Reflective Application Project

A project to showcase what we learned in JavaRef classes

Project's structure

```
src
├──client
│   ├──amateur
│   └──programmer
├──server
│   ├──application
│   ├──BRi
│   └──user
└──services
```

- Client - Contains a package for the Amateur app and another for the Programmer app.
- Server - Server folder. *duh*. Contains everything the server need to run correctly.
  - application - Contains only `Server.java`.
  - BRi - The folder for everything about Services and BRi bullshit.
  - user - Folder for the `User` Class and the Database.
- Services - Where services are stocked before putting them on the FTP server.

How the code works

It's not that I'm lazy, but I went into all those beautiful comments so I really think I don't need to explain a lot here.

If I have to describe the overall working process, it's very simple. We choose to use a `Singleton` to make the database shared between all the threads. And because of those nasty threads, we had to stick `synchronized` onto the database declaration method just in case two threads start simultaneously.

```
public synchronized static UserDB getDatabase()
```

We also ditch the Vector for a Collection, for obvious reason that are stated directly in the code.

The rest is pretty straight forward, two threads listen for connections, one for the Amateurs and the other for the Programmers. When a connexion is made, the work is handled by a self-destructive thread.

For most of the part, we used the 4th class' project as a starting base.

What works ?

Not as much as we intended to, but given the recent circumstances, we're not gonna complain.

- The Singleton works as intended so it's nice to have that.
- Users can login with the Amateur app for the Amateurs and same goes for the Programmers.
- Amateurs can't log as Programmers, but Programmers can use the Amateur app.
- A User can't log twice in the same time, to prevent obvious shit with the database.
- Services management works, fortunately... A Programmer can add/remove Services and change its FTP url.

What does't work

- Modifying Services.
- Everything that was optional, like the messaging shenanigans. It was optional so... ¯_(ツ)_/¯
- XML processing, not enough time to do it...

Authors

- **Marius Vallas** :
  - *Everything that **is not** user management and login* (thread and stuff...)
  - *Git management*
  - *Final code refactoring and code commenting*
- **Nils Carron** :
  - *Everything that **is** user management and login*
  - *Debugging*
  - *Some code refactoring*

License

Postscript

> **J'ai pas oublié l'interface cette fois**

Marius Vallas