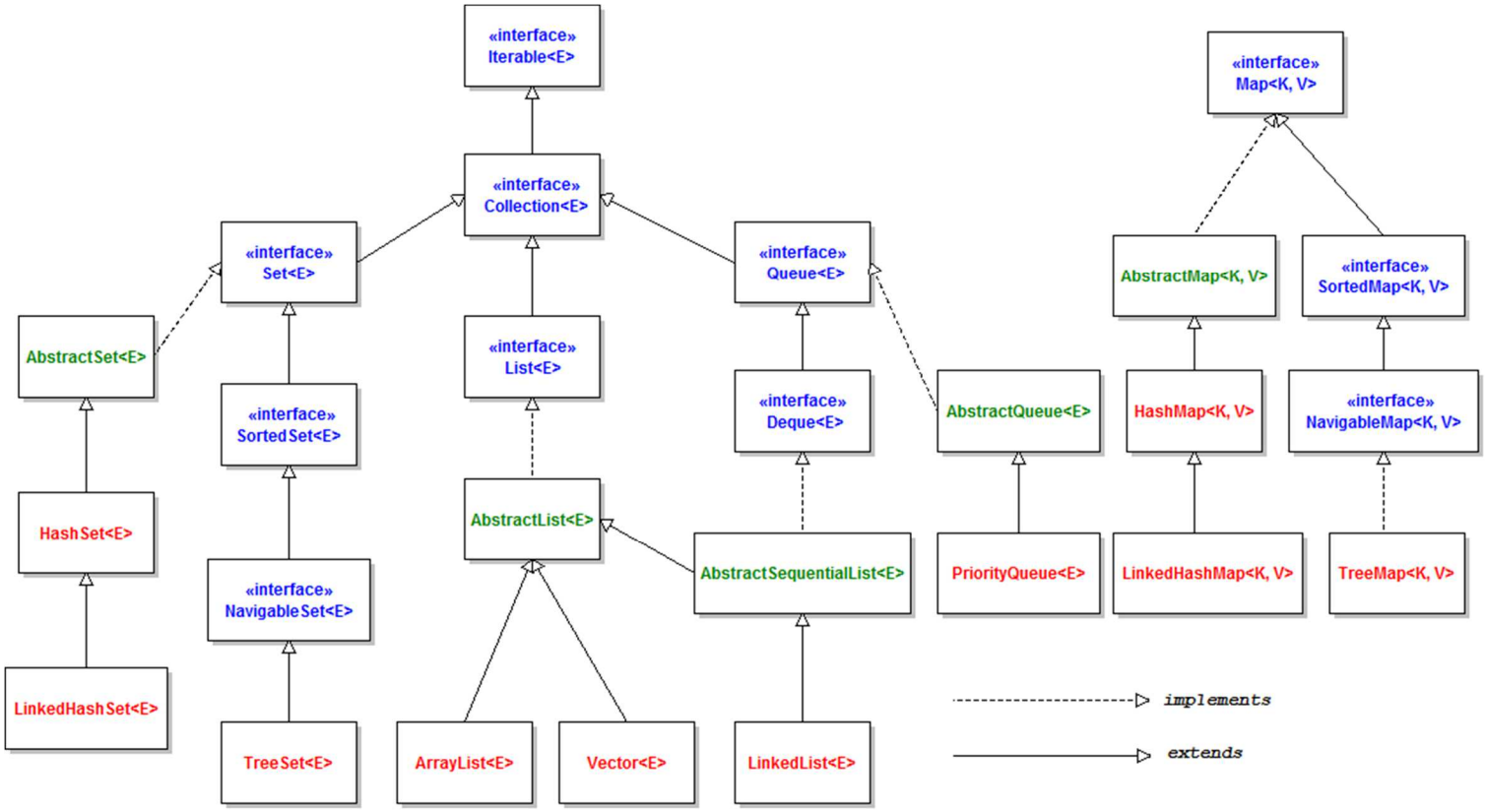
1ère année

Java _ TP n°4

1. Les collections

L'API Collections propose de structurer et de définir un ensemble d'interfaces et de classes de type collection. Ainsi les collections Java sont des objets qui permettent de gérer des ensembles d'objets. Ces collections peuvent avoir différentes caractéristiques : autoriser ou non les doublons, gérer un ordre de tri…
Ces collections sont organisées en une hiérarchie assez complexe.



On trouve 4 grandes familles de collections :

- List : collection d'éléments ordonnés qui accepte les doublons,
- Set : collection d'éléments non ordonnés par défaut, qui n'accepte pas les doublons,
- Map : collection d'éléments sous forme de paires < clé ; valeur >,
- Queue et Deque : collections qui stockent des éléments dans un certain ordre avant qu'ils ne soient extraits pour traitement,

L'API Collections possède deux grandes familles chacune définies par une interface :

- java.util.Collections : pour gérer un groupe d'objets
- java.util.Map : pour gérer des éléments de type paires de clé/valeur

Dans la suite de cet exercice, nous étudierons quelques caractéristiques des ArrayList et des HashMap. Pour cela nous vous proposons différents exercices simples. A vous de chercher par vous-même les

méthodes utiles des interfaces et classes étudiées permettant de répondre le plus simplement aux questions proposées.

1.1. ArrayList

Nous proposons de répondre à une série de questions permettant de manipuler des listes. L'exercice est en 2 temps. Dans un premier temps vous travaillerez avec des listes d'objets simples (String). Dans un second temps, vous utiliserez des listes d'objets plus complexes

ArrayList de String

Pour créer une ArrayList de String, vous devez déclarer et créer la liste:

```
ArrayList<String> list = new ArrayList<>();
```

Répondez aux questions suivantes (les méthodes à utiliser sont soit dans ArrayList soit dans Collections) :

1- Ajouter 5 couleurs au format texte (Bleu, Vert, Rouge, Violet, Orange par exemple) à votre liste.
2- Afficher votre liste
   a. Affichage direct : `System.out.println(list);`
   b. Affichage en parcourant chaque élément
3- Afficher le nombre d'éléments dans la liste.
4- Ajouter une couleur en 1ère position et afficher la liste.
5- Récupérer la couleur en position 3, et l'afficher.
6- Retirer et récupérer la couleur en position 2, et l'afficher.
7- La liste contient-elle la couleur "Blanc" ? et "Rouge" ?
8- Trier la liste et l'afficher.
9- Mélanger la liste et l'afficher.
10- Extraire une portion de la liste (entre les indices 2 et 4 par exemple), et afficher la sous-liste.
11- Echanger 2 éléments de la liste, et afficher la liste.
12- Créer une liste qui soit la fusion de 2 listes, et l'afficher.
13- Vider une liste.

ArrayList de Notes

Créer une classe Note contenant 2 attributs :
- Un nom (String) : le nom de la matière,
- Une valeur (double) : la valeur de la note.

Reprenons quelques questions de l'exercice précédent avec une liste de Note.
1- Ajouter 5 notes à votre liste (["Algo", 12.8], ["Maths", 10.3]…).
2- Afficher votre liste
   a. Affichage direct : `System.out.println(list);`
   b. Affichage en parcourant chaque élément
3- Ajouter une note en 1ère position et afficher la liste.
4- Récupérer la note en position 3, et l'afficher.
5- La liste contient-elle la note ["C++", 11.9] ? et ["Algo", 12.8] ? Attention, ici on s'intéresse au contenu des objets
6- Trier la liste et l'afficher
   a. On veut la trier par ordre alphabétique des noms
   b. Si on veut créer un autre tri (par ordre croissant des valeurs par exemple) ?
7- Extraire une portion de la liste (entre les indices 2 et 4 par exemple), et afficher la sous-liste. Modifier l'une des valeurs de la 1ère liste et réafficher les 2 listes.
8- Créer une liste qui soit la fusion de 2 listes, et l'afficher. Modifier l'un des valeurs d'une liste fusionnée et réafficher la fusion.

1.2. HashMap

Les maps permettent de stocker des données sous forme de paires <clé , valeur>. Les clés et les valeurs sont des objets qui peuvent être null dans les HashMaps.
Créons une HashMap avec des clés de type String et des "double" comme valeurs :

```
HashMap<String, Double> map = new HashMap<>();
```

Nous allons stocker des notes pour des différentes matières dans notre map, au lieu de créer une liste de notes. Répondez aux questions suivantes :

1- Ajouter 3 éléments à la map

```
map.put("Algo", 11.9);
map.put("Maths", 9.5);
map.put("Anglais", 12.3);
```

2- Afficher la valeur associée à la clé "Maths".
3- Afficher la liste des clés.
4- Afficher la liste des valeurs.
5- Afficher la liste des paires <clé, valeur> (à vous d'écrire le code qui le fait).
6- Que se passe-t-il si on écrit maintenant : `map.put("Algo", 13.6);`
7- Que renvoie l'instruction : `map.get("C++");`
8- La clé "C++" est-elle présente dans les entrées ?
9- Est-ce que la valeur 12.3 est contenu dans les valeurs stockées ?

2. Elevage de volailles

Un éleveur gère un ensemble de volailles (poulets et canards) qu'il élève jusqu'à ce qu'ils aient un poids suffisant pour être vendus. Une volaille est caractérisée par son poids et un numéro d'identification reporté sur une bague. Lorsqu'une volaille arrive à la ferme, elle est baguée et enregistrée (chaque numéro doit donc être différent). Il y a 2 sortes de volailles : les poulets et les canards. Le prix au kilo du poulet et du canard est différent (exprimé en euros/kilo). Le poids auquel on abat les volailles est différent pour chaque type (mais est le même pour tous les poulets ou tous les canards).
Chaque semaine, le poids de chaque volaille évolue et on analyse ce poids pour sélectionner celles qui sont à abattre. Celles-ci sont alors retirées de la liste des volailles, vendues, et sont remplacées par de nouvelles volailles.

• Ecrire les classes Volailles, Poulet, Canard.

Volailles
Attributs (possibilité d'en rajouter si besoin)

- protected static double cout; Coût hebdomadaire d'une volaille (on supposera que les coûts – nourriture, entretien,…- sont identiques pour tous les types de volailles).
- protected int id; Identifiant de la volaille.
- protected double poids; Poids de la volaille.

Méthodes (possibilité d'en rajouter si besoin)

- accesseurs/mutateurs (selon besoins)
- public abstract void evolutionHebdomadaire(); L'évolution dépendra de chaque type de volaille.
- public abstract double prixAchat(); Le prix d'achat dépend du type de volaille.
- public abstract double prixVente(); Le calcul du prix de chaque volaille est différent (car il dépend notamment du type de volaille, mais pas seulement).
- public abstract boolean aAbattre(); Les conditions pour être abattues peuvent être différentes selon les types de volailles.

Poulet
Attributs (possibilité d'en rajouter si besoin)
- private static double prixAuKilo; Prix au kilo des poulets lors de la vente.
- private static double poidsAbattage; Poids au-delà duquel est abattu un poulet.

Méthodes (possibilité d'en rajouter si besoin)
- accesseurs/mutateurs (selon besoins)
- public void evolutionHebdomadaire(); Chaque semaine un poulet prend entre 200g et 500g.
- public boolean aAbattre(); true si poids ≥ poidsAbattage, false sinon.
- public double prixAchat(); prix d'achat d'un poulet.
- public double prixVente(); si aAbattre(), prix = poids x prixAukilo – coutHebdomadaire , sinon prix = – coutHebdomadaire.

Canard
Attributs (possibilité d'en rajouter si besoin)
- private static double prixAuKilo; Prix au kilo des canards lors de la vente.
- private static double poidsAbattage; Poids au-delà duquel est abattu un canard.

Méthodes (possibilité d'en rajouter si besoin)
- accesseurs/mutateurs (selon besoins)
- public void evolutionHebdomadaire(); Chaque semaine un canard prend entre 200g et 550g.
- public boolean aAbattre(); true si poids ≥ poidsAbattage, false sinon.
- public double prixAchat(); prix d'achat d'un canard.
- public double prixVente(); si aAbattre(), prix = poids x prixAukilo – coutHebdomadaire , sinon prix = – coutHebdomadaire.

• Nous allons créer une classe représentant tout un élevage.
Classe **Elevage**
Attributs (possibilité d'en rajouter si besoin)
- private ArrayList<Volaille> liste; liste des volailles de l'élevage

Méthodes (possibilité d'en rajouter si besoin)
- accesseurs/mutateurs (selon besoins)
- public void ajouterVolaille(Volaille v); Ajoute une volaille à la liste.
- public ArrayList<Volaille> abattre(); sélectionne toutes les volailles à abattre, les supprime de la liste de l'élevage et les retourne dans une liste.
- public double prixRevient(); Calcul le prix de toutes les volailles (prix de vente – coût hebdomadaire).

• Dans le programme principal, nous allons simuler une année d'exploitation (52 semaines). Pour cela nous allons réaliser les actions suivantes
- Créer un élevage
- Fixer les coûts fixes (coût hebdomadaire d'une volaille, prix au kilo de chaque type de volaille, poids d'abattage de chaque type de volaille,…)
- Ajouter à votre élevage un nombre N (au choix) de volailles (aléatoirement entre les différents types de volailles)
- Pour chaque semaine
    - Faire évoluer toutes les volailles
    - Calculer le prix de revient pour la semaine en cours (et faire la somme de chaque semaine pour avoir un prix de revient sur l'année)
    - Supprimer toutes les volailles à abattre de l'élevage
    - Acheter des volailles (pour en avoir toujours N).
    - Afficher le prix de revient de la semaine, le prix de revient de l'année et chaque volaille (exemple: poulet *id* (poids/poidsAbattage)).

Un diagramme de classes et un exemple de données sont présentés ci-dessous.

• Si tout est bien codé, faire évoluer l'élevage ne devrait pas poser de problème. Ainsi, l'éleveur décide d'avoir aussi des poules. Les poules sont achetées déjà adultes et sont gardées tant qu'elles pondent suffisamment. Ajoutez une classe **Poule**.
Attributs (possibilité d'en rajouter si besoin)

- private static double prixAuKilo; Prix au kilo des poules lors de la vente.
- private static double prixOeuf; Prix de vente d'un œuf.
- private static int ponte; Nombre d'œufs pondus dans la semaine

Méthodes (possibilité d'en rajouter si besoin)

- accesseurs/mutateurs (selon besoins)
- public void evolutionHebdomadaire(); Chaque semaine une poule pond entre 0 et 8 oeufs.
- public boolean aAbattre(); true si une poule pond moins de 3 œufs pendant 2 semaines.
- public double prixAchat(); prix d'achat d'une poule.
- public double prixVente(); si aAbattre(), prix = poids x prixAukilo + nbre d'œufs x prix oeuf – coutHebdomadaire , sinon prix = nbre d'œufs x prix oeuf – coutHebdomadaire.

Exemple de données

	Poulet	Canard	Poule
Prix achat	1.5	1.5	5
Poids à l'achat (kg)	0.25	0.25	1.2
Prix vente au kilo	3.9	7.5	0.6
Poids d'abattage	1.5	2.5	
Prix vente d'un œuf			0.25
Coût hebdomadaire	1.0	1.0	1.0

Diagramme de classes