Nycola Plaisance

Conception et Développement Orientés Objet

420-C46-JO

Groupe : 01

**Rapport Simulateur de vol**

Travail remis à :

Charles Jacob

Cégep Régional de Lanaudière à Joliette

Date de remise : 31 mai 2021

# Application des patrons GOF

## MVC

Afin d’appliquer le patron MVC j’ai séparé mes classes en fonction de leur rôle, c’est-à-dire Modèle, Vue et Contrôleur. Le modèle représente l’ensemble des classes qui sont chargé de représenter de réelle chose tel que des aéroports, des aéronefs et ainsi de suite ou encore des choses un peu plus surréelles tel que les états des aéronefs. Le contrôleur lui se charge d’effectuer les traitements ordonner par la vue sur le modèle. La vue, elle, se charge principalement de fournir une interface pour l’utilisateur et de fournir au contrôleur le « input » de l’utilisateur.

## Patron « Factory » et le singleton

Afin d’appliquer le patron usine ainsi que le patron singleton j’ai écrit une classe qui sera chargé de créer tous types de clients et celle-ci sera un singleton par souci d’optimisation. En gros, la présence de mon usine est justifiée par le besoin d’instancier plusieurs classes (les clients) relativement semblable. Donc cette tache devait bien être exécuter par une classe et c’est ici que l’usine devient utile. Ensuite, j’ai pris la décision de faire de cette usine un singleton puisqu’il semblait logique de le faire ainsi vue que dans le cas contraire, un nouvel objet d’usine à client serait instancié à chaque fois que nous aurions besoin de créer un client et je vois ceci comme une perte de temps (et d’opérations), si on ne détruit simplement pas cet objet on peut alors le réutiliser !

## Patron Facade

Afin d’appliquer le patron façade j’ai fait bien attention à ne faire aucune communication entre les classes derrières les façades (c’est-à-dire les classes Scenario, Controller\_Simulation et View\_Simulation

## Patron Stratétat

# Décision de design (GRASP)

# Application de mes apprentissages