```java
package Serveur.server;

import java.io.FileInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;

import Serveur.maths.Fonction;

public class ServerTriangulation implements ConnectionObserver, Runnable {
    public static final int DEFAULT_TIMEOUT = 2000;
    public final int port;
    private ServerSocket socket;
    private volatile boolean active = false;
    protected ArrayList<ClientTriangulation> clients = new
ArrayList<ClientTriangulation>();
    private ConnectionListener listener;
    private Thread listenerThread;

    private String codeHtml[];
    private String entete = "HTTP/1.1 200 OK\r\n"
            + "Content-Type: text/html\r\n" + "Connection: close\r\n\r\n";

    public Thread thread;

    public ServerTriangulation(int port) throws IOException {
        initCodeHtml();

        this.port = port;
        socket = new ServerSocket(port);
        socket.setSoTimeout(DEFAULT_TIMEOUT);

        thread = new Thread(this);
        listener = new ConnectionListener(socket, this);
        listenerThread = new Thread(listener);
    }

    private void initCodeHtml() throws IOException {
        ArrayList<String> morceaux = new ArrayList<String>();

        FileInputStream in = new FileInputStream(
                "C:\\Users\\Kirito\\Documents\\TIPE\\site\\index.html");
        StringBuffer strBuf;
        String code = "";
        byte[] buf = new byte[1024];
        int len, i;

        do {
            len = in.read(buf);

            if (len > 0) {
                strBuf = new StringBuffer();

                for (i = 0; i < len; i++)
                    strBuf.append((char) buf[i]);

                code += strBuf.toString();
            }
        } while (len == buf.length);

        in.close();
        int index = 0, startIndex = 0;

        do {
            index = code.indexOf("[{", index);

            if (index != -1) {
                String morceau = code.substring(startIndex, index);

                index = code.indexOf("}]", index) + 2;
                morceaux.add(morceau);
                startIndex = index;
            }
        } while (index != -1);

        morceaux.add(code.substring(startIndex));

        codeHtml = morceaux.toArray(new String[morceaux.size()]);
        morceaux.clear();
    }

    @Override
    public void newClient(Socket socket) {
        try {
            socket.setSoTimeout(DEFAULT_TIMEOUT);

            ClientTriangulation client = new ClientTriangulation(socket, this);
            client.start();
            clients.add(client);

            System.out.print(
                    "Le client " + socket.getInetAddress() + " est connecté. ("
                            + clients.size() + " clients au total)");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public String constructHtml() {
        String code = "";

        for (int i = 0; i < codeHtml.length; i++) {
            code += codeHtml[i];

            if (i < codeHtml.length - 1) {
                code += Math.random();
            }
        }

        return entete + code;
    }

    @Override
    public void run() {
        active = true;

        while (active) {

        }

        listener.close();

        while (listener.isRunning());

        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        clients.clear();
        active = false;
    }

    public void start() {
        listenerThread.start();
        thread.start();
    }

    public boolean isActive() {
        return active;
    }

    public void close() {
        active = false;
    }

    public synchronized Object doModifications(Fonction f) {
        return f.method();
    }
}
```