```java
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;
import java.util.StringTokenizer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.ThreadPoolExecutor;

// The tutorial can be found just here on the SSaurel's Blog :
// https://www.ssaurel.com/blog/create-a-simple-http-web-server-in-java
// Each Client Connection will be managed in a dedicated Thread

public class JavaHTTPServer implements Runnable {

    static DocReader config = new DocReader();
    // config file
//    static Object o = config.jsonReader().get("WEB_ROOT");
//    static final File WEB_ROOT = new
File(config.jsonReader().get("WEB_ROOT").toString()); //doesnt work!!
//    static String webRoot =
config.jsonReader().get("WEB_ROOT").toString().replace("\"", "");
//unnecessary
    static final File WEB_ROOT = new
File(config.jsonReader().get("WEB_ROOT").toString());
    static final String DEFAULT_FILE = (String)
config.jsonReader().get("DEFAULT_FILE").toString();
    static final String FILE_NOT_FOUND = (String)
config.jsonReader().get("FILE_NOT_FOUND").toString();
    static final String METHOD_NOT_SUPPORTED = (String)
config.jsonReader().get("METHOD_NOT_SUPPORTED").toString();
    // port to listen connection
    static final int PORT =
Integer.parseInt(config.jsonReader().get("PORT").toString());

    static final String serverName = "ServerName";

    // verbose mode
    static final boolean verbose = true;

    // Client Connection via Socket Class
    private Socket connect;
    public JavaHTTPServer(Socket c) {
        connect = c;
    }

    public static void main(String[] args) {
        try {
            ServerSocket serverConnect = new ServerSocket(PORT);
            System.out.println("Server started.\nListening for connections
on port : " + PORT + " ...\n");

            // we listen until user halts server execution
            while (true) {
                JavaHTTPServer myServer = new
JavaHTTPServer(serverConnect.accept());

                if (verbose) {
                    System.out.println("Connection opened. (" + new Date()
+ ")");
                }

                // create dedicated thread to manage the client connection
                // problem for later
                ExecutorService service = Executors.newCachedThreadPool();
                ThreadPoolExecutor threadPool = (ThreadPoolExecutor)
service;
//                threadPool.submit();
                service.execute(
                        () -> {
                            //GET, HEAD, PUT
                            System.out.println("Do something");
                        });
                service.shutdown();


                Thread thread = new Thread(myServer);
                thread.start();
            }

        } catch (IOException e) {
            System.err.println("Server Connection error : " +
e.getMessage());
        }
    }

    @Override
    public void run() {
        // we manage our particular client connection
        BufferedReader in = null;
```

```java
        PrintWriter out = null;
        BufferedOutputStream dataOut = null;
        String fileRequested = null;
        String method = "";

        try { //try (Socket socket = socket) {
            // we read characters from the client via input stream on the
socket
            in = new BufferedReader(new
InputStreamReader(connect.getInputStream()));
            // we get character output stream to client (for headers)
            out = new PrintWriter(connect.getOutputStream());
            // get binary output stream to client (for requested data)
            dataOut = new BufferedOutputStream(connect.getOutputStream());

            connect.setSoTimeout(10000); //
            // timeout
            // loop

            // get first line of the request from the client
            String input = in.readLine();
            // we parse the request with a string tokenizer
            StringTokenizer parse = new StringTokenizer(input);
            method = parse.nextToken().toUpperCase(); // we get the HTTP
method of the client
            // we get file requested
            fileRequested = parse.nextToken().toLowerCase();

            // object oriented solution
            // we support only GET and HEAD methods, we check
            if (!method.equals("GET") && !method.equals("HEAD")) {
                if (verbose) {
                    System.out.println("501 Not Implemented : " + method +
" method.");
                }

                // we return the not supported file to the client
                File file = new File(WEB_ROOT, METHOD_NOT_SUPPORTED);
                int fileLength = (int) file.length();
                String contentMimeType = "text/html";
                //read content to return to client
                byte[] fileData = readFileData(file, fileLength);

                // we send HTTP Headers with data to client
                Header fiveOhOne = new Header(out, serverName + "501","501
Not implemented", contentMimeType, fileLength);
                // file
                dataOut.write(fileData, 0, fileLength);
                dataOut.flush();
            } else {
                // GET or HEAD method
                if (fileRequested.endsWith("/")) {
                    fileRequested += DEFAULT_FILE.replace("\"", "");
                }

                // if (file exits) {
                File file = new File(WEB_ROOT, fileRequested);
                int fileLength = (int) file.length();
                String content = getContentType(fileRequested);

                if (method.equals("GET") || method.equals("HEAD")) { //
GET method so we return content
                    byte[] fileData = readFileData(file, fileLength);

                    // print header method or request object DONE
                    // response object

                    // send HTTP Headers
                    Header twoOhOh = new Header(out, serverName + "200",
"200 OK", content, fileLength);
                    if (method.equals("GET")) {
                        dataOut.write(fileData, 0, fileLength);
                        dataOut.flush();
                    }
                }

                if (verbose) {
                    System.out.println("File " + fileRequested + " of type
" + content + " returned");
                }

            }

        } catch (FileNotFoundException fnfe) {
            try {
                fileNotFound(out, dataOut, fileRequested, method);
            } catch (IOException ioe) {
                System.err.println("Error with file not found exception :
" + ioe.getMessage());
            }

        } catch (IOException ioe) {
            System.err.println("Server error : " + ioe);
        } finally {
            try {
                in.close();
                out.close();
                dataOut.close();
                connect.close(); // we close socket connection
            } catch (Exception e) {
                System.err.println("Error closing stream : " +
e.getMessage());
            }

            if (verbose) {
                System.out.println("Connection closed.\n");
            }
        }
    }

    private byte[] readFileData(File file, int fileLength) throws
IOException {
        FileInputStream fileIn = null;
        byte[] fileData = new byte[fileLength];

        try {
            fileIn = new FileInputStream(file);
            fileIn.read(fileData);
        } finally {
            if (fileIn != null)
                fileIn.close();
        }

        return fileData;
    }

    // return supported MIME Types
    private String getContentType(String fileRequested) {
        if (fileRequested.endsWith(".htm") ||
fileRequested.endsWith(".html"))
            return "text/html";
        else if (fileRequested.endsWith(".js")) {
            return "text/javascript";
        } else if (fileRequested.endsWith(".pdf")) {
            return "application/pdf";
        } else if (fileRequested.endsWith(".css")) {
            return "text/css";
        } else if (fileRequested.endsWith(".json")) {
            return "application/json";
        } else
            return "text/plain";
    }

    private void fileNotFound(PrintWriter out, OutputStream dataOut,
String fileRequested, String method) throws IOException {
        File file = new File(WEB_ROOT.toString(),
FILE_NOT_FOUND.replace("\"", ""));
```

```
        int fileLength = (int) file.length(); //long
        String content = "text/html";
        byte[] fileData = readFileData(file, fileLength); //only works
with int

        Header fourOhFour = new Header(out, serverName + "404", "404 File
Not Found", content, fileLength);

        if (method.equals("GET")) {
            dataOut.write(fileData, 0, fileLength);
            dataOut.flush();
        }

        if (verbose) {
            System.out.println("File " + fileRequested + " not found");
        }
    }
}

// TODO: 2020-02-06
//  response object
//  header
//  read multiple text files(html, css, js, pdf)
//  1 or more image file type
//  post / input parameters in URL get json
//  (index)create one or more static web pages with info about what the
server can do and show access to json info

// TODO: 2020-02-11 change name for serverName

// TODO: 2020-02-11 modules
```