Forestier Quentin & Herzig Melvyn

POO2 – 13.04.2021

Squadron

Labo2



Table des matières

[Introduction 2](#_Toc69222365)

[Squadron 2](#_Toc69222366)

[Attributs 2](#_Toc69222367)

[Constructeurs 2](#_Toc69222368)

[Surcharges 2](#_Toc69222369)

[Ajout/Suppression d’un vaisseau dans l’escadron 2](#_Toc69222370)

[Exceptions 3](#_Toc69222371)

[Remarques 3](#_Toc69222372)

[Ship 3](#_Toc69222373)

[Attributs 3](#_Toc69222374)

[Classe parente Ship 3](#_Toc69222375)

[Classe Cargo 4](#_Toc69222376)

[Classes finales héritant de Ship 4](#_Toc69222377)

[Classes finales héritant de Cargo 4](#_Toc69222378)

[Constructeurs 4](#_Toc69222379)

[Classe parente Ship 4](#_Toc69222380)

[Classe Cargo 4](#_Toc69222381)

[Classes finales héritant de Ship 4](#_Toc69222382)

[Classes finales héritant de Cargo 4](#_Toc69222383)

[Surcharges 4](#_Toc69222384)

[Remarques 4](#_Toc69222385)

[Tests 5](#_Toc69222386)

[Tests généraux des vaisseaux 5](#_Toc69222387)

[Tests avec vaisseaux constants 5](#_Toc69222388)

[Tests avec vaisseaux non constants 5](#_Toc69222389)

[Tests escadrille 5](#_Toc69222390)

[Annexes 6](#_Toc69222391)

# Introduction

Le but de ce laboratoire est d’approfondir la programmation orientée objet, notamment par l’utilisation des classes et méthodes abstraites en C++. Nous avions pour travail d’implémenter une classe Squadron qui gère des escadrons de vaisseaux. Entre autres, la classe Squadron doit permettre de calculer la consommation totale d’un voyage sur une distance donnée.

# Squadron

## Attributs

Un escadron possède un nom, qui est stocké dans un string.

Un escadron peut avoir un chef, qui doit forcément être membre de l’escadron. Il est stocké au moyen d’un pointeur de vaisseaux.

Pour stocker les vaisseaux appartenant à un escadron, nous utilisons une liste de pointeurs de vaisseaux, allouée dynamiquement. La capacité de base est de 10 vaisseaux, mais augmente si besoin.

La liste a une capacité qui augmente si le nombre de vaisseaux dans l’escadron arrive à la capacité maximale.

## Constructeurs

Un escadron peut être initialisé de deux manières :

* **Par défaut** : Aucun paramètre n’est demandé. Il est possible de donner un nom à l’escadron. La capacité de l’escadron sera allouée à 10 vaisseaux, sans leader.
* **Par copie**: L’utilisateur passe en paramètre un objet Squadron qui sera copié.

## Surcharges

Différents opérateurs ont été surchargés :

* Operator+
* Operator-
* Operator<<
* Operator+=
* Operator-=
* Operator=

Les opérateurs +, +=, -, -= permettent d’ajouter, respectivement d’enlever des vaisseaux d’un escadron. Ces opérateurs se basent sur des méthodes qui demandent un foncteur SquadronOperation.

## Ajout/Suppression d’un vaisseau dans l’escadron

Les méthodes et opérateurs d’ajout et de suppression de vaisseaux font appel à des méthodes privées.

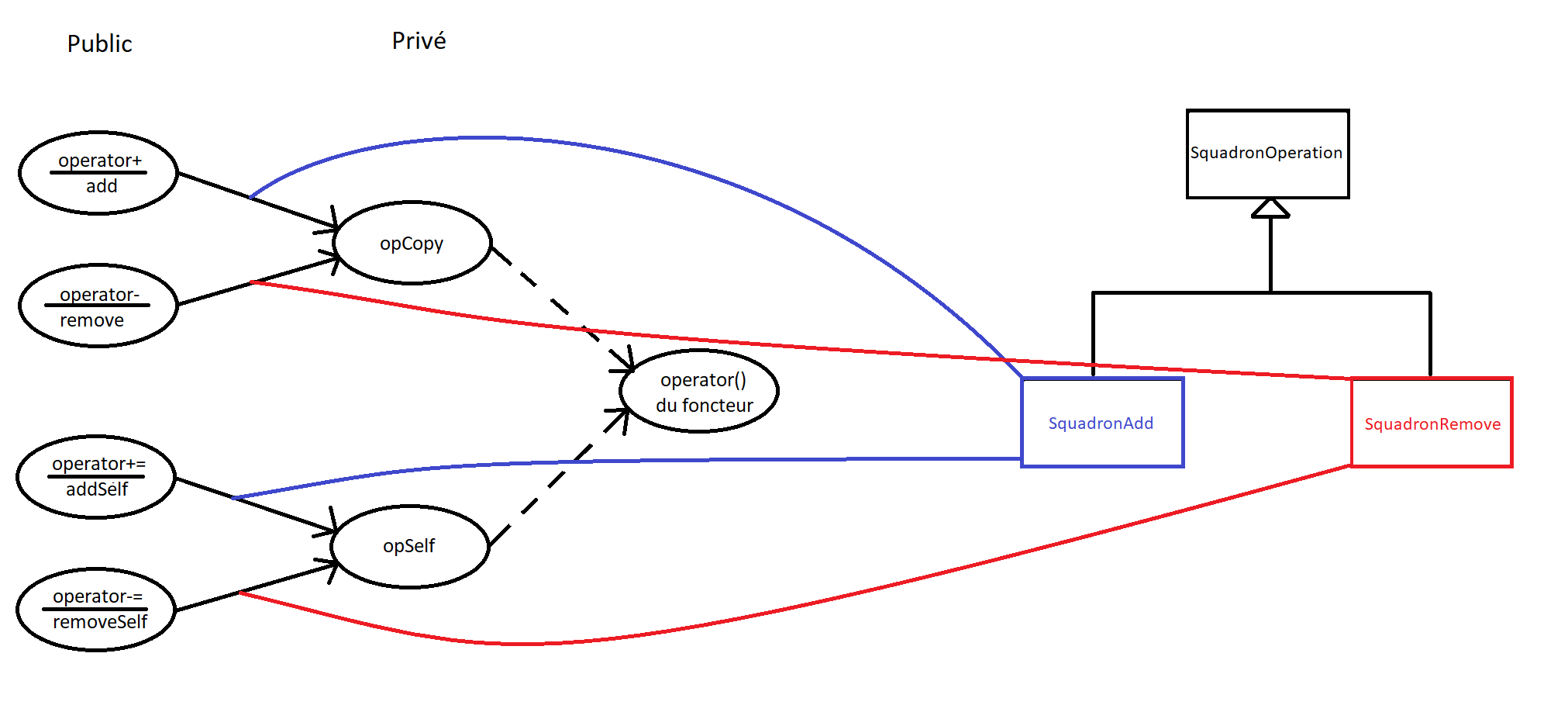
|  |  |  |  |
| --- | --- | --- | --- |
| Objectif | Opérateur | Méthode  publique | Méthode privée |
| Ajoute le vaisseau à une copie de l'escadron. | Operator+ | Add | opCopy (avec foncteur SquadronAdd) |
| Ajoute le vaisseau à l'escadron | Operator+= | AddSelf | opSelf(avec foncteur SquadronAdd) |
| Retire le vaisseau à une copie de l'escadron. | Operator- | Remove | opCopy(avec foncteur SquadronRemove) |
| Retire le vaisseau de l’escadron | Operator-= | RemoveSelf | opSelf(avec foncteur SquadronRemove |

Les méthodes et opérateurs appelant opCopy renverront une copie de l’escadron avec la modification effectuée.

Les méthodes et opérateurs appelant opSelf renverront l’escadron elle-même, et la modification sera effectuée directement sur l’escadron.

Les 2 méthodes ci-dessus demandent un foncteur SquadronOperation qui effectuera la modification.

Voici un schéma qui illustre le fonctionnement des 4 méthodes/opérateurs :



## Exceptions

Les méthodes, manipulant en lecture ou en écriture la liste de vaisseaux, peuvent renvoyer une exception *invalid\_arguement* si le vaisseau passé en paramètre est nul.

## Remarques

La capacité de la liste de pointeurs de Ship peut doubler en capacité lorsqu’elle est pleine, mais ne perdra jamais de capacité.

L’ordre des vaisseaux dans la liste n’est pas conservé lors d’une suppression. Le dernier vaisseau de la liste remplace simplement le vaisseau à supprimer.

# Ship

## Attributs

### Classe parente Ship

Un vaisseau possède un id, stocké dans un unsigned.

Ils ont aussi un nom, qui contient un surnom (facultatif), le modèle et l’id du vaisseau.

### Classe Cargo

Un vaisseau cargo et ceux qui en héritent possèdent un attribut de poids de cargaison, stocké dans un double.

### Classes finales héritant de Ship

Ces classes contiennent plusieurs attributs constants qui les définissent :

* Poids du vaisseau, stocké avec un double
* Vitesse du vaisseau, stocké avec un unsigned
* Modèle du vaisseau, stocké dans un string

Chaque classe finale possède également un compteur, qui définit l’ID du vaisseau.

### Classes finales héritant de Cargo

Un attribut constant est défini pour chaque classe représentant le poids de cargaison maximal supporté par le vaisseau. L’attribut est stocké dans un double.

## Constructeurs

### Classe parente Ship

Il n’existe qu’un constructeur à 2 paramètres. Ces paramètres sont l’id et le nom du vaisseau.

### Classe Cargo

L’unique constructeur de Cargo possède 3 paramètres. L’id, le poids de la cargaison et le nom du vaisseau.

### Classes finales héritant de Ship

Les classes ont un seul constructeur à 1 paramètre, qui est le nom du vaisseau. L’id est obtenu par rapport au nombre de vaisseaux déjà crée de ce type.

### Classes finales héritant de Cargo

Les classes ont un seul constructeur à 2 paramètres, qui sont le poids de cargaison et le nom du vaisseau. L’id est obtenu par rapport au nombre de vaisseaux déjà crée de ce type.

## Surcharges

La classe Ship surcharge l’operator<< afin de pouvoir afficher les vaisseaux facilement.

Toutes les classes finales doivent surcharger les méthodes :

* getWeight()
* getSpeed()
* getModel()

Qui définissent précisément les spécifications du type de vaisseau.

La surcharge renvoie simplement la constante correspondante stockée dans les différents vaisseaux.

## Remarques

Nous avons décidé de supprimer le constructeur de copie et l’opérateur d’affectations, car la gestion d’ID n’était pas spécifiée.

# Tests

## Tests généraux des vaisseaux

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Shuttle | StarDestroyer | TIELN | TIEIN |
| Constructeur vide | Réussite | Réussite | Réussite | Réussite |
| Constructeur avec le nom | ////// | /////// | Réussite | Réussite |
| Constructeur avec le poids | Réussite | Réussite | ////// | ////// |
| Constructeur poids et nom | Réussite | Réussite | ////// | ////// |
| Échec du constructeur de copie | Réussite | Réussite | Réussite | Réussite |
| Échec opérateur affectation | Réussite | Réussite | Réussite | Réussite |

## Tests avec vaisseaux constants

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Shuttle | StarDestroyer | TIELN | TIEIN |
| toStream | Réussite | Réussite | Réussite | Réussite |
| getId | Réussite | Réussite | Réussite | Réussite |
| getModel | Réussite | Réussite | Réussite | Réussite |
| getWeight | Réussite | Réussite | Réussite | Réussite |
| getSpeed | Réussite | Réussite | Réussite | Réussite |
| getEmptyWeight | Réussite | Réussite | ////// | ////// |
| getMaxLoadWeight | Réussite | Réussite | ////// | ////// |
| setName ne compile pas | Réussite | Réussite | Réussite | Réussite |
| setLoadWeight ne compile pas | Réussite | Réussite | ////// | ////// |

## Tests avec vaisseaux non constants

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Shuttle | StarDestroyer | TIELN | TIEIN |
| toStream | Réussite | Réussite | Réussite | Réussite |
| getId | Réussite | Réussite | Réussite | Réussite |
| getModel | Réussite | Réussite | Réussite | Réussite |
| getWeight | Réussite | Réussite | Réussite | Réussite |
| getSpeed | Réussite | Réussite | Réussite | Réussite |
| getEmptyWeight | Réussite | Réussite | ////// | ////// |
| getMaxLoadWeight | Réussite | Réussite | ////// | ////// |
| setName | Réussite | Réussite | Réussite | Réussite |
| setWeight avec un poids valide | Réussite | Réussite | ////// | /////// |
| setWeight avec un poids trop élevé lève invalid\_argument | Réussite | Réussite | ////// | ////// |
| setWeight avec un poids négatif lève invalid\_argument | Réussite | Réussite | ////// | ////// |

## Tests escadrille

|  |  |  |
| --- | --- | --- |
| Description | Const. | Non const. |
| Constructeur vide | Réussite | Réussite |
| Constructeur avec nom | Réussite | Réussite |
| Operateur+ (dans les deux sens) | Réussite | Réussite |
| Méthode add (alternative à l’opérateur +) | Réussite | Réussite |
| Operateur- (dans les deux sens) | Réussite | Réussite |
| Méthode remove (alternative à l’opérateur -) | Réussite | Réussite |
| L’ajout avec copie sur un nullptr lève invalid\_argument | Réussite | Réussite |
| Le retrait avec copie sur un nullptr lève invalid\_argument | Réussite | Réussite |
| Operateur+= | ////// | Réussite |
| Méthode addSelf (alternative à l’opérateur +=) | ////// | Réussite |
| Operateur-= | ////// | Réussite |
| Méthode remove (alternative à l’opérateur -=) | ////// | Réussite |
| L’ajout sur soi-même avec un nullptr lève invalid\_argument | ////// | Réussite |
| Le retrait sur soi-même avec un nullptr lève invalid\_argument | ////// | Réussite |
| setName | ////// | Réussite |
| setLeader d’un vaisseau membre | ////// | Réussite |
| setLeader d’un vaisseau non membre -> sans effet | ////// | Réussite |
| setLeader de nullptr retire le leader. | ////// | Réussite |
| unsetLeader sans leader n’a pas d’effet | ////// | Réussite |
| unsetLeader, sur une escadrille avec leader, le retire | ////// | Réussite |
| Consommation sans vaisseau affiche 0 | Réussite | Réussite |
| La consommation, avec des vaisseaux, est bonne | ////// | Réussite |
| Dépassement de la capacité initiale de 10 vaisseaux, l’escadrille se redimensionne. | ////// | Réussite |
| Operateur << sur escadrille avec vaisseaux | Réussite | Réussite |
| Operateur << sur escadrille sans vaisseau | Réussite | Réussite |

# Annexes

* Code source en PDF.
* UML du projet.