Laboratoire 2: Squadron

Durée du laboratoire: 3 séances. A rendre le jeudi 14 avril 2021, au début du cours.

Depuis que nous avons découvert que les vaisseaux pouvaient avoir des problèmes d'approvisionnement en carburant[1], il nous faut un outil pour déterminer le rayon d'actions de nos escadrilles ou la quantité de carburant nécessaire pour qu'elles puissent effectuer des opérations.

Définir une classe `Squadron` représentant une escadrille de plusieurs vaisseaux. Une escadrille possède un nom et éventuellement un vaisseau chef d'escadrille. Cette classe met à disposition des méthodes permettant de :

- ajouter ou supprimer un vaisseau de l'escadrille de deux manières : en rendant une nouvelle instance de la classe `Squadron` ou en modifiant l'instance manipulée,
- les opérations d'ajout et de suppression doivent également pouvoir se faire à l'aide des opérateurs `+`, `-`, `+=` et `-=` ,
- déterminer la consommation en carburant en fonction des paramètres : poids, distance et vitesse,
- afficher dans un flux (`std::ostream`) l'escadrille (nom, chef d'escadrille et l'ensemble de ses vaisseaux),
- définir le nom de l'escadrille,
- définir ou supprimer le chef d'escadrille.

Vaisseaux

- Chaque vaisseau est d'un modèle donné et peut avoir un surnom spécifique. Le nom identifiant d'un vaisseau est déterminé par son surnom (s'il en possède un) suivi de son modèle ainsi que d'un numéro auto-incrémenté. Par exemple : [TIE/LN #4] pour un TIE sans surnom, ou Executor [Star Dreadnought #1] pour le croiseur de Vador.
- Un vaisseau peut atteindre une vitesse maximale (unité : MGLT) dans l'espace conventionnel selon son modèle. Certains modèles peuvent servir au transport de marchandises et ont alors une cargaison (en tonnes). Le poids de cette cargaison ne peut pas dépasser une certaine valeur selon le modèle de vaisseau.

  Pour simplifier (...), nous supposerons qu'il est possible de déterminer la consommation[2] d'un vaisseau dans une situation normale selon la formule suivante :

$$\text{consommmation} = \frac{\sqrt[3]{\text{poids}}}{2} \times \log 10(\text{poids} \times \text{vitesse}) \times \log 10(\text{distance} + 1)$$

1. Ne pas se référer à l'épisode VIII, qui n'a jamais existé.
2. Consommation totale approximative et arbitraire (incluant la propulsion initiale, les éventuelles corrections de trajectoire, décélération, etc.)

Unités à utiliser

- Vitesse : MGLT
- Poids : tonnes
- Distance : millions de km (mio. km)
- Consommation : tonnes

Vaisseaux et caractéristiques

- Chasseur TIE : modèle TIE/LN, 6 tonnes, pas de chargement, vitesse 100 MGLT,
- Intercepteur TIE : modèle TIE/IN, 5 tonnes, pas de chargement, vitesse 110 MGLT,
- Navette impériale : modèle Lambda-class shuttle, 360 tonnes, chargement maximal de 80 tonnes, vitesse 54 MGLT,
- Star Dreadnought : modèle Super-class Star Destroyer, $9*10^9$ tonnes, chargement maximal $250*10^3$ tonnes, vitesse max 40 MGLT.

Remarques et contraintes

- Les structures de données de la `stl` (i.e. : `vector`, `list`…) sont interdites,
- Pour la classe `Squadron`, les méthodes d'ajout de vaisseaux doivent prendre un pointeur en paramètre (et non pas une référence),
- Utiliser la classe `string` (`<string>`),
- Attention à bien factoriser le code,
- Un même vaisseau peut appartenir à plusieurs escadrilles,
- Pour que le mécanisme de liaison dynamique soit activé sur une méthode, il faut que celle-ci soit définie comme virtuelle (mot clé `virtual`). Cela est obligatoire pour le destructeur dès qu'il y a un héritage.
- Pour indiquer qu'une méthode est abstraite, il faut ajouter = 0 à la fin de sa signature. Par exemple :
  `virtual void methodeAbstraite(int i) const = 0;`
- Base à utiliser pour la déclaration de la classe `Ship` (à compléter) :

```
#ifndef SHIP_HPP
#define SHIP_HPP

#include <ostream>

class Ship;

std::ostream& operator << (std::ostream& os, const Ship& ship);

class Ship
{
public:
  virtual ~Ship();

  /* à compléter */

  virtual std::ostream& toStream(std::ostream& os) const;

private:
   /* à compléter */
};

#endif /* SHIP_HPP */
```

Exemple d'utilisation

Le code ci-dessous,

```
TIE* blackLeader = new TIE();
blackLeader->setNickname("Black leader");
TIE* blackTwo = new TIE();
Shuttle* shuttle = new Shuttle(23.4); // 23.4 tonnes de marchandises

Squadron squad("Black Squadron");
squad += blackLeader;
squad += blackTwo;
squad += shuttle;

squad.setLeader(blackLeader);

cout << squad << endl;
```

doit produire le résutat :

```
Squadron: Black Squadron
  max speed: 54 MGLT
  total weight: 395.40 tons

-- Leader:
Black leader [TIE/LN #1]
  weight : 6.00 tons
  max speed : 100 MGLT

-- Members:
[TIE/LN #2]
  weight : 6.00 tons
  max speed : 100 MGLT

[Lambda-class shuttle #1]
  weight : 383.40 tons
  max speed : 54 MGLT
  cargo : 23.4 tons (max : 80.0)
```