PLANCHE D'EXERCICES N°2
- PROGRAMMATION ORIENTÉE OBJET EN JAVA -

Une proposition de correction des Tps est disponible sur github via ce lien :
https://github.com/m-assili/dev-isimg/tree/master/2020_2021/POO_Java

Exercice 1: Notion de classe

Soit une personne caractérisée par son lastName et firstName de type chaine de caractères. On doit être en mesure d'attribuer un lastName et un firstName pour une personne en s'assurant que ces informations contiennent bien des lettres seulement.

On pourra aussi afficher une personne comme dans l'exemple qui suit "je suis Nicola Tesla".

1. Ecrire la classe Personne

2. Ecrire une classe Principale qui créé une personne, lui affecte ses caractéristiques et puis l'afficher.

Exercice 2: classe, constructeur, toString, equals, getters, setters

Soit la classe Livre suivante:

```java
public class Livre {
	// Attributs
	private String titre, auteur;
	private int nbPages;

	// Constructeur
	public Livre(String unAuteur, String unTitre) {
		auteur = unAuteur;
		titre = unTitre;
	}

	// Accesseur
	public String getAuteur() {
		return auteur;
	}

	// Modificateur
	public void setNbPages(int n) {
		nbPages = n;
	}
}
```

1. Créer une deuxième classe **TestLivres** qui contiendra la méthode main.
2. Créer les deux livres suivants

   livre1: « Thinking in Java » de Bruce Eckel

   livre2: « Programmer en Java » de Claude Delannoy
3. Afficher les auteurs de ces deux livres.
4. Ajouter les **accesseurs** (getters) pour les attributs *titre* et *nbPages*
5. Ajouter les **modificateurs (setters)** pour les attributs *auteur* et *titre*
6. Changer le modificateur de *nbPages* : sa valeur doit être strictement positive.
7. Dans la méthode main

   a. Affecter à chaque livre son nombre de pages.

   b. Afficher les deux livres

   c. Calculez le nombre total de pages de ces 2 livres et affichez-le.
8. Dans la classe Livre, ajoutez une méthode *afficher()* qui affiche une description du livre (auteur, titre et nombre de pages). Utilisez *afficher()* dans la méthode main de TestLivres.
9. Ajouter l'instruction System.out.println(livre1) où livre désigne un des livres que vous avez créés.
10. Ajouter une méthode *toString()* qui retourne une chaîne de caractères qui décrit le livre. Exécutez à nouveau la classe TestLivres. Remarquez ce qui est affiché maintenant par l'instruction System.out.println(livre)
11. Modifier la méthode *afficher()* pour utiliser *toString()*
12. Enlever (commenter provisoirement) le seul constructeur de la classe Livre. Sans ajouter de nouveau constructeur, peut-on quand même créer un nouveau livre dans la méthode main ?
13. Remettez le constructeur que vous avez enlevé. Est-ce que le code de la méthode main de la question précédente fonctionne toujours (tester) ?
14. Ajoutez 2 constructeurs pour avoir 3 constructeurs dans la classe :
    - un constructeur qui n'a pas de paramètre,
    - un qui prend en paramètre l'auteur et le titre du livre,
    - et l'autre qui prend en plus le nombre de pages.
15. Utilisez les 3 constructeurs (et éventuellement d'autres méthodes) pour créer 3 nouveaux livres de votre choix dans la méthode main de TestLivres.
16. Dans la méthode main, créer un nouveau livre identique (même valeurs d'attributs) à un livre déjà créé, puis tester l'égalité des deux livres avec l'opérateur ==. Que remarquez-vous ? Que-faut-il faire pour tester l'égalité de deux objets ?

17. Ajouter une méthode *equals()* qui compare deux livres.
18. Ajouter un attribut **nbLivres** désignant le nombre de livres créés. Quelle est la spécificité de cet attribut ? Où doit-on initialiser la valeur de cet attribut ?
19. Modifier les constructeurs afin de mettre à jour la valeur de l'attribut **nbLivres**
20. Ajouter une méthode *getNbLivres()* permettant de renvoyer le nombre de livres crées. Quel est le type de cette méthode ?
21. Dans la méthode main, afficher le nombre de livres créés après chaque instanciation d'un livre.
22. Mettez la classe Livre dans un package que vous nommez librairie
23. Résoudre les éventuels problèmes d'importation de la classe Livre dans la classe TestLivres.

Exercice 3: encore des classes...

On désire modéliser un Robot simple. Un Robot est caractérisé par son nom, sa position (x et y) sur un plan, et une direction (NORD, EST, SUD, OUEST).

On pourra créer des robots de plusieurs façons:

- en fournissant leur nom, leur position et leur direction.
- des robots portant le nom "anonyme", dans ce cas on ne sait ni leur position ni leur direction.
- Des robots ayant un nom, à l'origine du repère et leur direction est NORD.

Un robot peut :

- avancé d'un pas dans sa direction.
- tourner à droite.

On pourra aussi afficher les informations sur ce robot à tout moment.

1. Ecrire la classe RobotSimple
2. Ecrire une classe TestRobot qui teste la classe RobotSimple.

Exercice 4 (héritage, association entre classes)

On souhaite effectuer la gestion des factures d'un magasin pour cela on vous demande d'écrire 3 classes.

Dans ce magasin, un article est caractérisé par :

- un code (nombre entier),
- une désignation (chaine de caractères),

- un prix (nombre réel),

1. Ecrire la classe « Article »
    - Ecrire les attributs
    - Ecrire le constructeur sans paramètres et un constructeur d'initialisation des attributs.
    - Ecrire la méthode getPrix() pour retourner le prix de l'article
    - Ecrire la méthode setPrix() pour changer le prix de l'article
    - Ecrire la méthode toString() qui renvoie les caractéristiques d'un article séparées par un point-virgule.
    - Ecrire la méthode statique equals() qui teste l'égalité de deux articles passés en paramètres. Deux articles sont égaux s'ils ont les mêmes valeurs d'attributs.

Un article en solde contient une information additionnelle :

- Remise : nombre réel qui représente le pourcentage de réduction sur le prix d'origine.

2. Ecrire la classe ArticleSolde
    - Ecrire les attributs.
    - Ecrire les constructeurs par défaut et d'initialisation.
    - Redéfinir la méthode getPrix() afin de tenir en compte du solde.

Le magasin établit des factures numérotées automatiquement (en partant de 1) et datées.

Une facture est caractérisée par :

- Un numéro de facture (automatiquement incrémenté)
- Date de facture : prend par défaut la date du système

    Date date = new Date(); // Il faut importer la classe java.util.Date

- un ensemble de LigneFacture (maximum 5000 lignes).

Une LigneFacture contient l'article acheté et la quantité achetée. On pourra créer une ligne de facture en fournissant un article et une quantité.

3. Ecrire la classe LigneFacture
4. Ecrire la classe Facture
    - Ecrire les attributs.
    - Ecrire le constructeur Facture()

- Ecrire la méthode void ajouterAchat(LigneFacture a) qui permet d'ajouter une ligne dans la facture

- Ecrire la méthode double montant() qui retourne le montant total de la facture.

5. Compléter la classe Main suivante :

```
public class Main {

public static void main(String[] args) {

        Article a = new Article(10, "PC Lenovo", 2000, "informatique");

        Article b = new ArticleSolde(20, "Bureau", 1000, "bureautique", 10);

        Article c = new Article(30, "Imprimante Rx", 500, "informatique");

        Article d = new Article(40, "Rame papier", 10, "bureautique");
 /* Effectuer les achats suivants : 2 PC, 2 bureaux, 1 imprimante et 5 rames de papier. */

                // Créer une facture et ajouter les achats



                // Afficher le montant total de la facture */

        }

 }
```