Fondamentaux du développeur

TP - Java & Programmation Orientée Objet

Philippe Roussille

3iL INGENIEURS
</les sciences informatiques>

1 Objectifs

1. Comprendre la structure d'un programme Java.
2. Savoir créer et manipuler des classes et objets.
3. Découvrir l'encapsulation, les constructeurs et les méthodes.
4. Mettre en œuvre l'héritage et les méthodes redéfinies (`@Override`).
5. Introduire la notion de membres `static`.

2 Partie 1 - Vos premiers canards

2.1 Exercice 1 - Mon premier canard

Créez une classe `Canard`. Elle doit contenir :

— un attribut `couleur` (texte)
— un attribut `taille` (nombre entier, en cm)

Dans la méthode `main`, créez un canard jaune de 10 cm et affichez ses valeurs avec `System.out.println`.

Question : Que se passe-t-il si vous oubliez d'initialiser un attribut ?

2.2 Exercice 2 - Encapsulation et constructeur

Modifiez la classe `Canard` :

— Les attributs deviennent `private`.
— Ajoutez un **constructeur** permettant de définir couleur et taille dès la création.
— Ajoutez des **getters** et **setters**.

Dans le `main`, créez deux canards différents et affichez leurs caractéristiques.

Question : Pourquoi est-il préférable de rendre les attributs privés ?

2.3 Exercice 3 - Méthodes d'instance

Ajoutez à votre classe `Canard` :

- Une méthode `cancaner()` qui affiche `"Coin coin !"`.
- Une méthode `grandir()` qui augmente la taille du canard de 1 cm.

Créez plusieurs canards et faites-les grandir, puis afficher leur taille finale.

Question : Que se passerait-il si `taille` était `static` ?

3 Partie 2 - Les canards à accessoires

3.1 Exercice 4 - Le chapeau du canard

Créez une classe `Accessoire` avec :

- un attribut `nom`,
- un attribut `couleur`,
- un constructeur et un `toString()` propre.

Puis ajoutez à `Canard` un attribut `accessoire` (type `Accessoire`), et une méthode `afficherDescription()` qui affiche par exemple :

```
Canard jaune (12 cm) avec un chapeau rouge.
```

Créez plusieurs canards avec différents accessoires.

3.2 Exercice 5 - Le canard chef

Créez une classe `CanardChef` qui **hérite** de `Canard`.

- Ajoutez un attribut `nombreSubalternes`.
- Redéfinissez la méthode `afficherDescription()` pour inclure cette information.

Testez en créant un `CanardChef` avec un accessoire et deux subalternes.

Question : Quelle méthode sera appelée si vous écrivez `chef.afficherDescription()` ? Pourquoi ?

4 Partie 3 - Des canards au monde réel

4.1 Exercice 6 - Comptage de canards

Ajoutez à la classe `Canard` un attribut **statique** `compteur` qui compte le nombre total de canards créés. Incrémentez-le dans le constructeur.

Dans `main`, affichez le nombre total de canards après en avoir créé plusieurs.

Question : Quelle différence entre un attribut `static` et un attribut d'instance ?

4.2 Exercice 7 - L'étang

Créez une classe `Etang` contenant :

— Une liste de canards (`ArrayList<Canard>`).
— Une méthode `ajouterCanard(Canard c)`.
— Une méthode `cancanerTous()` qui fait cancanner chaque canard.

Dans `main`, créez un étang, ajoutez quelques canards, puis faites-les tous cancaner.

Question : Que se passerait-il si vous remplaciez `ArrayList` par un simple tableau ?

5 Partie 4 - Passage à des objets du quotidien

5.1 Exercice 8 - Changement de décor : la bibliothèque

Créez une classe `Livre` avec :

— `titre`, `auteur`, `annee`
— une méthode `afficher()` qui affiche proprement ces informations.

Créez une classe `Bibliotheque` qui contient une `ArrayList<Livre>` et une méthode `afficherTous()`.

Dans `main`, ajoutez quelques livres et affichez la collection.

5.2 Exercice 9 - Héritage et spécialisation

Créez deux sous-classes :

— `Roman` (ajoutez un champ `genre`)
— `Manuel` (ajoutez un champ `matiere`)

Redéfinissez la méthode `afficher()` dans chaque sous-classe pour afficher les détails spécifiques.

Testez avec quelques instances.

Question : Quelle méthode est appelée si vous manipulez un `Roman` stocké dans une `ArrayList<Livre>` ?

6 Pour aller plus loin (facultatif)

1. Ajoutez une méthode `equals()` pour comparer deux canards (même couleur et taille).
2. Faites un `toString()` pour toutes vos classes.