Ex.No.5

String Handling and Socket Programming

16.09.2020

Mugilan E.S.

2019202033

1.

Implement a socket program for exchanging message between a client and server.

Server.java

```
package lab.five.chat;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
public class Server {
	private static ServerSocket serverSocket;
	private static int PORT = 2345;
	public static void main(String[] args) throws IOException, EOFException {
		serverSocket = new ServerSocket(PORT);
		System.out.println("Waiting for the Client...");
		Socket socket = serverSocket.accept();
		System.out.println("Connection Established.");
		DataInputStream inputStream;
		Scanner in = new Scanner(System.in);
		DataOutputStream outputStream;
		while (true) {
			inputStream =
					new DataInputStream(socket.getInputStream());
			String message = inputStream.readUTF();
			System.out.println("Client: " + message);
			outputStream =
					new DataOutputStream(socket.getOutputStream());
			if (message.equalsIgnoreCase("exit")) {
				System.out.println();
				System.out.println("Exiting Server...");
				inputStream.close();
				outputStream.close();
				socket.close();
				break;
			}
			String outMessage = in.nextLine();
			System.out.println("Server: " + outMessage);
```

```
                outputStream.writeUTF(outMessage);
            }
            System.out.println("Shutting Down Socket Server...");
            serverSocket.close();
        }
}
```

Client.java

```
package lab.five.chat;
import java.io.*;
import java.net.Socket;
import java.util.Scanner;
public class Client {
    private static int PORT = 2345;
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("localhost", PORT);
        System.out.println("Connection Established");
        DataInputStream inputStream;
        Scanner in = new Scanner(System.in);
        DataOutputStream outputStream;
        while (true) {
            inputStream = new DataInputStream(socket.getInputStream());
            outputStream = new DataOutputStream(socket.getOutputStream());
            System.out.println();
            System.out.print("You: ");
            String message = in.nextLine();
            outputStream.writeUTF(message);
            if(message.equalsIgnoreCase("exit")) {
                System.out.println("Exiting...");
                inputStream.close();
                outputStream.close();
                socket.close();
                break;
            }
            System.out.println("Server: " + inputStream.readUTF());
        }
        System.out.println("Exited.");
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
Waiting for the Client...
Connection Established.
Client: hi
hello
Server: hello
Client: how are you ?
i am fine
Server: i am fine
Client: exit

Exiting Server...
Shutting Down Socket Server...

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
Connection Established

You: hi
Server: hello

You: how are you ?
Server: i am fine

You: exit
Exiting...
Exited.

Process finished with exit code 0
```

2.

Use socket program to get system's

I. Date

II. Time

III. IP Address

Server.java

```
package lab.five.sysinfo;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Date;
public class Server {
    private static ServerSocket serverSocket;
    private static int PORT = 7654;
    public static void main(String[] args) throws IOException {
        serverSocket = new ServerSocket(PORT);
        System.out.println("Waiting for the client...");
        Socket socket = serverSocket.accept();
```

```
            System.out.println("Connection Established.");
            DataInputStream inputStream= new DataInputStream(socket.getInputStream());
            DataOutputStream outputStream = new DataOutputStream(socket.getOutputStream());
            while (true) {
                int choice = inputStream.readInt();
                System.out.println("Client: " + choice);
                String message = "";
                if (choice == 1) {
                    System.out.println("Getting Current Date");
                    String date = new SimpleDateFormat("MMM dd, yyyy").format(new Date());
                    message = "Current Date: " + date;
                    System.out.println(message);
                } else if (choice == 2) {
                    System.out.println("Getting Current Time");
                    String time = new SimpleDateFormat("hh:mm:ss a").format(new Date());
                    message = "Current Time: " + time;
                    System.out.println(message);
                } else if (choice == 3) {
                    System.out.println("Getting IP Address");
                    InetAddress host = InetAddress.getLocalHost();
                    message = "IP Address is " + host.getHostAddress();
                    System.out.println(message);
                } else {
                    System.out.println("Shutting Down Server...");
                    outputStream.writeUTF("exit");
                    inputStream.close();
                    outputStream.close();
                    socket.close();
                    break;
                }
                outputStream.writeUTF(message);
            }
            System.out.println("Shutting down socket server...");
            serverSocket.close();
        }
}
```

Client.java

```
package lab.five.sysinfo;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
```

```java
public class Client {
    private static int PORT = 7654;
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("localhost", PORT);
        System.out.println("Connection Established.");
        DataInputStream inputStream =
                new DataInputStream(socket.getInputStream());
        DataOutputStream outputStream =
                new DataOutputStream(socket.getOutputStream());
        Scanner in = new Scanner(System.in);
        while (true) {
            System.out.println("\n1 - Get Current Date\n2 - Get Current " +
                    "Time\n3 - Get the IP Address\nOther numbers to Exit");
            System.out.print("Enter your choice: ");
            int choice = in.nextInt();
            outputStream.writeInt(choice);
            String response = inputStream.readUTF();
            if (response.equalsIgnoreCase("exit")) {
                System.out.println("Exiting...");
                inputStream.close();
                outputStream.close();
                socket.close();
                break;
            }
            System.out.println(response);
        }
        System.out.println("Server Exited.");
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
Waiting for the client...
Connection Established.
Client: 1
Getting Current Date
Current Date: Sep 16, 2020
Client: 2
Getting Current Time
Current Time: 08:50:22 pm
Client: 3
Getting IP Address
IP Address is 192.168.43.84
Client: 4
Shutting Down Server...
Shutting down socket server...

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
Connection Established.

1 - Get Current Date
2 - Get Current Time
3 - Get the IP Address
Other numbers to Exit
Enter your choice: 1
Current Date: Sep 16, 2020

1 - Get Current Date
2 - Get Current Time
3 - Get the IP Address
Other numbers to Exit
Enter your choice: 2
Current Time: 08:50:22 pm

1 - Get Current Date
2 - Get Current Time
3 - Get the IP Address
Other numbers to Exit
Enter your choice: 3
IP Address is 192.168.43.84

1 - Get Current Date
2 - Get Current Time
3 - Get the IP Address
Other numbers to Exit
Enter your choice: 4
Exiting...
Server Exited.

Process finished with exit code 0
```

3.

Multiple processes listening on a same socket (broadcast message). Use an application to illustrate the above statement.

ThreadServer.java

```java
package lab.five.broadcast;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
class Clients extends Thread {
    final DataInputStream dis;
    final DataOutputStream dos;
    final Socket s;
    public Clients(Socket s, DataInputStream dis, DataOutputStream dos) {
        this.s = s;
        this.dis = dis;
        this.dos = dos;
    }
    public void run() {
        String receives;
        while (true) {
```

```
                try {
                    dos.writeUTF("Write anything ... Type Exit to terminate " +
                            "connection");
                    receives = dis.readUTF();
                    if (receives.equals("Exit")) {
                        System.out.println("Client    " + this.s + "    has sent exit" +
                                " ... ");
                        System.out.println("Terminating connection");
                        this.s.close();
                        System.out.println("Closed Connection");
                        break;
                    }
                    dos.writeUTF(receives);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            try {
                this.dis.close();
                this.dos.close();
            } catch (Exception a) {
                a.printStackTrace();
            }
        }
}
public class ThreadServer {
    public static void main(String args[]) throws IOException {
        Scanner sc = new Scanner(System.in);
        ServerSocket ss = new ServerSocket(5056);
        while (true) {
            Socket s = null;
            try {
                s = ss.accept();
                System.out.println("A new client is connected :: " + s);
                DataInputStream dis = new DataInputStream(s.getInputStream());
                DataOutputStream dos =
                        new DataOutputStream(s.getOutputStream());
                System.out.println("Assigning new thread for this client");
                Thread t = new Clients(s, dis, dos);
                t.start();
                String msg;
                System.out.println("Server Message = :: ");
                msg = sc.nextLine();
                dos.writeUTF(msg);
```

```
                } catch (Exception e) {
                        s.close();
                        e.printStackTrace();
                }
        }
    }
}
```

CThread1.java

```
package lab.five.broadcast;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Scanner;
public class CThread1 {
    public static void main(String[] args) throws IOException {
        try {
            Scanner sc = new Scanner(System.in);
            InetAddress ip = InetAddress.getByName("localhost");
            Socket s = new Socket(ip, 5056);
            DataInputStream dis = new DataInputStream(s.getInputStream());
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            while (true) {
                System.out.println(dis.readUTF());
                String send = sc.nextLine();
                dos.writeUTF(send);
                if (send.equals("Exit")) {
                    System.out.println("Terminating Connection :: " + s);
                    s.close();
                    System.out.println("Connection closed");
                    break;
                }
                String rec = dis.readUTF();
                System.out.println("Server Says ::    " + rec);
            }
            sc.close();
            dis.close();
            dos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

CThread2.java

```
package lab.five.broadcast;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Scanner;
public class CThread2 {
    public static void main(String[] args) throws IOException {
        try {
            Scanner sc = new Scanner(System.in);
            InetAddress ip = InetAddress.getByName("localhost");
            Socket s = new Socket(ip, 5056);
            DataInputStream dis = new DataInputStream(s.getInputStream());
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            while (true) {
                System.out.println(dis.readUTF());
                String send = sc.nextLine();
                dos.writeUTF(send);
                if (send.equals("Exit")) {
                    System.out.println("Terminating Connection :: " + s);
                    s.close();
                    System.out.println("Connection closed");
                    break;
                }
                String rec = dis.readUTF();
                System.out.println("Server Says ::    " + rec);
            }
            sc.close();
            dis.close();
            dos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
A new client is connected :: Socket[addr=/127.0.0.1,port=54321,localport=5056]
Assigning new thread for this client
Server Message = ::
hello client
A new client is connected :: Socket[addr=/127.0.0.1,port=54327,localport=5056]
Assigning new thread for this client
Server Message = ::
hello client
A new client is connected :: Socket[addr=/127.0.0.1,port=54332,localport=5056]
Assigning new thread for this client
Server Message = ::
Client  Socket[addr=/127.0.0.1,port=54321,localport=5056]  has sent exit ...
Terminating connection
Closed Connection
Client  Socket[addr=/127.0.0.1,port=54332,localport=5056]  has sent exit ...
Terminating connection
Closed Connection
```

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
Write anything ... Type Exit to terminate connection
hello server
Server Says ::  hello client
hello server
exit
Server Says ::  Write anything ... Type Exit to terminate connection
exit
Exit
Terminating Connection :: Socket[addr=localhost/127.0.0.1,port=5056,localport=54321]
Connection closed

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/openjdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Users/mugilan-codes/Library/Application Support/
Write anything ... Type Exit to terminate connection
hello
Server Says ::  hello
Write anything ... Type Exit to terminate connection
Exit
Terminating Connection :: Socket[addr=localhost/127.0.0.1,port=5056,localport=54332]
Connection closed

Process finished with exit code 0
```

4.

Use a MULTI THREADED SERVER to implement the following scenario.

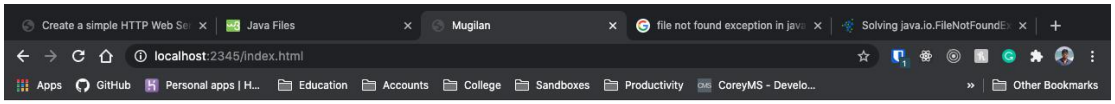
A Web server in two steps. In the end, you will have built a multi-threaded Web server that is capable of processing multiple simultaneous service requests in parallel. You should be able to demonstrate that your Web server is capable of delivering your home page to a Web browser.

Server.java

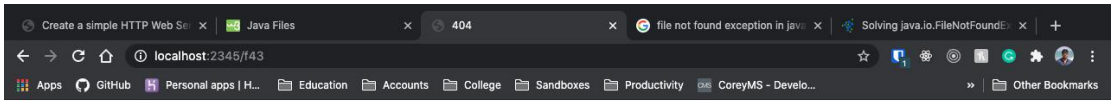
```
package lab.five.multithread;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
public class Server {
    private static String getHeaderURL(InputStream inputStream) {
        String header = "";
        Reader reader = new InputStreamReader(inputStream);
        try {
            int c;
            while ((c = reader.read()) != -1) {
                header += (char) c;
                if (header.contains("\n")) {
                    break;
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return header;
    }
    public static void main(String[] args) throws IOException {
        int PORT = 2345;
        ServerSocket serverSocket = new ServerSocket(PORT);
        System.out.println("Listening on PORT " + PORT);
        while (true) {
            try (Socket socket = serverSocket.accept()) {
                InputStream inputStream = socket.getInputStream();
                String val = getHeaderURL(inputStream);
                String[] vals = val.split("/");
                String value = "";
                if (vals.length != 0) {
                    if (vals.length > 1) {
                        value = vals[1].split(" ")[0].trim();
```

```
                        }
                    }
                    String httpResponse = "HTTP/1.1 200 OK\r\n";
                    if (value.equalsIgnoreCase("index.html")
                            || value.equalsIgnoreCase("")) {
                        System.out.println("Valid " + value + " " + httpResponse);
                        httpResponse += " Content-Type: text/html \r\n\r\n";
                        BufferedReader f =
                                new BufferedReader(new FileReader(new File("src" +
                                        "/lab/five/multithread/index.html")));
                        String str;
                        while ((str = f.readLine()) != null) {
                            httpResponse += str;
                        }
                        socket.getOutputStream()
                                .write(httpResponse.getBytes(StandardCharsets.UTF_8));
                        socket.getOutputStream().flush();
                        socket.close();
                    } else {
                        System.out.println("Invalid Page");
                        httpResponse = "HTTP/1.1 404 \r\n";
                        httpResponse += " Content-Type: text/html \r\n\r\n";
                        BufferedReader f =
                                new BufferedReader(new FileReader(new File("src" +
                                        "/lab/five/multithread/404.html")));
                        String str;
                        while ((str = f.readLine()) != null) {
                            httpResponse += str;
                        }
                        socket.getOutputStream()
                                .write(httpResponse.getBytes(StandardCharsets.UTF_8));
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

Output:



Mugilan Codes



404 Error - File Not Found

Source Code:

https://github.com/Mugilan-Codes/java-lab-exercises