```
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;
/**
        NAME        : ADEREMI, Dayo Owolabi
        MATRIC NO   : 189074114
*/

public class MultiThreadedJavaHTTPServer implements Runnable{

        static final File WEB_ROOT = new File(".");
        static final String DEFAULT_FILE = "index.html";
        static final String FILE_NOT_FOUND = "404.html";
        static final String METHOD_NOT_SUPPORTED = "not_supported.html";
        // port to listen connection
        static final int PORT = 8080;

        // 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("Connecton opened. (" + new
Date() + ")");
                                }

                                // create dedicated thread to manage the client connection
                                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; PrintWriter out = null; BufferedOutputStream
dataOut = null;
                String fileRequested = null;
```

```
                try {
                        // 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());

                        // 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);
                        String method = parse.nextToken().toUpperCase(); // we get the
HTTP method of the client
                        // we get file requested
                        fileRequested = parse.nextToken().toLowerCase();

                        // 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
                                out.println("HTTP/1.1 501 Not Implemented");
                                out.println("Server: Java HTTP Server from SSaurel :
1.0");
                                out.println("Date: " + new Date());
                                out.println("Content-type: " + contentMimeType);
                                out.println("Content-length: " + fileLength);
                                out.println(); // blank line between headers and content,
very important !
                                out.flush(); // flush character output stream buffer
                                // file
                                dataOut.write(fileData, 0, fileLength);
                                dataOut.flush();

                        } else {
                                // GET or HEAD method
                                if (fileRequested.endsWith("/")) {
                                        fileRequested += DEFAULT_FILE;
                                }

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

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

                                        // send HTTP Headers
                                        out.println("HTTP/1.1 200 OK");
                                        out.println("Server: Java HTTP Server from SSaurel
: 1.0");
                                        out.println("Date: " + new Date());
                                        out.println("Content-type: " + content);
                                        out.println("Content-length: " + fileLength);
```

```
                                        out.println(); // blank line between headers and
 content, very important !
                                        out.flush(); // flush character output stream
 buffer

                                        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);
                        } 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
                        return "text/plain";
        }

        private void fileNotFound(PrintWriter out, OutputStream dataOut, String
```

```
 fileRequested) throws IOException {
                 File file = new File(WEB_ROOT, FILE_NOT_FOUND);
                 int fileLength = (int) file.length();
                 String content = "text/html";
                 byte[] fileData = readFileData(file, fileLength);

                 out.println("HTTP/1.1 404 File Not Found");
                 out.println("Server: Java HTTP Server from SSaurel : 1.0");
                 out.println("Date: " + new Date());
                 out.println("Content-type: " + content);
                 out.println("Content-length: " + fileLength);
                 out.println(); // blank line between headers and content, very important !
                 out.flush(); // flush character output stream buffer

                 dataOut.write(fileData, 0, fileLength);
                 dataOut.flush();

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

 }
```