

Object Oriented Programming with Java

Zheng Chen



Networking

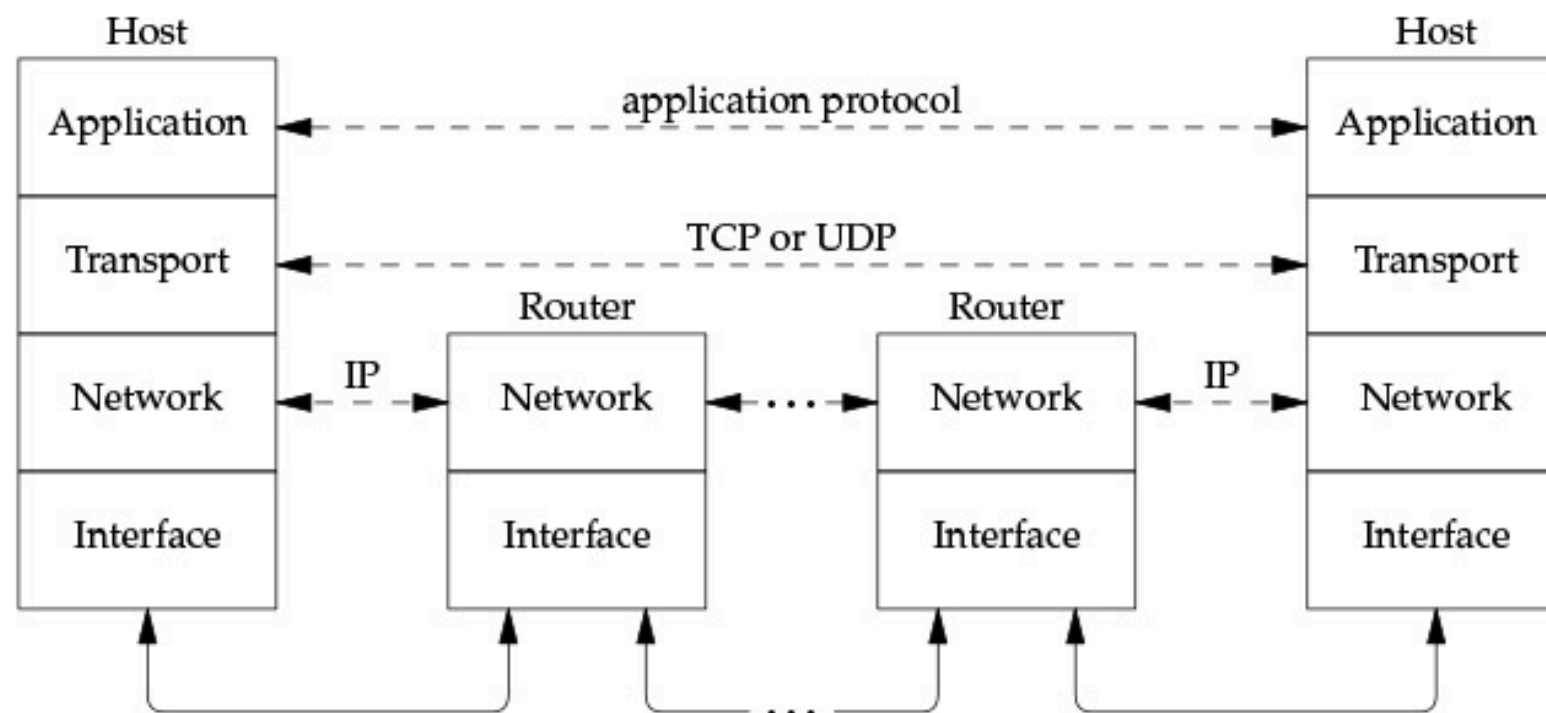
1. Networking Basics
2. HTTP and URL
3. Socket Programming
4. Datagram Programming



**KEEP
CALM
AND
CODE
JAVA**

Networking Basics

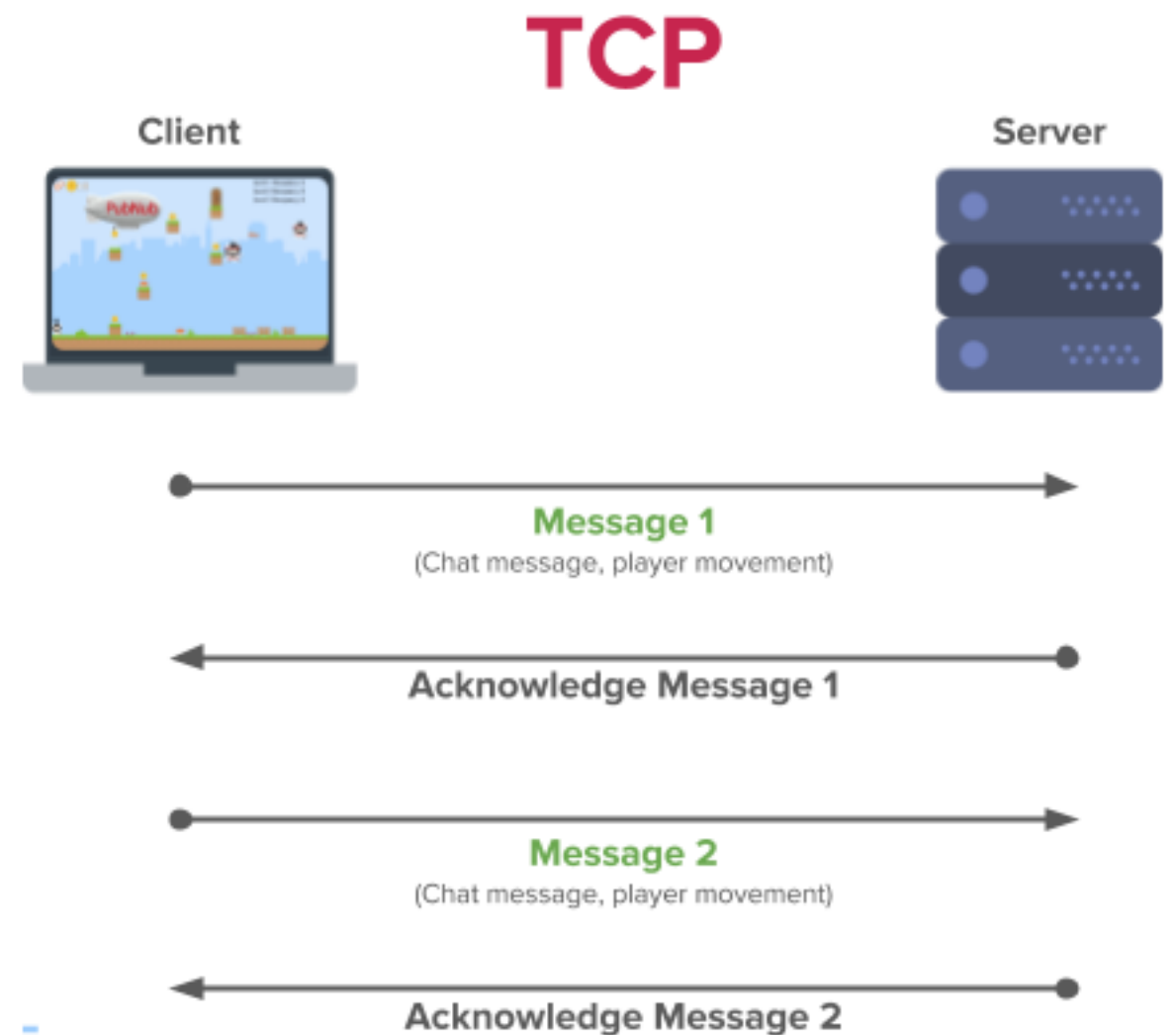
- Computers running on the Internet communicate to each other using either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP).



- When you write Java programs that communicate over the network, you are programming at the application layer.
- You need to decide which Transport Protocol your programs should use.

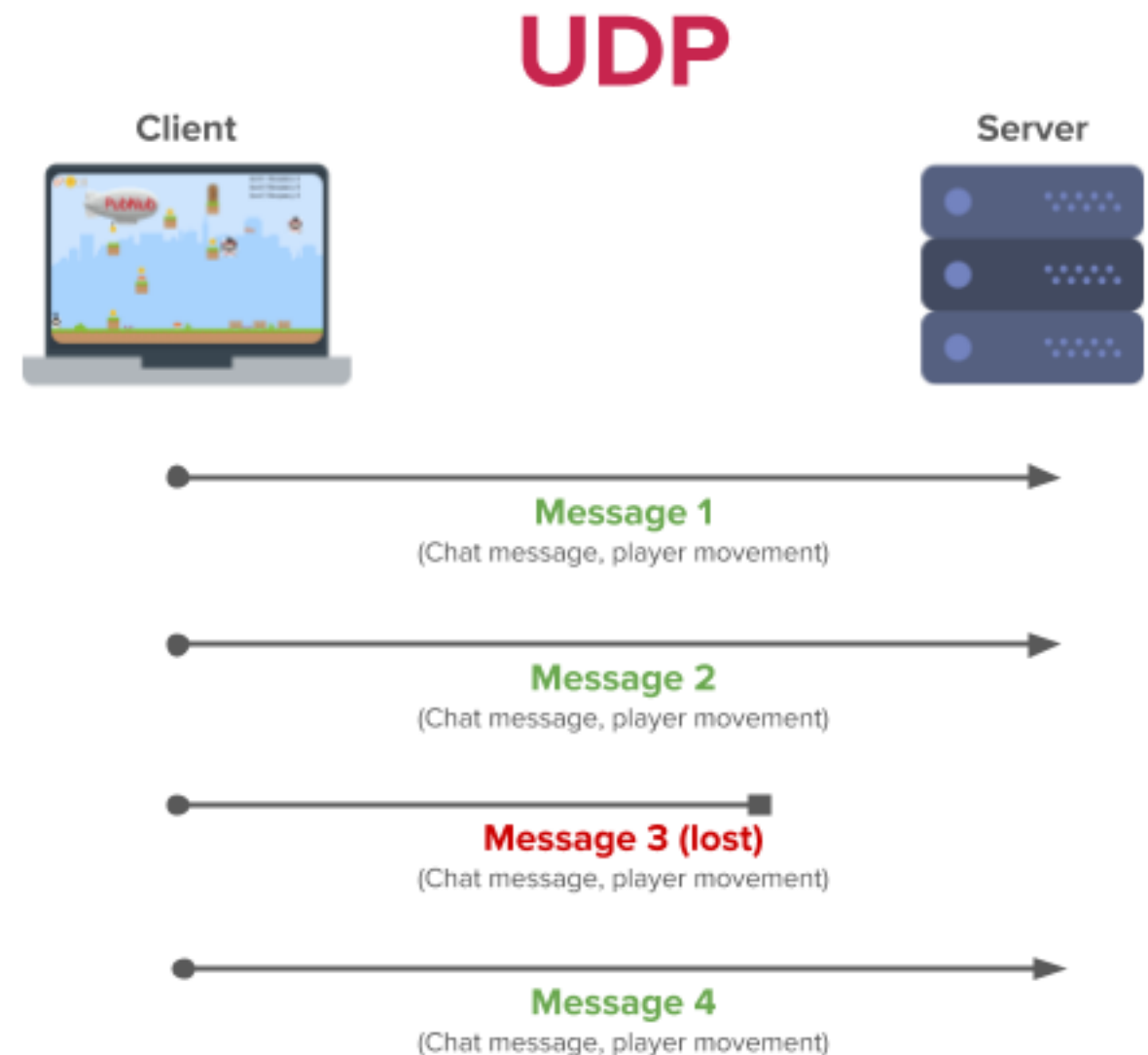
Transmission Control Protocol (TCP)

- TCP (Transmission Control Protocol) is a connection-based protocol that provides a reliable flow of data between two computers.
- TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.



User Datagram Protocol (UDP)

- UDP (User Datagram Protocol) is a protocol that sends independent packets of data, called datagrams, from one computer to another with no guarantees about arrival.
- UDP is not connection-based like TCP.



Port

- In computer networking, a **port** is a communication endpoint.
- Ports are identified for each protocol and address combination by 16-bit unsigned numbers, commonly known as the **port number**, ranging from **0 to 65535**.
- A port number is always associated with an IP address of a host and the protocol type of the communication.

IP address

- An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.
- Internet Protocol version 4 (IPv4) defines an IP address as a **32-bit number**.
- IP addresses are written and displayed in human-readable notations, such as 192.168.0.1

Domain Name and DNS

- A domain name is an identification string that defines a realm of administrative autonomy, authority or control within the Internet.
- Domain names are formed by the rules and procedures of the Domain Name System (DNS).

www.baidu.com -> 182.61.200.7

Networking

1. Networking Basics
- 2. HTTP and URL**
3. Socket Programming
4. Datagram Programming



**KEEP
CALM
AND
CODE
JAVA**

URL

- A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.
- A typical URL could have the form:

`http://www.example.com/index.html`

Or

`http://www.example.com/xxx.jpeg`

Accessing resource over network

1. Use URL to indicate the resource
2. Establish a connection
3. Get the InputStream
4. Read bytes

Example: accessing resource over network

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

public class DownloadWebpageExample {
    public static void main(String[] args) throws Exception {
        URL url = new URL("https://www.baidu.com/index.html");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        InputStream in = con.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        String nextLine = "";
        while ((nextLine = reader.readLine()) != null) {
            System.out.println(nextLine);
        }
    }
}
```

Start Page x DownloadWebpageExample.java x

Source

History

```

1  import java.io.*;
2  import java.net.*;
3
4  public class DownloadWebpageExample {
5      public static void main(String[] args) throws Exception {
6          URL url = new URL("https://www.baidu.com/index.html");
7          HttpURLConnection con = (HttpURLConnection) url.openConnection();
8          InputStream in = con.getInputStream();
9          BufferedReader reader = new BufferedReader(new InputStreamReader(in));
10         String nextLine = "";
11         while ((nextLine = reader.readLine()) != null) {
12             System.out.println(nextLine);
13         }
14     }
15 }

```

Output - TestAlgorithm (run) x

run:

<!DOCTYPE html>

```

<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html; charset=utf-8><meta http-equiv=X-UA-Compatible content
=IE=Edge><meta content=always name=referrer><link rel=stylesheet type=text/css href=https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r
/www/cache/bdorz/baidu.min.css><title>百度一下, 你就知道</title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <div class=he
rapper> <div class=s_form> <div class=s_form_wrapper> <div id=lg> <img hidefocus=true src=/www.baidu.com/img/bd_logo1.png width=27
0 height=129> </div> <form id=form name=f action=/www.baidu.com/s class=fm> <input type=hidden name=bdorz_come value=1> <input typ
e=hidden name=ie value=utf-8> <input type=hidden name=f value=8> <input type=hidden name=rsv_bp value=1> <input type=hidden name=rs
v_idx value=1> <input type=hidden name=tn value=baidu><span class="bg s_ipt_wr"><input id=kw name=wd class=s_ipt value maxlength=25
5 autocomplete=off autofocus=autofocus></span><span class="bg s_btn_wr"><input type=submit id=su value=百度一下 class="bg s_btn" autofo
cus></span> </form> </div> </div> <div id=u1> <a href=http://news.baidu.com name=tj_trnews class=mnav>新闻</a> <a href=https://www.ha
o123.com name=tj_trhao123 class=mnav>hao123</a> <a href=http://map.baidu.com name=tj_trmap class=mnav>地图</a> <a href=http://v.baidu
.com name=tj_trvideo class=mnav>视频</a> <a href=http://tieba.baidu.com name=tj_trtieba class=mnav>贴吧</a> <noscript> <a href=http://
ww.baidu.com/bdorz/login.gif?login&tpl=mn&u=http%3A%2F%2Fwww.baidu.com%2f%3fbdorz_come%3d1 name=tj_login class=lb>登录</a> </
noscript> <script>document.write('<a href="http://www.baidu.com/bdorz/login.gif?login&tpl=mn&u='+ encodeURIComponent(window.location
n.href+ (window.location.search == "" ? "?" : "&")+ "bdorz_come=1")+ "&" name="tj_login" class="lb">登录</a>');
</script> <a href=/www.baidu.com/more/ name=tj_briicon class=bri style="display: block;">更多产品</a> </div> </div> <
div> <div id=ftCon> <div id=ftConw> <p id=lh> <a href=http://home.baidu.com>关于百度</a> <a href=http://ir.baidu.com>About Baidu</a> <
p> <p id=cp>&copy;2017&nbsp;Baidu&nbsp;<a href=http://www.baidu.com/duty/>使用百度前必读</a>&nbsp;<a href=http://jianyi.baidu.com/ cla
cp-feedback>意见反馈</a>&nbsp;<img src=/www.baidu.com/img/gd.gif> </p> </div> </div> </div> </body> </html>
BUILD SUCCESSFUL (total time: 6 seconds)

```

HTML

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Here is the Title
    </title>
  </head>
  <body>
    <h1>
      Here is a Heading
    </h1>
    <p>
      Here is a paragraph.
    </p>
  </body>
</html>
```

HTTP - HyperText Transfer Protocol

- Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML.
- HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response.
- In most case, the client in HTTP is a browser, so it also known as **browser-server model**.

Important HTTP requests

- **GET** -- The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
- **POST** -- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
- **PUT** -- Replaces all the current representations of the target resource with the uploaded content.
- **DELETE** -- Removes all the current representations of the target resource given by URI.

Send HTTP requests in Java

1. Defining the request URL
2. Adding Request Parameters
3. Setting Request Headers ***
4. Configuring Timeouts ***
5. Handling Cookies ***
6. Reading the Response

```

import java.io.*;
import java.net.*;
import java.util.*;

public class HttpGetExample {
    public static void main(String[] args)
        throws Exception {
        String url = "https://www.bing.com/search";
        Map<String, String> params
            = new HashMap<>();
        params.put("q", "电子科技大学");
        String getUrl = url + "?";
        for (String key : params.keySet()) {
            getUrl += key + "="
                + params.get(key) + "&";
        }
        URL realUrl = new URL(getUrl);
        //打开和URL之间的连接
        URLConnection conn
            = realUrl.openConnection();
        //设置通用的请求属性
        conn.setRequestProperty(
            "accept", "*/*");
        conn.setRequestProperty(
            "connection", "Keep-Alive");

        conn.setRequestProperty("user-agent",
            "Mozilla/4.0 (compatible; MSIE 6.0;
            Windows NT 5.1;SV1)");
        //建立实际的连接
        conn.connect();
        //获取所有的响应头字段
        Map<String, List<String>> map
            = conn.getHeaderFields();
        //遍历所有的响应头字段
        for (String key : map.keySet()) {
            System.out.println(key
                + "-->" + map.get(key));
        }
        // 定义 BufferedReader输入流来读取URL的响应
        BufferedReader in =
            new BufferedReader(
                new InputStreamReader(
                    conn.getInputStream()));
        String line;
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }
    }
}

```

Handling HTTP requests

```
import com.sun.net.httpserver.*;
import java.io.*;
import java.net.InetSocketAddress;

public class BasicHttpServerExample {

    static String htmlPage =
        "<!DOCTYPE html>\n"
        + "<html>\n"
        + "    <head>\n"
        + "        <title>\n"
        + "            Here is the Title\n"
        + "        </title>\n"
        + "    </head>\n"
        + "    <body>\n"
        + "        <h1>\n"
        + "            Here is a Heading\n"
        + "        </h1>\n"
        + "        <p>\n"
        + "            Here is a paragraph.\n"
        + "        </p>\n"
        + "    </body>\n"
        + "</html>";
```

```
public static void main(String[] args)
    throws Exception {
    HttpServer server = HttpServer.create(
        new InetSocketAddress(80), 0);
    HttpContext context = server.createContext("/");
    context.setHandler(new HttpHandler() {
        @Override
        public void handle(HttpExchange exchange)
            throws IOException {
            String response = htmlPage;
            exchange.sendResponseHeaders(
                200, response.getBytes().length);
            OutputStream os =
                exchange.getResponseBody();
            os.write(response.getBytes());
            os.close();
        }
    });
    server.start();
}
```

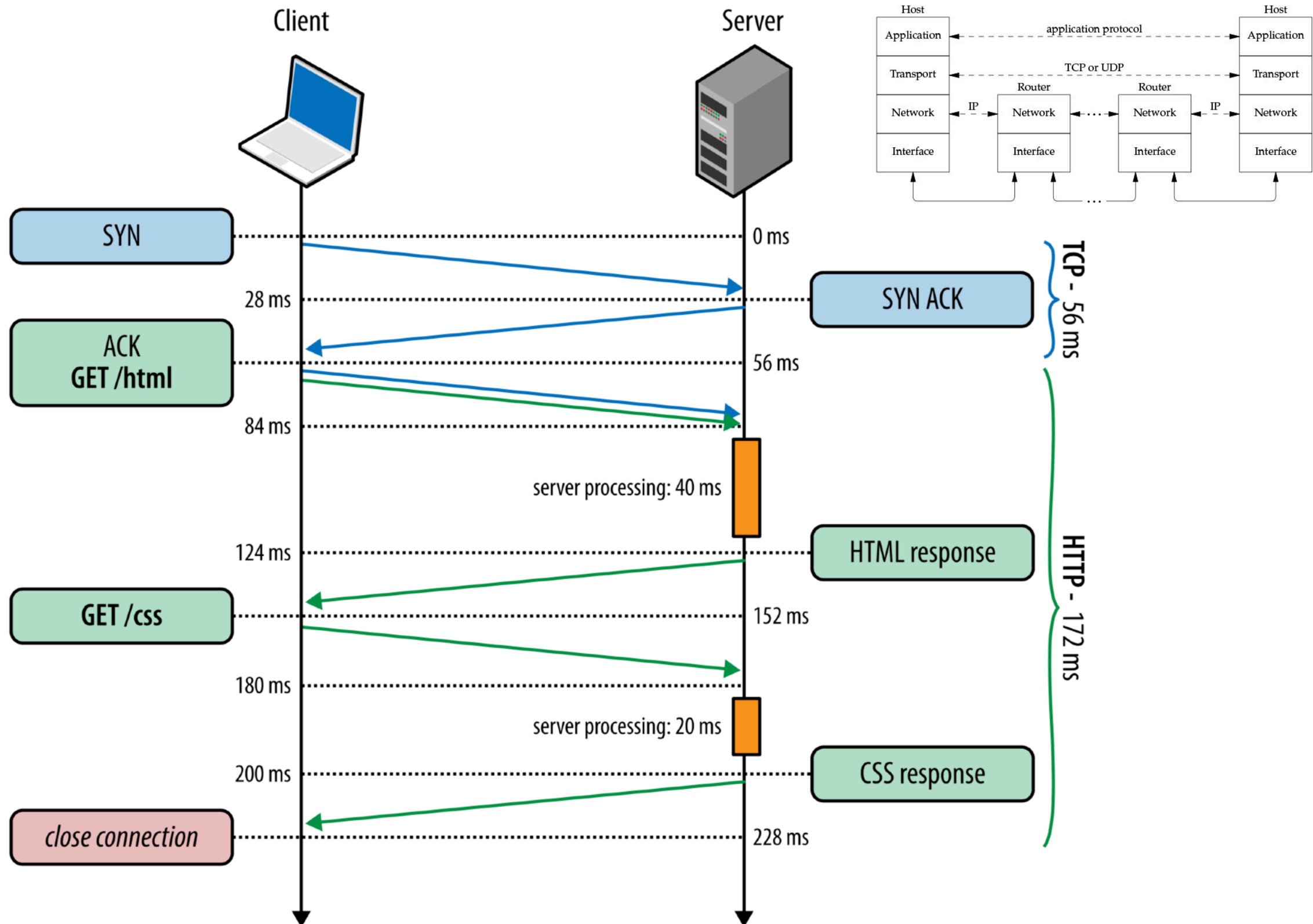
Networking

1. Networking Basics
2. HTTP and URL
- 3. Socket Programming**
4. Datagram Programming



**KEEP
CALM
AND
CODE
JAVA**

Http request over TCP connection



Socket Programming

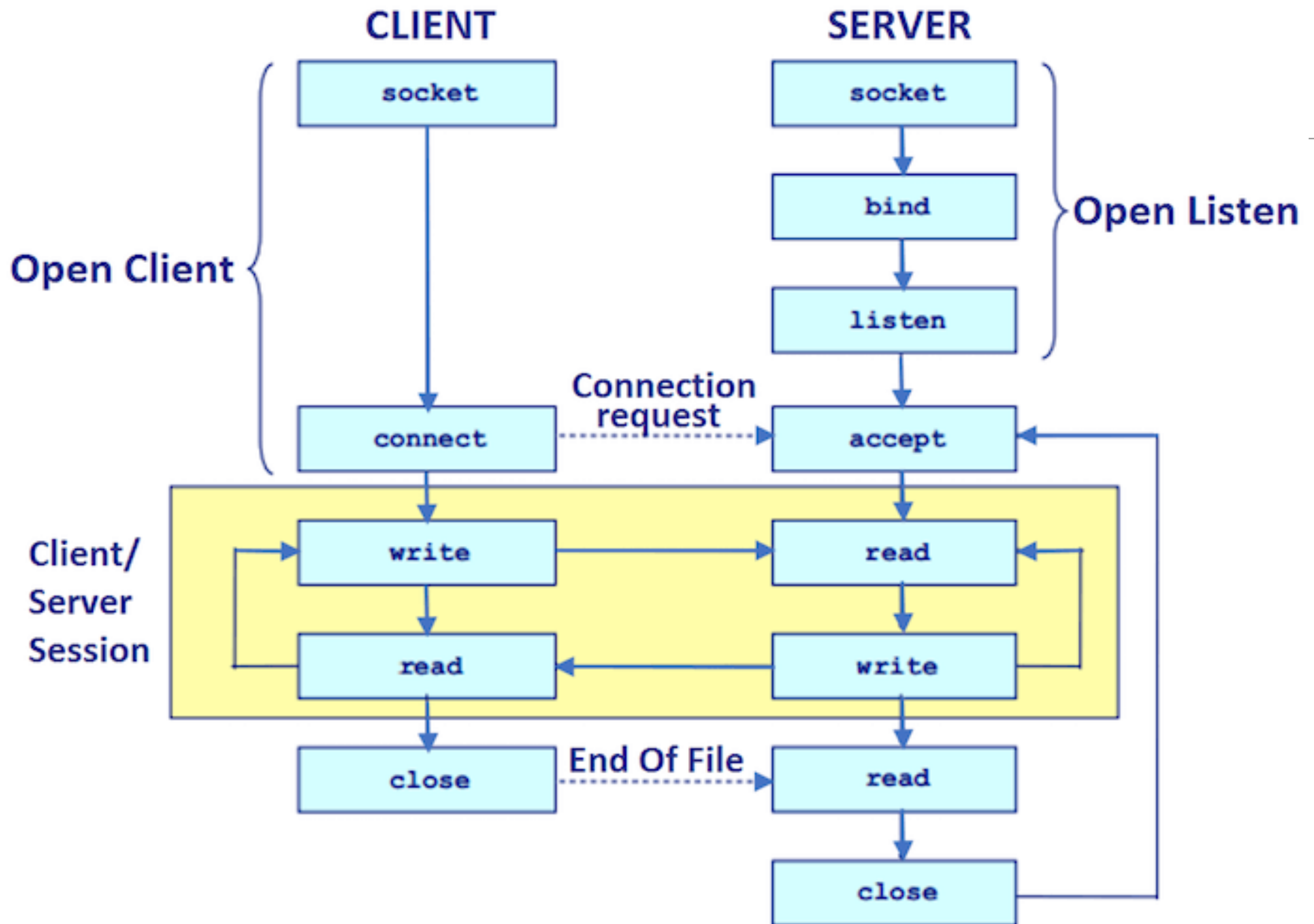
- Java Socket programming is used for communication between the applications running on the network.
- A socket is one end-point of a two-way communication link.
- Socket classes are used to represent the connection between a client program and a server program.
- The java.net package provides two classes--**Socket** and **ServerSocket**--that implement the client side of the connection and the server side of the connection, respectively.

ServerSocket Class Methods

Method	Description
<code>public ServerSocket(int port)</code>	attempts to create a server socket bound to the specified port.
<code>public Socket accept()</code>	returns the socket and establish a connection between server and client.
<code>public synchronized void close()</code>	closes the server socket.

Socket Class Methods

Method	Description
<code>public Socket(String host, int port)</code>	attempts to connect to the specified server at the specified port.
<code>public InputStream getInputStream()</code>	returns the InputStream attached with this socket.
<code>public OutputStream getOutputStream()</code>	returns the OutputStream attached with this socket.
<code>public synchronized void close()</code>	closes this socket



Example — Server

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

public class MyServer {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept();//establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String) dis.readUTF();
            System.out.println("message= " + str);
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Example — Client

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

public class MyClient {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 6666);
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            dout.writeUTF("Hello Server");
            dout.flush();
            dout.close();
            s.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



How about two-way communication?

```

public class Client {
    static String ServerIP = "localhost";
    static int ServerPort = 1234;
    public static void main(String args[]) throws Exception {
        Scanner scn = new Scanner(System.in);
        Socket s = new Socket(ServerIP, ServerPort);
        DataInputStream dis =
            new DataInputStream(s.getInputStream());
        DataOutputStream dos =
            new DataOutputStream(s.getOutputStream());

        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    while (true) {
                        // read the message to deliver.
                        String msg = scn.nextLine();
                        // write on the output stream
                        dos.writeUTF(msg);
                    }
                } catch (IOException e) {

```

```

                    e.printStackTrace();
                }
            }
        }).start();

        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    while (true) {
                        // read the message sent to this client
                        String msg = dis.readUTF();
                        System.out.println(msg);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
}

```

Multi_threaded Client

```

public class Server {

    static int Port = 1234;
    static List<ClientHandler> ar
        = new ArrayList<>();
    static int i = 0;

    public static void main(String[] args)
        throws IOException {
        ServerSocket ss = new ServerSocket(Port);
        Socket s;
        // looping for getting client request
        while (true) {
            // Accept the incoming request
            s = ss.accept();
            System.out.println(
                "New client request received : "
                + s);

            // obtain input and output streams
            DataInputStream dis =
                new DataInputStream(
                    s.getInputStream());
            DataOutputStream dos =
                new DataOutputStream(
                    s.getOutputStream());
        }
    }
}

```

```

// Create a new handler object for handling this request.
System.out.println(
    "Creating a new handler for this client...");
ClientHandler mtch =
    new ClientHandler(
        s, "client " + i++,
        dis, dos);

// add this client to active clients list
System.out.println(
    "Adding this client to active client list");
ar.add(mtch);

// Start a new Thread with this object.
new Thread(mtch).start();
}

```

Multi_threaded Server

```

class ClientHandler implements Runnable {

    Scanner scn = new Scanner(System.in);
    private String name;
    final DataInputStream dis;
    final DataOutputStream dos;
    Socket s;
    boolean isloggedin;

    // constructor
    public ClientHandler(Socket s, String name,
        DataInputStream dis, DataOutputStream dos) {
        this.dis = dis;
        this.dos = dos;
        this.name = name;
        this.s = s;
        this.isloggedin = true;
    }
}

```

Server ClientHandler

```

public void run() {
    try {
        String received;
        while (true) {
            // receive the string
            received = dis.readUTF();
            System.out.println(received);
            if (received.equals("logout")) {
                this.isloggedin = false;
                this.s.close();
                break;
            }
            // send to all other clients
            for (ClientHandler mc : Server.ar) {
                if (!mc.name.equals(name)) {
                    mc.dos.writeUTF(
                        this.name + ": "
                        + received);
                }
            }
        }
        // closing resources
        this.dis.close();
        this.dos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Networking

1. Networking Basics
2. HTTP and URL
3. Socket Programming
4. **Datagram Programming**



**KEEP
CALM
AND
CODE
JAVA**

Datagram

- **UDP (User Datagram Protocol)** is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss-tolerating connections between applications on the internet.
- A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.
- Java **DatagramSocket** class represents a connection-less socket for sending and receiving datagram packets.

DatagramSocket class

- **DatagramSocket**(int port) — Creates a datagram socket, bound to the specified local address.
- void **receive**(DatagramPacket p) — Receives a datagram packet from this socket.
- void **send**(DatagramPacket p) — Sends a datagram packet from this socket.
- void **close**() — Closes this datagram socket.

DatagramPacket class

- **DatagramPacket**(byte[] barr, int length): it creates a datagram packet. This constructor is used to receive the packets.
- **DatagramPacket**(byte[] barr, int length, InetAddress address, int port): it creates a datagram packet. This constructor is used to send the packets.

Example — sending

```
import java.net.*;

public class DSender {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        String str = "Welcome java";
        InetAddress ip = InetAddress.getByName("127.0.0.1");
        DatagramPacket dp = new DatagramPacket(
            str.getBytes(), str.length(), ip, 3000);
        ds.send(dp);
        ds.close();
    }
}
```

Example — receiving

```
import java.net.*;

public class DReceiver {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(), 0, dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

Homework

Build your own chatting room.

