Séries de TDs/TPs N° : (3 & 4)

-prépare par : **BENHAMMOU MOHAMED (IDAI)**

-encadré par: **PR. El Mokhtar EN-NAIMI**

Exercice 1:

Livre.java:

Objectif :

L'objectif principal de cet exercice est d'apprendre et de mettre en pratique les concepts fondamentaux de la programmation orientée objet (POO) dans un langage comme Java . Ces concepts incluent l'encapsulation, les constructeurs, les getters et setters, ainsi que l'utilisation de méthodes pour gérer les données et les afficher.

Étapes réalisées :

1. **Définition de la classe Livre :**
   - La classe représente un modèle pour un livre, avec trois attributs essentiels :
     - **Titre** : Le nom du livre.
     - **Auteur** : Le nom complet de l'auteur.
     - **Prix** : Le coût du livre.
   - Ces attributs encapsulent les données pour éviter tout accès direct depuis l'extérieur.
2. **Méthodes d'accès (Getters et Setters) :**
   - Les getters permettent de lire les valeurs des attributs tout en respectant l'encapsulation.
   - Les setters permettent de modifier les valeurs des attributs en offrant un contrôle supplémentaire (par exemple, vérifier la validité des données avant de les modifier).
3. **Constructeur surchargé :**
   - Ce constructeur permet d'initialiser un objet Livre directement avec des valeurs spécifiques (fournies par l'utilisateur). Cela améliore l'efficacité et la lisibilité du code en évitant de devoir appeler plusieurs fois des setters après la création d'un objet.
4. **Méthode afficher() :**
   - Cette méthode encapsule la logique d'affichage et garantit que toutes les informations sont présentées de manière uniforme. C'est une bonne pratique car elle centralise la responsabilité d'affichage dans une seule méthode.
5. **Programme principal (Main) :**
   - Permet de tester la classe en simulant une interaction utilisateur :
     - L'utilisateur saisit les informations d'un livre.
     - Un objet Livre est créé et initialisé avec ces données.
     - Les informations du livre sont affichées en sortie.

```
tp3et 4_d76059ac\bin' 'Livre'
Entrez le titre du livre:PROGRAMATION ORIENTEE OBJET
Entrez l'auteur du livre: Johan Dahl et Kristen Nygaard
Entrez le prix du livre: 1500

Les informations du livre sont :
Titre : PROGRAMATION ORIENTEE OBJET
Auteur : Johan Dahl et Kristen Nygaard
Prix : 1500.0 DH
PS C:\Users\pc\Desktop\java tp3et 4>
```

Debug: hell...
Debug: Livre

Java: Ready Ln 43, Col 1 Spaces: 4 UTF-8 CRLF Java Go Live CODEGPT

Exercice 2:

Employe.java:

Objectif :

Cet exercice a pour but de créer une classe Employe en Java permettant de gérer les informations d'un employé, d'effectuer des calculs (âge, ancienneté) et de manipuler son salaire en fonction de son ancienneté. Il inclut également un programme de test pour vérifier les fonctionnalités implémentées.

Étapes réalisées :

1. **Définition des attributs :**
   La classe Employe contient les attributs suivants :
   - matricule : identifiant unique de l'employé.
   - nom : nom de l'employé.
   - prenom : prénom de l'employé.
   - anneeNaissance : année de naissance de l'employé.
   - anneeEmbauche : année où l'employé a été embauché.
   - salaire : salaire actuel de l'employé.
2. **Encapsulation :**
   - Les attributs sont déclarés private pour garantir l'encapsulation.
   - Des **getters** et **setters** ont été définis pour permettre l'accès et la modification des attributs tout en respectant les principes de programmation orientée objet.
3. **Constructeur :**
   - Un constructeur surchargé permet d'initialiser les valeurs des attributs lors de la création d'un objet.
   - Ce constructeur est testé en saisissant les données depuis l'utilisateur.
4. **Méthodes ajoutées :**
   - **getAnciennete()** : calcule le nombre d'années écoulées depuis l'embauche de l'employé.
   - **getAge()** : calcule l'âge actuel de l'employé en utilisant l'année de naissance.
   - **AugmentationDuSalaire()**: modifie le salaire de l'employé selon son ancienneté :
     - <5 ans : augmentation de 2%.
     - <10 ans : augmentation de 5%.
     - ≥10 ans : augmentation de 10%.
   - **AfficherEmploye()** : affiche les informations complètes de l'employé, y compris son âge, son ancienneté et son salaire.

```
 le prénom de l'EMPLOYE : MOHAMED
l'année de naissance de l'EMPLOYE : 2003
 l'année d'embauche de l'EMPLOYE : 2023
le salaire de l'EMPLOYE :7000

Informations de l'EMPLOYE avant augmentation :
Matricule : 1002
Nom : BENHAMMOU
Prénom : MOHAMED
Année de naissance : 2003
```

```
Informations de l'EMPLOYE après augmentation de salaire :
Matricule : 1002
Nom : BENHAMMOU
Prénom : MOHAMED
Année de naissance : 2003
Année d'embauche : 2023
Salaire : 7140.0 DH
Âge : 21 ans
Ancienneté : 1 ans
PS C:\Users\pc\Desktop\java tp3et 4>
```

Debug: hell...
Debug: Livre
Debug: Em...

Exercice 3:

Q1 : Définition des classes Document, Livre et Dictionnaire

1. **Class Document :**
   - Représente une entité générique avec deux attributs principaux : un numéro unique et un titre.
   - Le constructeur permet d'initialiser ces deux attributs lors de l'instanciation.
   - La méthode toString retourne une description textuelle du document.
2. **Class Livre :**
   - Hérite de la classe **Document** et ajoute deux attributs supplémentaires : un auteur et un nombre de pages.
   - La méthode toString est redéfinie pour inclure les informations propres à un livre.
3. **Class Dictionnaire :**
   - Hérite de la classe **Document** et ajoute deux attributs : la langue et le nombre de tomes.
   - La classe inclut un constructeur par défaut pour plus de flexibilité lors de l'initialisation.
   - La méthode **toString** est redéfinie pour retourner une description du dictionnaire.

Q2 : Classe Bibliotheque pour tester les entités

La classe Bibliotheque contient une méthode principale main, qui permet de tester l'implémentation des classes :

- Création de deux livres («Les Misérables» et «Le Petit Prince»).
- Création d'un dictionnaire («Dictionnaire Larousse»).
- Affichage des descriptions textuelles des objets créés.

Q3 : Classe ListeDeDocuments pour gérer les documents

La classe ListeDeDocuments fournit un conteneur pour stocker et manipuler plusieurs documents. Voici les fonctionnalités clés :

1. **Ajout de documents :**
   - La méthode ajouterDocument ajoute un objet de type Document à une liste interne.
2. **Affichage des auteurs :**
   - La méthode tousLesAuteurs parcourt la liste et affiche les auteurs pour chaque document de type Livre.
   - Les autres documents affichent un message indiquant qu'ils n'ont pas d'auteur.
3. **Affichage des descriptions :**
   - La méthode tousLesDocuments appelle la méthode toString de chaque document pour afficher sa description.

Q4 : Afficher tous les auteurs

La méthode tousLesAuteurs est testée dans la classe Bibliotheque. Elle permet de vérifier que :

- Les auteurs des livres sont correctement affichés.
- Les dictionnaires ou autres types de documents ne contenant pas d'auteur affichent un message adapté.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Liste des auteurs:
Numéro: 1, Auteur: Victor Hugo
Numéro: 2, Auteur: Antoine de Saint-Exupéry
```

Q5 : Afficher tous les documents

La méthode tousLesDocuments est également testée. Elle parcourt la liste des documents et utilise leur méthode toString respective pour afficher leurs descriptions complètes.

```
Liste des documents:
Livre[Numéro=1,Titre="Les Misérables", Auteur="Victor Hugo",Pages=1232]
Livre[Numéro=2,Titre="Le Petit Prince", Auteur="Antoine de Saint-Exupéry",Pages=96]
Dictionnaire[Numéro=3,Titre="Dictionnaire Larousse",Langue="Français", Tomes=2]
PS C:\Users\pc\Desktop\java tp3et 4>
```

Q6 : Tests finaux dans la classe Bibliotheque

Tous les tests sont réunis dans la méthode main, assurant que :

- Les entités sont correctement initialisées.
- Les documents peuvent être ajoutés à une liste et manipulés collectivement.
- Les méthodes d'affichage fonctionnent comme prévu.

Exercice 4:

Objectif : L'objectif de cet exercice est de créer une hiérarchie de classes en Java pour modéliser des formes géométriques telles que les carrés et les rectangles en utilisant la programmation orientée objet. Le programme doit permettre géredes instances de ces formes, de les stocker dans une liste statique et d'afficher leurs propriétés.

Classe Figure:

La classe Figure est une classe abstraite qui sert de base pour les autres formes géométriques. Elle contient trois attributs principaux : l'**abscisse**, l'**ordonnee**, et la **couleur** de la forme. La classe contient également un constructeur pour initialiser ces attributs, ainsi que des méthodes pour récupérer leurs valeurs. Une liste statique **instances** est utilisée pour stocker toutes les instances des objets de type Figure et ses sous-classes. La méthode getInstances() retourne une copie de cette liste afin d'éviter toute modification non désirée. La méthode toString() est redéfinie pour fournir une représentation lisible de l'objet.

Classe Carre

La classe Carre hérite de Figure et représente un carré. Elle ajoute un attribut supplémentaire, **cote**, qui représente la longueur du côté du carré. Le constructeur initialise cet attribut en plus des attributs hérités de la classe Figure. La méthode toString() est redéfinie pour afficher les informations spécifiques à un carré, y compris sa taille.

Classe Rectangle

La classe Rectangle, également une sous-classe de Figure, représente un rectangle. Elle possède deux nouveaux attributs : **longueur** et **largeur**. Le constructeur initialise ces attributs ainsi que ceux de la classe parent. De même, la méthode toString() est redéfinie pour fournir une représentation complète du rectangle.

Classe principale TestFigures

La classe TestFigures contient la méthode main() qui est utilisée pour tester le bon fonctionnement des classes Figure, Carre et Rectangle. Des instances de Carre et Rectangle sont créées avec des valeurs spécifiques, puis affichées. Le programme affiche les instances spécifiques de chaque sous-classe et une liste contenant toutes les instances de Figure.

3. Exécution du programme:

Lors de l'exécution du programme, l'utilisateur verra un affichage des différentes instances créées pour les carrés et les rectangles. Pour chaque forme géométrique, le programme affichera les valeurs de ses attributs, comme l'abscisse, l'ordonnée, la couleur, la longueur du côté (pour les carrés) et les dimensions (pour les rectangles).

```
Instances de Carre:
Carre[abscisse=5, ordonnee=5, couleur=300, cote=5]

Instances de Rectangle:
Rectangle[abscisse=5, ordonnee=6, couleur=300, longueur=30, largeur=40]

Toutes les instances de Figure:
Rectangle[abscisse=5, ordonnee=6, couleur=300, longueur=30, largeur=40]
Carre[abscisse=5, ordonnee=5, couleur=300, cote=5]
PS C:\Users\pc\Desktop\java tp3et 4>
```