# TP Noté AOOP 2019

A.Robert et O.Goudet

April 2019

## 1 Consignes

Les programmes sont à rédiger en Java. Tous les documents sont autorisés. Vous déposerez l'intégralité des fichiers dans une archive portant votre nom sur l'espace prévu à cet effet sur Moodle. Pensez à tester régulièrement votre code. Deux fichiers "Avions1.csv" et "Avions2.csv" sont fournis sur Moodle. Ils contiennent chacun une liste de passagers et seront utiles pour effectuer des tests.

## 2 Exercice

On cherche à modéliser des avions avec leurs passagers et leurs bagages, via la programmation orientée objet. Pour cela un avion aura une cabine et une soute, qui contiendront respectivement des passagers et des bagages.

- Ecrivez une classe **Bagage** avec deux attributs : poids (Entier) et identifiant (Entier). Implémentez son constructeur avec ces deux variables en paramètre et ajouter une méthode *toString()*.
- Ecrivez une classe **Passager** avec trois attributs : un nom (String), un age (Entier) et un bagage (Bagage). Implémentez son constructeur avec tous les attributs en paramètre et ajouter une méthode *toString()*.
- Créez la classe **Cabine**. Une cabine a les attributs suivants : un nombre de colonnes (Entier) et de rangées (Entier), un nombre de passagers courant (Entier), ainsi qu'un tableau de Passagers à deux dimensions (qui représente les places auxquelles sont assis les passagers). Implémentez son constructeur avec en paramètre seulement les nombres de colonnes et de rangées. Dans ce constructeur on initialisera la taille du tableau de Passagers suivant ces nombres de colonnes et de rangées et le nombre de passagers courant sera mis à la valeur 0. Ajoutez ensuite les méthodes suivantes :

- une méthode *ajouterPassager(Passager passager)* qui ajoute le passager passé en paramètre dans la première place libre du tableau et incrémente le nombre de passagers courant dans la cabine.
- une méthode *enleverPassager(Passager passager)* qui libère la place occupée par le passager dans le tableau et décrémente le nombre de passagers courant dans la cabine.
- une méthode *estRemplie()* qui renvoie *true* si il n'y a plus de places dans la cabine et *false* sinon.

- Créez la classe **Soute**. Une soute a les attributs suivants : une ArrayList de bagages, un poids maximal (Entier) et un poids courant (Entier). Implémentez son constructeur avec en paramètre seulement le poids maximal.
  - Ajouter une méthode *ajouterBagage(Bagage bagage)* qui ajoute un bagage dans la liste de la soute et met à jour son poids courant.
  - Ajouter une méthode *enleverBagage(Bagage bagage)* qui retire un bagage de la liste et met à jour son poids courant.
  - Ajouter une méthode *peutTransporter(Bagage bagage)* qui retourne vrai si le fait d'ajouter ce bagage ne fait pas dépasser le poids maximal autorisé de la soute, et faux sinon.
  - Les bagages peuvent avoir besoin d'être triées du plus lourd au plus léger afin de mieux répartir le poids dans l'avion. Pour cela, faites en sorte que la classe Bagage implémente l'interface Comparable et ajoutez une méthode *trierBagages()* dans la classe Soute qui effectuera ce tri.

- Assemblage de l'avion. Créez une classe abstraite **Avion** qui est composée d'un attribut Cabine, d'un attribut Soute et d'un attribut Modele (String). Ajoutez les méthodes suivantes :
  - une méthode abstraite *decoller()* qui sera instanciée de façon spécifique pour chaque type d'avion concret.
  - une méthode concrète *ajouterPassager(Passager passager)* qui ajoute le passager passé en paramètre en cabine ainsi que le bagage qu'il possède dans la soute. Avant d'ajouter ce passager avec son bagage on vérifiera qu'il y a de la place en cabine **et** que la soute peut transporter son bagage. Si le passager peut être ajouté dans l'avion cette méthode *ajouterPassager()* renverra *true*, *false* sinon
  - une méthode concrète *enleverPassager(Passager passager)* qui retire un passager de la cabine ainsi que son bagage de la soute.
  - une méthode concrète *chargerAvion(String fileName)* qui prend en entrée le chemin d'un fichier contenant une liste de passagers au format csv (comme le fichier "Avion1.csv" dont les champs sont séparés par des virgules), et ajoute successivement ces passagers dans l'avion

tant que la capacité maximale n'est pas atteinte (que ce soit en terme de places dans la cabine ou de poids maximal dans la soute).

- une méthode concrète *sauvegardeListeBagages(String fileName)* qui prend en entrée un chemin de fichier et sauvegarde toute la liste des bagages de l'avion. Juste avant d'effectuer cette sauvegarde on triera la liste des bagages par ordre de poids en utilisant la méthode *trierBagages()* de l'objet Soute.

- Créez deux classes concrètes qui héritent d'Avion :
  - L'**A320** a une cabine qui peut contenir 180 passagers et une soute d'une tonne maximum (1000).
  - Le **Jet** est paramétrable, il prend dans son constructeur un objet de type Cabine ainsi qu'un objet de type Soute dont les paramètres seront définis par l'utilisateur.

  Pour chacun de ces modèles d'avion, implémenter la méthode *decoller()* qui doit afficher en sortie console par exemple pour un A320 : "A320 en vol avec xx passagers", où xx est le nombre de passagers dans l'avion.

- Créez une classe **Aeroport** qui contient la méthode *main()* et permettra d'effectuer les tests suivants :
  - Instanciez un A320 et chargez le fichier "Avion1.csv" contenant une liste de passagers avec la méthode *chargerAvion*.
  - Utiliser la méthode *sauvegardeListeBagages* pour enregistrer un fichier "ListeBagage1.csv" qui contiendra la liste triée des bagages.
  - Effectuez la même chose avec le fichier "Avion2.csv" et enregistrez les bagages dans un fichier "ListBagage2.csv".
  - Instanciez un jet, ajoutez y quelques passagers et sauvegardez la liste des bagages dans un fichier "ListeBagage3.csv".
  - Faites décoller les différents avions...

- Organisez vos classes dans différents packages.

- Bonus : ajoutez une Javadoc