POA – Laboratoire Squadron

# Objectif

Le but de ce laboratoire est d’implémenter en C++ une application permettant de calculer la consommation en carburant d’une escadrille de vaisseaux Star Wars.

Il faut donc implémenter un moyen de représenter différents modèles de vaisseaux et de pouvoir créer une escadrille composée d’un certain nombre de vaisseaux. L’application doit également permettre la modification de cette escadrille, l’assignement d’un chef et bien sûr, le calcul de la consommation d’un vaisseau et d’une escadrille.

Pour ce laboratoire, l’utilisation des structures de données de la librairie standard est interdite.

# Conception

## Classe Ship

La classe **Ship** est une classe abstraite qui représente un vaisseau de notre application. Elle permet de factoriser les notions communes à tous les modèles de vaisseau. Notamment le nom, le numéro de série et la formule de calcul de consommation de carburant.

## Sous-Classes de Ship

Les vaisseaux concrets sont des sous-classes de **Ship** et implémente les méthodes virtuelles *getModel, getModelSpeedMax* et *getModelWeight* retournant des informations en fonctions de leur modèle. De cette manière les vaisseaux respectent l’interface nécessaire de **Ship** mais n’ont pas besoin de stocker des valeurs inutiles qui seraient redondante à tous les vaisseaux du même modèle. De plus, cette architecture nous permet de partager des caractéristiques avec des sous-modèles si besoin.

Tous les vaisseaux concrets on compteur de numéro de série auto-incrémenté à l’instanciation d’un objet.

## Classe CargoShip

Cette sous-classe de **Ship** est également abstraite et représente la sous-population des vaisseaux qui peuvent transporter de la marchandise. Via la variable *currentCapacity* il est possible de gérer la capacité du cargo.

## Classe Squadron

L’escadrille est modélisée par la classe **Squadron** qui utilise une classe **Member** comme nœud de liste chainée de pointeur sur vaisseau. Nous avons décidé que **Member** soit une classe interne car elle concerne uniquement Squadron et son implémentation de liste chainée. Des vaisseaux peuvent être ajouter ou supprimer de cette liste via des méthodes ou les opérateurs (+, -, +=, -=).

# Test de la classe Ship

|  |  |  |
| --- | --- | --- |
| **Test effectué** | **Résultat attendu** | **Résultat obtenu** |
| Construction d’un objet | Ok | Ok |
| Construction d’un objet constant | Ok | Ok |
| Modification du nom du vaisseau | Ok | Ok |
| Récupération du nom du vaisseau | Ok | Ok |
| Calcul de la consommation du vaisseau | Ok | Ok |
| Calcul de la consommation du vaisseau avec une vitesse ou une distance négative | Erreur | Erreur |
| Calcul de la consommation du vaisseau avec une vitesse supérieur à sa vitesse maximum | Erreur | Erreur |
| Récupération des caractéristiques du modèle | Ok | Ok |
| Affichage du vaisseau dans un flux | Ok | Ok |
| Le numéro de série du vaisseau est correctement attribué | Ok | Ok |

# Test de la classe CargoShip

|  |  |  |
| --- | --- | --- |
| **Test effectué** | **Résultat attendu** | **Résultat obtenu** |
| Instanciation d’un objet | Ok | Ok |
| Instanciation d’un objet avec un poids de cargaison plus grand que sa capacité maximale | Erreur | Erreur |
| Affichage du vaisseau dans un flux | Ok | Ok |

# Test de la classe Squadron

## Constructeur/Destructeur

|  |  |  |
| --- | --- | --- |
| **Test effectué** | **Résultat attendu** | **Résultat obtenu** |
| Construction d’un objet avec un nom | Ok | Ok |
| Construction d’un objet constant avec un nom | Ok | Ok |
| Construction d’un objet par copie | Ok | Ok |
| Construction d’un objet constant par copie | Ok | Ok |
| Construction d’un objet par copie d’une escadrille constante | Ok | Ok |
| Construction d’un objet constant par copie d’une escadrille constante | Ok | Ok |

## Operateurs

|  |  |  |
| --- | --- | --- |
| **Test effectué** | **Résultat attendu** | **Résultat obtenu** |
| Opérateur d’affectation entre 2 escadrilles | Ok | Ok |
| Affichage dans un flux | Ok | Ok |
| Ajout d’un vaisseau avec l’opérateur + | Ok | Ok |
| Ajout d’un vaisseau déjà dans l’escadrille avec l’opérateur + | Erreur | Erreur |
| Ajout d’un vaisseau avec l’opérateur += | Ok | Ok |
| Ajout d’un vaisseau déjà dans l’escadrille avec l’opérateur += | Erreur | Erreur |
| Suppression d’un vaisseau avec l’opérateur - | Ok | Ok |
| Suppression d’un vaisseau qui n’est pas dans l’escadrille avec l’opérateur - | Erreur | Erreur |
| Suppression d’un vaisseau avec l’opérateur -= | Ok | Ok |
| Suppression d’un vaisseau qui n’est pas dans l’escadrille avec l’opérateur -= | Erreur | Erreur |
| Récupération d’un vaisseau avec l’opérateur [] | Ok | Ok |
| Récupération d’un vaisseau avec l’opérateur [] en dehors des limites de la taille de l’escadrille | Erreur | Erreur |

## Autres

|  |  |  |
| --- | --- | --- |
| **Test effectué** | **Résultat attendu** | **Résultat obtenu** |
| Récupération du nom de l’escadrille | Ok | Ok |
| Ajout d’un chef | Ok | Ok |
| Ajout d’un chef qui ne fait pas partie de l’escadrille | Erreur | Erreur |
| Rétrogradation du chef | Ok | Ok |
| Rétrogradation du chef dans une escadrille où il n’y a pas de chef | Ok | Ok |
| Quand le chef est supprimé de l’escadrille, il n’est plus chef | Ok | Ok |
| Calcul de la consommation de l’escadrille | Ok | Ok |
| Calcul de la consommation du vaisseau avec une vitesse ou une distance négative | Erreur | Erreur |
| Calcul de la consommation du vaisseau avec une vitesse supérieur à sa vitesse maximum | Erreur | Erreur |
| Calcul de la vitesse maximale de l’escadrille | Ok | Ok |
| Calcul du poids total de l’escadrille | Ok | Ok |

# UML

