

## Lab Sheet 4

[1] WAP to display protocol, authority, host, port, path, query, file name and ref from the URL  
"http://example.com:80/docs/books/tutorial" + "/index.html?name=networking#  
DOWNLOADING"

### Source code:

```
package Lab4;

import java.net.MalformedURLException;
import java.net.URL;

public class Question1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String name = "http://example.com:80/docs/books/tutorial" +
            "/index.html?name=networking#\r\n" + "DOWNLOADING";

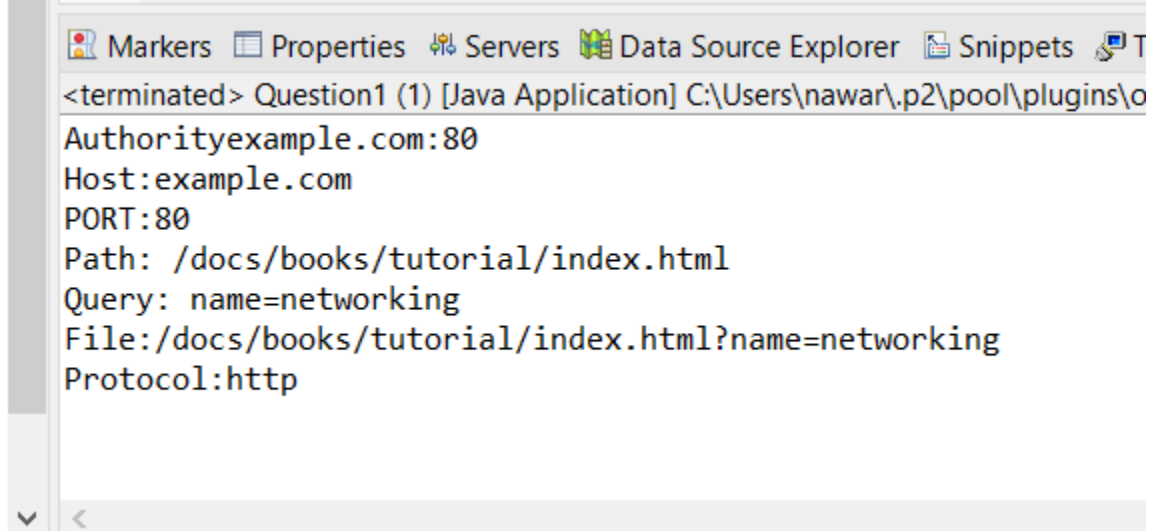
        try {
            URL uname= new URL(name);
            System.out.println("Authority"+ uname.getAuthority());
            System.out.println("Host:"+uname.getHost());
            System.out.println("PORT:"+uname.getPort());
            System.out.println("Path: " + uname.getPath());
            System.out.println("Query: " + uname.getQuery());
            System.out.println("File:"+uname.getFile());
            System.out.println("Protocol:"+uname.getProtocol());

        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            System.out.println(e.getMessage());
        }

    }

}
```

### Output:



[2] WAP to read directly from a URL. [HINT: you may use any URL for reference].

### Source code:

```
package Lab4;

import java.io.*;
import java.net.URL;
import java.net.URLConnection;

public class Question2 {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub

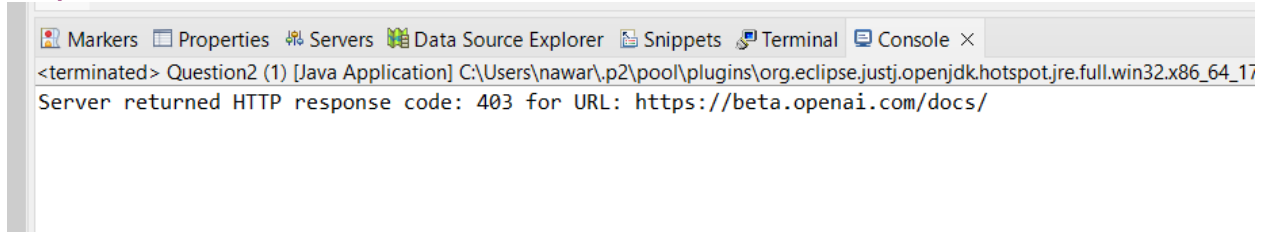
        String urlString = "https://beta.openai.com/docs/";
        URL u= new URL(urlString);
        URLConnection con = u.openConnection();
        try {
            BufferedReader br= new BufferedReader(new
InputStreamReader(con.getInputStream()));

            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (Exception e) {
            // TODO: handle exception
            System.out.println(e.getMessage());
        }

    }
}
```

```
}
```

### Output:



[3] WAP to implement chatting programming between server and multiple clients.

### Source code:

```
package Lab4;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class ChatServer {
    private static Set<PrintWriter> clientWriters = new HashSet<>();

    public static void main(String[] args) throws IOException {
        try (ServerSocket serverSocket = new ServerSocket(1234)) {
            System.out.println("Server is running and waiting for clients...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " + clientSocket);

                PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(),
true);
                clientWriters.add(writer);

                Thread clientHandler = new Thread(new ClientHandler(clientSocket,
writer));
                clientHandler.start();
            }
        }

        private static class ClientHandler implements Runnable {
            private Socket clientSocket;
            private PrintWriter writer;
```

```

    public ClientHandler(Socket clientSocket, PrintWriter writer) {
        this.clientSocket = clientSocket;
        this.writer = writer;
    }

    @Override
    public void run() {
        try (Scanner scanner = new Scanner(clientSocket.getInputStream())) {
            while (scanner.hasNextLine()) {
                String message = scanner.nextLine();
                broadcast(message);
            }
        } catch (IOException e) {
            System.out.println("Error handling client: " + e.getMessage());
        } finally {
            if (writer != null) {
                clientWriters.remove(writer);
            }
        }
    }
}

private static void broadcast(String message) {
    for (PrintWriter writer : clientWriters) {
        writer.println(message);
    }
}
}

```

```
package Lab4;
```

```

import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

```

```

public class ChatClient {
    public static void main(String[] args) throws IOException {
        try (Socket socket = new Socket("localhost", 1234);
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            Scanner in = new Scanner(socket.getInputStream());
            Scanner scanner = new Scanner(System.in)) {

            Thread receiveThread = new Thread(() -> {
                while (in.hasNextLine()) {
                    System.out.println("Server: " + in.nextLine());
                }
            });
            receiveThread.start();

            System.out.println("Enter your name:");

```

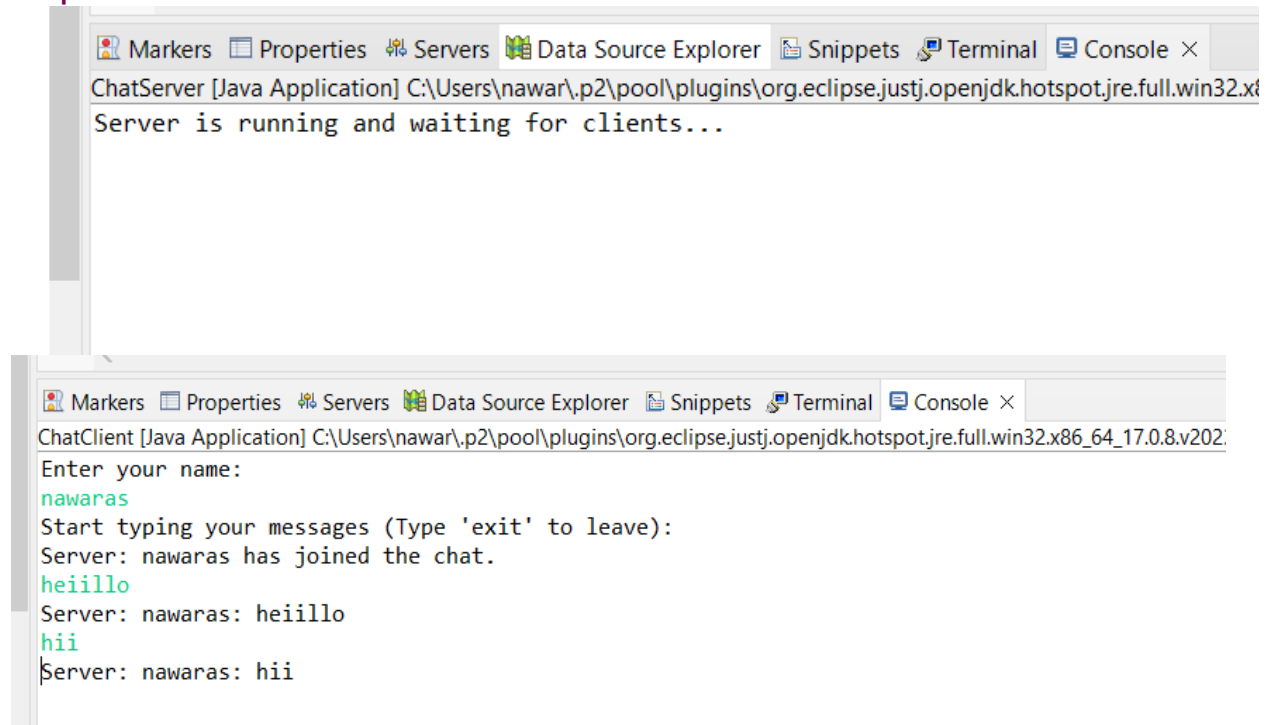
```

String name = scanner.nextLine();
out.println(name + " has joined the chat.");

System.out.println("Start typing your messages (Type 'exit' to leave):");
while (true) {
    String message = scanner.nextLine();
    if ("exit".equalsIgnoreCase(message)) {
        break;
    }
    out.println(name + ": " + message);
}
}
}
}
}

```

### Output:



The screenshot shows two Eclipse IDE windows. The top window, titled 'ChatServer [Java Application]', shows the output: 'Server is running and waiting for clients...'. The bottom window, titled 'ChatClient [Java Application]', shows the user input 'nawaras', the prompt 'Start typing your messages (Type 'exit' to leave):', and the server response 'Server: nawaras has joined the chat.' followed by the user input 'heiillo', the prompt, and the server response 'Server: nawaras: heiillo'. Finally, the user input 'hii' is shown, followed by the prompt and the server response 'Server: nawaras: hii'.

```

Markers Properties Servers Data Source Explorer Snippets Terminal Console ×
ChatServer [Java Application] C:\Users\nawar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20231011-1945\jre\bin\java.exe
Server is running and waiting for clients...

Markers Properties Servers Data Source Explorer Snippets Terminal Console ×
ChatClient [Java Application] C:\Users\nawar\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20231011-1945\jre\bin\java.exe
Enter your name:
nawaras
Start typing your messages (Type 'exit' to leave):
Server: nawaras has joined the chat.
heiillo
Server: nawaras: heiillo
hii
Server: nawaras: hii

```