Année Universitaire: 2022-2023

TD N°2

Matière : Programmation Orientée Objet
Niveau : 2IOT
Enseignante : Mme Jihen KHALFALLAH

Exercice 1 :

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 connaît pas encore précisément mais qu'il faudra certainement ajouter un jour (bandes dessinées, dictionnaires bilingues,...).

On définira une classe **Bibliothèque** réduite à la seule méthode main pour tester les différentes classes de ce TD.

Tous les documents possèdent un titre. Quand un document est créé, son titre est donné à la création et par la suite, il ne change plus.

1. Définir la classe **Document** avec son constructeur public, et la propriété *titre* privée et son accesseur.
2. On veut attribuer un *numéro d'enregistrement* unique dès que l'on crée un objet **Document**.
3. On veut que le premier document créé ait le numéro 0, et que ce numéro s'incrémente de 1 à chaque création de document. Quelles propriétés faut-il ajouter à la classe Document ? Les-quelles doivent être static ? Les ajouter sans leurs accesseurs, et modifier le constructeur. Puis ajouter une méthode *getNumero* renvoyant le numéro d'enregistrement du document.
4. Définir la méthode *toString* renvoyant la chaîne de caractères constituée du numéro d'enregistrement et du titre du document.

La bibliothèque est appelée à traiter des documents de nature diverse : des livres, des dictionnaires, et autres types de documents.

A chaque livre est associé, 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 de document.

5. Définissez les classes **Livre** et **Dictionnaire** étendant la classe **Document**. Définissez pour chacune un constructeur permettant d'initialiser toutes ses variables d'instances respectives.
6. Redéfinissez la méthode *toString*() dans les classes **Livre** et **Dictionnaire** pour qu'elle renvoie une chaîne de caractères décrivant un livre ou un dictionnaire, en plus de la description normale d'un document.
7. Ecrire une classe exécutable nommée **Bibliothèque** qui teste les classes précédemment définies.

Exercice 2 :

Le but de l'exercice est de créer une hiérarchie de classes pour représenter les étudiants d'une université. Il y a 3 types d'étudiants : en Licence, ceux en Master, et ceux en Doctorat.

Chaque étudiant a un *nom*, une *adresse* et un *numéro*. Les étudiants ont un profil. Pour les étudiants en Licence on parle de parcours. Les étudiants en Master une spécialité et les étudiants en Doctorat un directeur de recherche.

1. Créer la classe **Etudiant** qui contient les trois attributs privés:
   - ✓ Nom : de type String
   - ✓ Adresse : de type String
   - ✓ Numero : de type int

   Et définir les méthodes suivantes (publiques):
   - Le constructeur de la classe de signature Etudiant (String nom, String adresse, int numero) en initialisant les champs.
   - Les méthodes accesseur *get*() et *set*()
   - Une méthode *afficher*() de signature void afficher() au format décrit ci-après :
   - ✓ Nom : CHACHIA MELEK
   - ✓ Adresse : SOUSSE
   - ✓ Numéro : 2
2. Définir les classes nécessaires à cette hiérarchie de classes, en leurs ajoutant les méthodes (constructeurs, get()et set() etc...).

❖ Surcharge de la méthode afficher() :

3. Dans la classe Etudiant, créer une méthode de signature *void afficher*(boolean compact) qui affiche :
   a. De la même manière que la méthode *void afficher*() précédemment défini(si compact = false)
   b. Si compact= true, l'affichage doit être de la manière suivante [CHACHIA MELEK, SOUSSE ,2]

❖ Redéfinition de la méthode afficher() :

4. Dans la classe **EtudiantLicence**, créer la méthode *void afficher*() qui appelle la méthode *afficher*() de **Etudiant** (au moyen de `super`) puis affiche son parcours.
5. Idem pour **EtudiantMaster**, idem pour **EtudiantDoctorat.**

❖ Les classes abstraites :

Dans cette partie, nous supposons que la classe **Etudiant** fasse partie d'une bibliothèque, nous allons concevoir des méthodes que les classes qui héritent de la classe doivent utiliser.

6. Rendre la classe **Etudiant** abstraite (en ajoutant le mot `abstract` devant class)
7. Tester l'affichage dans une classe **Main** qui construit un ensemble d'étudiants de différents profils et affiche une liste d'étudiants avec les informations les concernant.
8. Déclarer une méthode abstraite, ne retournant rien (c'est-à-dire void), sans arguments, nommée *afficherProfil()*.La méthode *afficherProfil()* dans la classe **EtudiantLicence** (resp. dans les classes **EtudiantMaster** et **EtudiantDoctorat**) n'effectue qu'une chose : afficher la chaîne " Etudiants en Licence " (resp. les chaînes " Etudiants en Master ", " Etudiants en Doctorat ").
9. Tester l'affichage dans la classe **Main**.