Université AbdelMalek Essaâdi
Faculté des Sciences et Techniques
Département Génie Informatique
Tanger

Année Universitaire : 2023/2024
POO en JAVA
(Filière: Licence, LDAI *(***S5**))
Prof. E. M. EN-NAIMI
Intervenant : Dr. NDAMA Oussama

Séries de TDs/TPs N° : (3 & 4)

Exercice 1:

1. Définir une classe **Livre** avec les attributs suivants : Titre, Auteur (Nom complet), Prix ?
2. Définir les méthodes d'accès aux différents attributs de la classe (getters et setters) ?
3. Définir un **constructeur surchargé** permettant d'initialiser les attributs par des valeurs saisies par l'utilisateur ?
4. Définir la méthode **Afficher()** permettant d'afficher les informations du livre en cours ?
5. Écrire **un programme testant la classe Livre ?**

Exercice 2:

1. Définir une classe Employé caractérisée par les attributs : **Matricule**, **Nom**, **Prénom, AnneeNaissance**, **AnneeEmbauche**, **Salaire** ?
2. Définir les méthodes d'accès aux différents attributs de la classe (getters et setters)?
3. Définir un constructeur surchargé permettant d'initialiser les attributs par des valeurs saisies par l'utilisateur ?
4. Ajouter à la classe la méthode **getAnciennete( )** qui retourne le nombre d'années d'ancienneté de l'employé ?
5. Ajouter à la classe la méthode **getAge( )** qui retourne l'âge de l'employé ?
6. Ajouter à la classe la méthode **AugmentationDuSalaire( )** qui augmente le salaire de l'employé en prenant en considération l'ancienneté :

   - Si Ancienneté < 5 ans, alors on ajoute 2%.

   - Si Ancienneté < 10 ans, alors on ajoute 5%.

   - Sinon (c'est-à-dire : Ancienneté >= 10 ans), on ajoute 10%.

7. Ajouter la méthode **AfficherEmployé()** qui affiche les informations de l'employé ?
8. Ecrire **un programme de test pour la classe Employé** ?

Exercice 3 :

On modélise une application devant servir à l'inventaire d'une bibliothèque. Elle devra traiter des documents de nature diverse : des livres, des dictionnaires, et autres types de documents qu'on ne connait pas encore précisément mais qu'il faudra certainement ajouter un jour (articles, bandes dessinées...). Tous les documents possèdent un numéro d'enregistrement et un titre. A chaque livre est associe, en plus, un auteur et un nombre de pages, les dictionnaires ont, eux, pour attributs supplémentaires une langue et un nombre de tomes.

On veut manipuler tous les articles de la bibliothèque au travers de la même représentation : celle d'un document.

1. Définissez les classes **Document, Livre** et **Dictionnaire**. Définissez pour chacune un constructeur permettant d'initialiser toutes ses variables d'instances.
2. Définissez une classe **Bibliothèque** réduite à une méthode **main** permettant de tester les classes précédentes (ainsi que les suivantes).
3. Définissez la classe **ListeDeDocuments** permettant de créer une liste vide de documents, puis y adjoindre une fonction permettant d'ajouter un document.
4. Dans la classe **ListeDeDocuments** définissez une méthode **tousLesAuteurs()** qui affiche la liste des numéros des documents de la liste avec, pour chacun, l'éventuel auteur.
5. Redéfinnisez la méthode **toString()** dans la classe **Document** ainsi que dans les classes **Livre** et **Dictionnaire** et qui renvoie une chaine de caractères d´écrivant un document, un livre ou un dictionnaire. Ajoutez alors dans la classe **ListeDeDocuments** une méthode **tousLesDocuments()** qui affiche consécutivement la description de tous les documents.
6. Proposez quelques lignes de codes à ajouter à la classe **Bibliothèque** afin de tester la classe **ListeDeDocuments.**

Exercice 4 :

Ecrire trois classes **Figure, Carre**, et **Rectangle**, telles que :

1. **Figure** a des attributs **abscisse** et **ordonnée**, ainsi qu'une **couleur** (encodée par un entier).
2. **Carre** et **Rectangle** héritent de **Figure**, mais ont en plus une ou deux longueur(s) pour les côtes.
3. **Figure** a un attribut privé **Vector** référençant toutes les instances de sa classe et de ses sous-classes.
4. **Figure** a une méthode statique **getInstances()** renvoyant ce vecteur.
5. **Carre** et **Rectangle** redéfinissent cette méthode **getInstances()** de manière à ne récupérer que les instances qui correspondent à leur type.