt and title as the first image.

```
shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ java Server
Server started
Waiting for a client...
Client accepted
Hii
Good Morning
How Are You?
I Am Fine...
Byy
Over
Closing connection

shubham@DESKTOP-RB7GMLA MINGW64 /d/xampp/htdocs/Assignments/CN
$ java Client
Connected
Hii
Good Morning
How Are You?
I Am Fine...
Byy
Over
```

And “Over” Will Closing Connection Message at Server Side.