Lopez Clément

Robert Guillaume

1A

Projet CPOA

# Table des matières

[Table des matières 1](#_Toc472705395)

[Introduction 2](#_Toc472705396)

[Existant 3](#_Toc472705397)

[Solutions techniques 4](#_Toc472705398)

[Patron 4](#_Toc472705399)

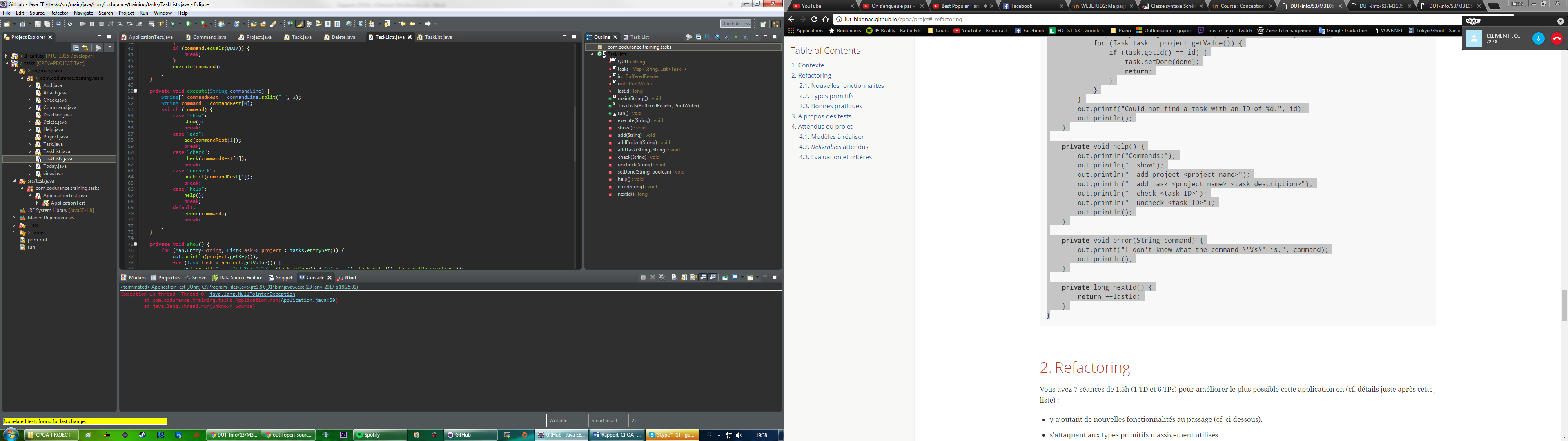
[Fonctionnalités 4](#_Toc472705400)

# Introduction

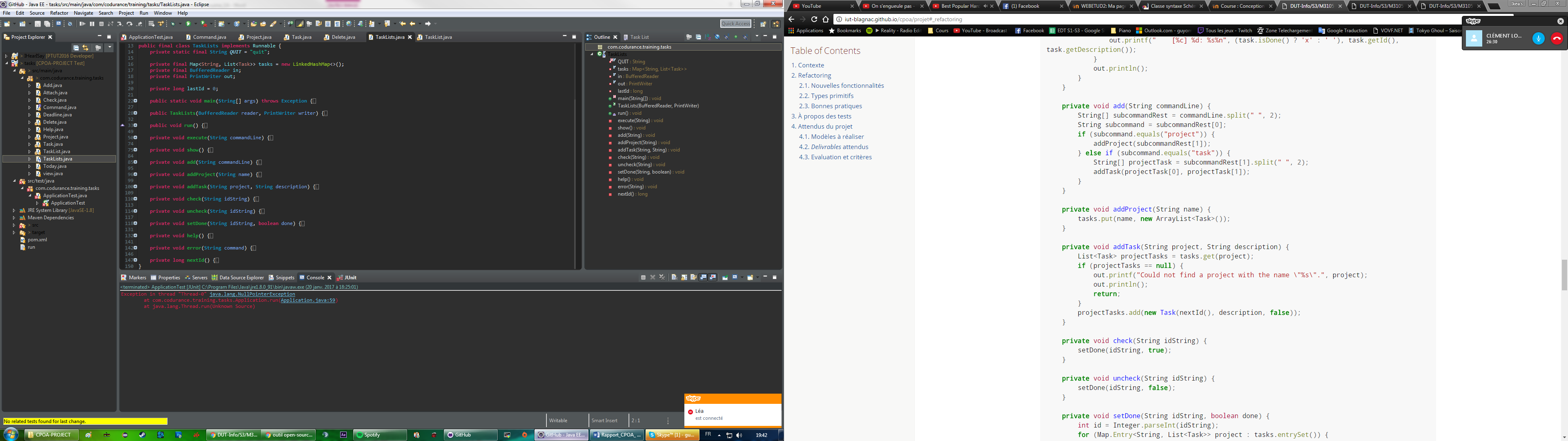
Ce projet permet de mettre en place les bonnes pratiques et méthodes de programmation apprises lors des cours de CPOA. Pour cela nous avons à notre disposition une application qui doit être améliorée. Cependant cette application n’est pas optimisée pour l’extension. Notre but ici est d’y ajouter des fonctionnalités et de permettre d’avoir un code qui soit expliqué et ouvert aux futures extensions. Nous devons permettre aux prochains utilisateurs de comprendre très rapidement comment marche l’application.

# Existant

En début de projet nous avons dû nous familiariser avec l’existant. Le code permettant de gérer les différentes commandes est un switch :



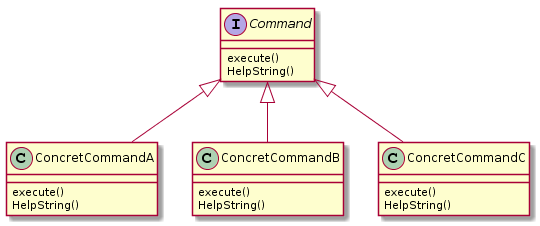
Par conséquent à chaque ajout d’une nouvelle fonctionnalité il nous faut agrandir ce switch. Cela peut vite devenir très long et compliqué. Aussi, toutes les méthodes utilisées sont dans cette classe.



Pour finir, ces méthodes de sont pas commentées c’est pourquoi il a fallu du temps avant de comprendre comment elles fonctionnaient.

# Solutions techniques

## Patron

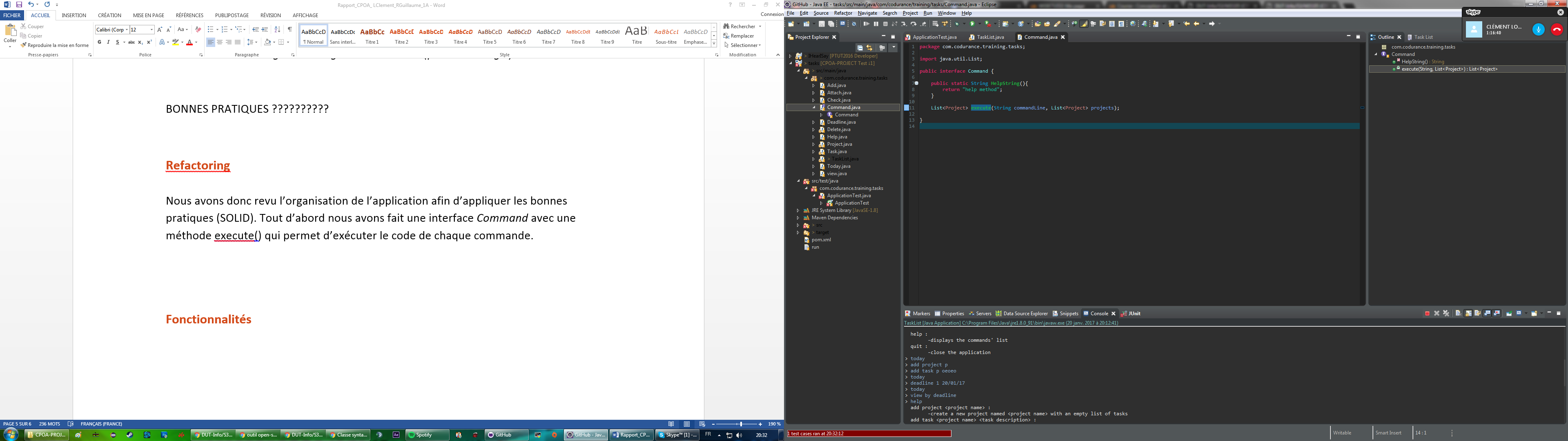
Afin d’optimiser l’extension et l’utilisation de cette application nous avons utilisé le patron Stratégie :

*Figure 1 : Diagramme de classe (patron stratégie)*

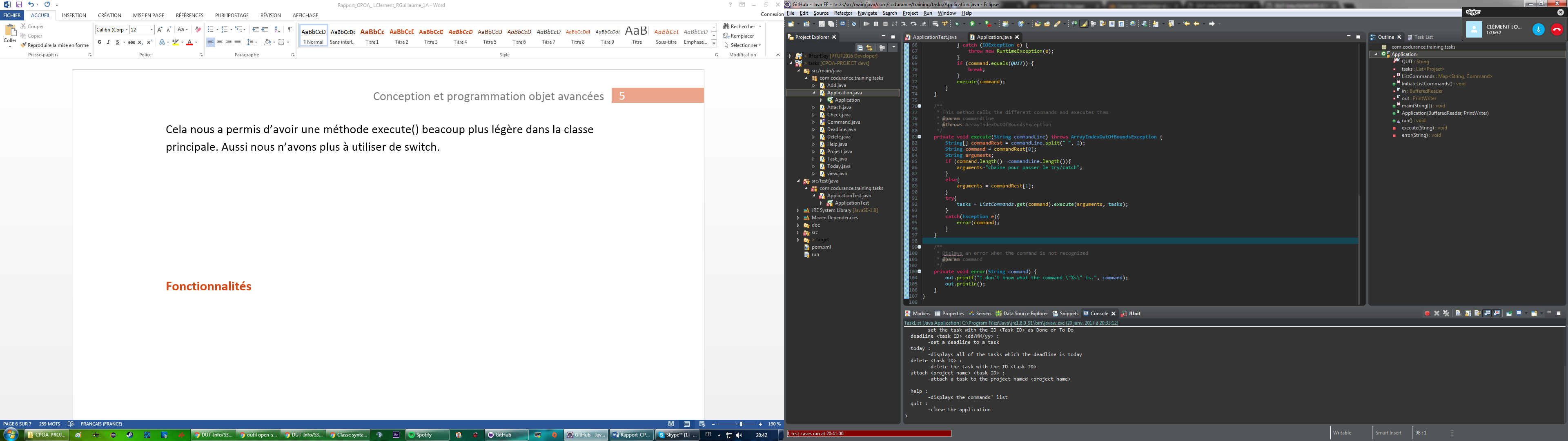
BONNES PRATIQUES ??????????

## Re-factoring

Nous avons donc revu l’organisation de l’application afin d’appliquer les bonnes pratiques (SOLID). Tout d’abord nous avons fait uneinterface *Command* avec une méthode execute() qui permet d’exécuter le code de chaque commande.



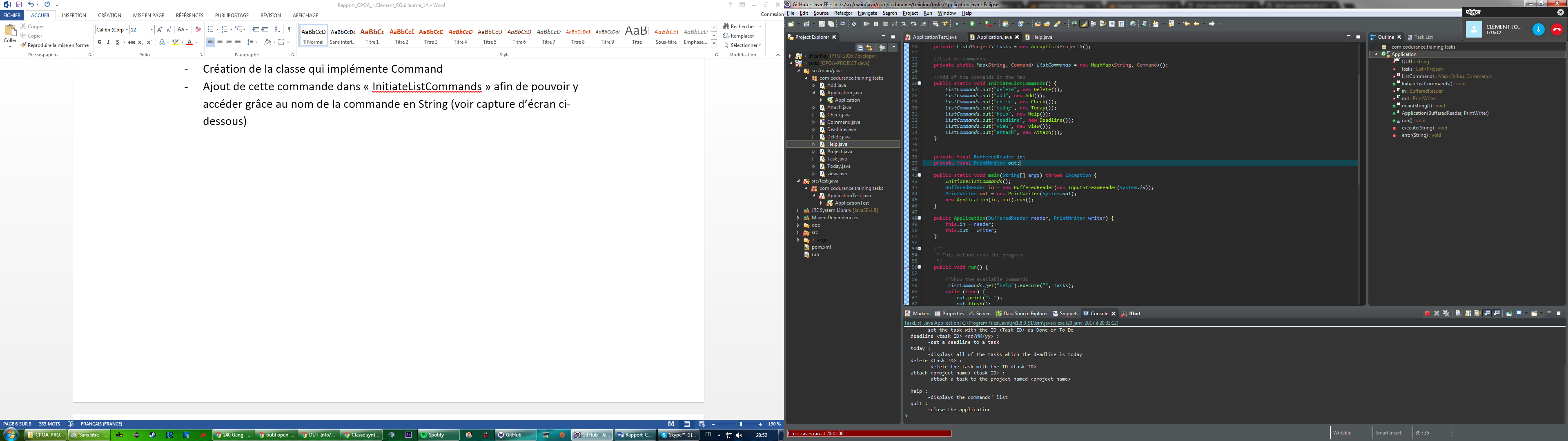
Cela nous a permis d’avoir une méthode execute() beacoup plus légère dans la classe principale. Aussi nous n’avons plus à utiliser de switch. Grâce à cette réorganisation, on pourra ajouter autant de commande que l’on veut, sans modifier une seule ligne de cette méthode.



## Fonctionnalités

Grâce à cette réorganisation expliquée précédemment, lorsque l’on veut ajouter une nouvelle commande on procède en 2 étapes :

* Création de la classe qui implémente Command
* Ajout de cette commande dans « InitiateListCommands » afin de pouvoir y accéder grâce au nom de la commande en String (voir capture d’écran ci-dessous)



COMMANDES ?????

## Conclusion