SOEN 6441-WW: Advanced Programming Practices
Winter 2020 - Project Description

Dr. Constantinos Constantinides, P.Eng.

January 20, 2020

Contents

1 Introduction

Posted: Monday 20 January.

Due: Monday 30 March by 16:30.

You must form a team of 3-4 people. You will demonstrate your code and the functionality during the a demo in class. Please note that each person in a team should be prepared to do a presentation of any part of the code and answer questions.

2 Ground rules

This an assessment exercise. You may not seek any assistance while expecting to receive credit. **You must work strictly within your team and seek no assistance for this project (from the instructor, fellow classmates and other teams or external help)**. Failure to do so will result in penalties or no credit.

3 Your assignment

Our base application is accompanying this document.

4 Logging

The various parts of logging are described in the following subsections.

4.1 Logging connection requests and successful connections

Define an aspect `Logging` that will record **connection requests** and successful **connections**, keeping track of the clients who are logged. For the scenario below

```
Server host = new Server("My Server");

Client jack = new Client("Jack", "evil.net");
Client jill = new Client("Jill", "evil.net");
Client jekyll  = new Client("Jekyll", "student.net");
Client hyde = new Client("Hyde", "evil.net");
```

```
jack.connect(host);    // accommodate
jill.connect(host);    // accommodate
jekyll.connect(host);  // accommodate
```

the output should be as follows:

```
CONNECTION REQUEST >>> Jack@evil.net requests connection to My Server.

Connection established between Jack@evil.net and My Server.
Clients logged in: [Jack@evil.net]


CONNECTION REQUEST >>> Jill@evil.net requests connection to My Server.

Connection established between Jill@evil.net and My Server.
Clients logged in: [Jack@evil.net, Jill@evil.net]


CONNECTION REQUEST >>> Jekyll@student.net requests connection to My Server.

Connection established between Jekyll@student.net and My Server.
Clients logged in: [Jack@evil.net, Jill@evil.net, Jekyll@student.net]
```

4.2 Logging disconnects

Extend `Logging` to record **disconnects**. For example, for the statement below

```
...
jekyll.disconnect(host);   // accommodate
```

the output should be as follows:

```
Connection broken between Jekyll@student.net and My Server.
Clients logged in: [Jack@evil.net, Jill@evil.net]
```

5 Authentication

5.1 Capturing suspicious calls

Let us now consider that any request to obtain all current clients breaks the terms of service and any client who makes such a request should be disconnected and their address should be blacklisted.

Define aspect `Authentication` that will capture any **suspicious calls** and proceed to 1) disconnect the client, and 2) blacklist the client's domain (which will subsequently deny access to any client from the same domain). For example, for the statement below

```
jack.getAllClients();   // suspicious: Do not accommodate, blacklist and disconnect
```

the output should be

```
WARNING >>> Suspicious call from evil.net: call(Server.getClients())

Connection broken between Jack@evil.net and My Server.
Clients logged in: [Jill@evil.net]
```

Why is `Jill` still a client? After all, her address has been blacklisted. Indeed her address has been blacklisted, but this occurred only after she logged in to the server (due to the actions of another evil.net client). In fact, `Jill` was never disconnected. We will have to fix this as follows: From this moment onwards (i.e. the moment Jill's address has been blacklisted) once she sends any message, not only it will not be accommodated, but Jill will be disconnected. For example, for the following scenario

```
jill.getAllClients();   // already blacklisted; Do not accommodate and disconnect
jill.getAllClients();   // not accommodated
jill.disconnect(host);  // not accommodated
```

the output should be as follows:

```
Connection broken between Jill@evil.net and My Server.
Clients logged in: []
```

6 Revisiting Logging

6.1 Ensuring an unconditional logging of connection requests

We want to extend `Logging` to guarantee that even in the presence of a blacklisted address, connection requests made by clients from those addresses will still be logged. For example, for the following scenario

```
jill.connect(host);   // recorded, but not accommodated
hyde.connect(host);   // recorded, but not accommodated
host.getClients();
```

the output should be as follows:

```
CONNECTION REQUEST >>> Jill@evil.net requests connection to My Server.

CONNECTION REQUEST >>> Hyde@evil.net requests connection to My Server.
Clients: []
```

7 Putting everything together

For the scenario below:

```
Server host = new Server("My Server");

Client jack = new Client("Jack", "evil.net");
Client jill = new Client("Jill", "evil.net");
Client jekyll  = new Client("Jekyll", "student.net");
Client hyde = new Client("Hyde", "evil.net");

jack.connect(host);      // accommodate
jill.connect(host);      // accommodate
jekyll.connect(host);    // accommodate
jekyll.disconnect(host); // accommodate

jack.getAllClients();    // suspicious: Do not accommodate, blacklist and disconnect
```

```
jill.getAllClients();    // already blacklisted; Do not accommodate and disconnect
jill.getAllClients();    // not accommodated
jill.disconnect(host);   // not accommodated

jill.connect(host);      // recorded, but not accommodated

hyde.connect(host);      // recorded, but not accommodated

host.getClients();
```

the output should be

```
CONNECTION REQUEST >>> Jack@evil.net requests connection to My Server.

Connection established between Jack@evil.net and My Server.
Clients logged in: [Jack@evil.net]


CONNECTION REQUEST >>> Jill@evil.net requests connection to My Server.

Connection established between Jill@evil.net and My Server.
Clients logged in: [Jack@evil.net, Jill@evil.net]


CONNECTION REQUEST >>> Jekyll@student.net requests connection to My Server.

Connection established between Jekyll@student.net and My Server.
Clients logged in: [Jack@evil.net, Jill@evil.net, Jekyll@student.net]


Connection broken between Jekyll@student.net and My Server.
Clients logged in: [Jack@evil.net, Jill@evil.net]


WARNING >>> Suspicious call from evil.net: call(Server.getClients())

Connection broken between Jack@evil.net and My Server.
```

```
Clients logged in: [Jill@evil.net]



Connection broken between Jill@evil.net and My Server.
Clients logged in: []



CONNECTION REQUEST >>> Jill@evil.net requests connection to My Server.

CONNECTION REQUEST >>> Hyde@evil.net requests connection to My Server.
Clients: []
```

8 What to produce and what to submit

You must produce **two aspect definitions**: `Logging` and `Authentication`. Please print them out, and staple them together. Have your project date stamped by our CSE receptionist and place it in my mailbox (in main CSE office).