**Exercice 1 : Livre**

1. Définir une classe **Livre** avec les attributs suivants : Id, Titre, Auteur (Nom complet), Prix.
2. Définir les accesseurs aux différents attributs de la classe.
3. Définir un **constructeur** permettant d’initialiser les attributs d'un objet livre par des valeurs saisies par l’utilisateur. Sachant que Id doit être auto-incrément.
4. Définir la méthode **toString ( )**permettant d’afficher les informations du livre en cours.
5. Écrire un programme testant la classe Livre.

**Exemple d’exécution :**

Donner le titre du livre n° 1 :Langage C  
Donner l'auteur du livre n° 1 :IDMANSOUR  
Donner le prix du livre n° 1 :200  
Donner le titre du livre n° 2 :Langage JAVA  
Donner l'auteur du livre n° 2 :LACHGAR  
Donner le prix du livre n° 2 :300  
Donner le titre du livre n° 3 :Langage C#  
Donner l'auteur du livre n° 3 :CHAOULID  
Donner le prix du livre n° 3 :250  
Le prix du livre Langage C de l'auteur IDMANSOUR est :200 DH  
Le prix du livre Langage JAVA de l'auteur LACHGAR est :300 DH  
Le prix du livre Langage C# de l'auteur CHAOULID est :250 DH  
Le nombre de livres est 3

**Exercice 2 : Point**

1. Définir une classe **Point**caractérisée par son abscisse et son ordonné.
2. Définir à l’aide des **getters**et les **setters** les méthodes d’accès aux attributs de la classe.
3. Définir le **constructeur** sans paramètre et d’initialisation de la classe.
4. Définir la méthode **distance ()**qui retourne la distance entre l’origine du repère et le point en cours.
5. Définir la méthode **toString()**permettant d'afficher les cordonnées d'un Point.
6. Écrire un programme permettant de tester la classe.

**Exemple d’exécution :**

La distance entre l’origine et le Point (2,3) est : 3.605551275463989  
La distance entre l’origine et le Point (1,4) est : 4.123105625617661  
La distance entre l’origine et le Point (0,2) est : 2.0

**Note**

* La distance entre deux points (x1,y1) et (x2,y2) est égale à :  
  distance 
* La classe java.lang.Math contient des méthodes static permettant de calculer la racine carré et la puissance d'un certain nombre, voici la déclaration de ces méthodes : Class Math {

public static double sqrt(double nb) :     // retourne la racine carré de nb

public static double pow(double nb, double puissance) : // retourne nbpuissance

…

}

**Exercice 3 : Rectangle**

1. Définir une classe Rectangle ayant les attributs suivants : Longueur et Largeur.
2. Ajouter un constructeur d’initialisation.
3. Définir les accesseurs aux attributs de la classe.
4. Ajouter les méthodes suivantes :
   * perimetre ( ) : retourne le périmètre du rectangle.
   * aire ( ) : retourne l'aire du rectangle.
   * isCarre ( ) : vérifie si le rectangle est un carré.
   * toString ( ) : expose les caractéristiques d’un rectangle comme suit :  
     Longueur : […] - Largeur : […] - Périmètre : […] - Aire : […] - C'est un carré / Ce n'est pas un carré

**Exemple d’exécution :**

-Longueur : 12.6 -Largeur : 5.3 -Perimetre : 35.8 -Aire : 66.78 -Ce n'est pas un carré  
-Longueur : 3.0 -Largeur : 3.0 -Perimetre : 12.0 -Aire : 9.0 -C'est un carré  
-Longueur : 10.5 -Largeur : 5.0 -Perimetre : 31.0 -Aire : 52.5 -Ce n'est pas un carré

**Exercice 4 : Voiture**

1. Définir une classe **Voiture**avec les attributs suivants : Id, Marque, Vitesse, Puissance.
2. Définir un **constructeur** permettant d’initialiser les attributs d'un objet voiture par des valeurs passées en paramètre. Sachant que Id doit être auto-incrément.
3. Définir les accesseurs  aux différents attributs de la classe.
4. Définir la méthode **toString ( )**permettant d’afficher les informations d'une voiture.
5. Écrire un programme testant la classe Voiture.

**Exemple d’exécution :**

id=1, marque=BMW, vitesse=220.0, puissance=8  
id=2, marque=Ford, vitesse=200.0, puissance=6  
id=3, marque=Audi, vitesse=240.0, puissance=8  
Le nombre des voitures est : 3

* Création des classes et objets.
* Manipulation des tableaux d'objets.
* Traduction d'une association entre les classes.

**Exercice 5 : élève**

Un élève sera ici modélisé par la classe Eleve d'un paquetage nommé gestionEleves, de la façon suivante.

La classe Eleve possède trois attributs privés :

* Son nom, nommé nom, de type String,
* Un ensemble de notes, nommé listeNotes, qui sont des entiers rangés dans un tableau int[]
* Une moyenne de type double, nommée moyenne, qui doit toujours être égale à la moyenne des notes contenues dans l'attribut listeNotes. Un élève sans aucune note sera considéré comme ayant une moyenne nulle.

La classe Eleve possède un constructeur permettant uniquement d'initialiser le nom de l'élève.

La classe Eleve possède aussi cinq méthodes publiques :

* Un getter pour la moyenne de l'élève c'est-à-dire une méthode d'en-tête  
      **public double** getMoyenne()  
  renvoie la valeur de l'attribut moyenne ;
* Un getter pour le nom de l'élève c'est-à-dire une méthode d'en-tête  
      **public** String getNom()  
  renvoie le nom de l'élève ;
* Un getter pour la liste des notes de l'élève c'est-à-dire une méthode d'en-tête  
      **public** int[] getListeNotes()  
  renvoie la liste des notes de l'élève ;
* La méthode d'en-tête  
      **public void** ajouterNote(**int** note)  
  ajoute la note reçue en paramètre à listeNotes ; si la note reçue en paramètre est négative, la note introduite est 0 ; si la note reçue en paramètre est supérieure à 20, la note introduite est 20 ; la méthode actualise en conséquence l'attribut moyenne ; l'actualisation est faite à temps constant, et non pas en un temps proportionnel au nombre de notes déjà enregistrées.
* La méthode d'en-tête  
      **public String** toString()  
  qui retourne une description de l'élève considéré (par exemple : "Sophie (12.25)").

*Indication* : si note est une variable de type **int**, l'instruction :  
    listeNotes.add(note);  
ajoute un Integer contenant la valeur note à listeNotes.

Après avoir terminé la classe Eleve, écrire un programme qui teste cette classe.  
*Indication (détail)* : si la méthode toString décrite ci-dessus a été définie dans la classe Eleve, si eleve est de type Eleve, l'instruction :  
    System.out.println(eleve);  
est équivalente à l'instruction :  
    System.out.println(eleve.toString());

**Exercice 6 : Etudiant - Spécialité**

Soit le diagramme de classe suivant :

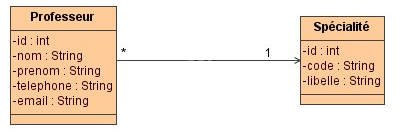


Diagramme de classe

1. Développer la classe **Spécialité** et la classe **Professeur** dans le package « **fr.afpa**».

Chaque classe doit comporter :

* + Un constructeur d’initialisation ;
  + Les accesseurs ;
  + La méthode toString.

**NB :** l’identifiant est auto incrément.

1. Développer une classe de test dans le package « **fr.afpa.test** ».

Dans la classe de test créer :

* **5 Spécialités :**
  + JAVA/JEE
  + .net
  + Gestion de projet
  + CISCO
  + PHP
* **4 Professeurs** : Les deux premiers enseignent JAVA/JEE les deux autres enseignent CISCO.
* Afficher la liste des professeurs par spécialité :

**Exemple :**

**Spécialité JAVA/JEE**

                                               -  SAFI Amal [safi [AT] gmail [DOT] com](mailto:safi@gmail.com)

                                               -  ALAMI Said [alami [AT] yahoo [DOT] fr](mailto:alami@yahoo.fr)

**Spécialité CISCO**

                                               - ALAOUI Reda [alaoui [AT] yahoo [DOT] fr](mailto:alaoui@yahoo.fr)

                                               - KAMALI Imane [i [DOT] kamali [AT] gmail [DOT] com](mailto:i.kamali@gmail.com)

**Note**

Le nom de professeur doit être en majuscule, le prénom doit commencer par une lettre en majuscule et les autres lettres doivent être en minuscule.

**Exemple d’exécution :**

Professeur par spécialité :

Spécialité JAVA/JEE

-SAFI Salim safi [AT] gmail [DOT] com  
-RASHIDI Mohamed rashidi [AT] gmail [DOT] com

Spécialité .net

Aucun professeur dans cette spécialité

Spécialité Gestion de projet

Aucun professeur dans cette spécialité

Spécialité CISCO

-RAMI Amal rami [AT] gmail [DOT] com  
-SIMON Thomas simon [AT] gmail [DOT] com

Spécialité PHP

Aucun professeur dans cette spécialité

**Exercice 7 : Etudiant - Filière**

Soit le digramme de classe suivant :

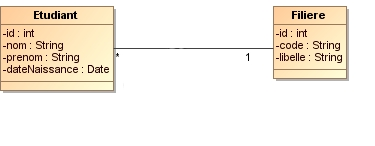


Diagramme de classe

 1. Développer la classe **Étudiant** et la classe **Filière**  dans le package « **fr.afpa**».

    Chaque classe doit comporter :

* Un constructeur d’initialisation ;
* Les accesseurs en cas de besoin ;
* La méthode toString.

**NB :** l’identifiant est auto incrément.

 2. Développer une classe de test dans le package «**fr.afpa.test**».

* Dans la classe cette classe :
* Créer cinq étudiants.
* Afficher les étudiants par filière.

**Exemple d’exécution :**

La liste des filières de notre école :

Filière : Informatique  
Filière : Techno de l'info et de communication  
Filière : Génie civil

Liste des étudiants par filière :

Filière : Informatique

Je suis l'étudiant Rashid Mohamed ma date de naissance est : 1995/01/02  
Je suis l'étudiant Yves Chakib ma date de naissance est : 1992/04/02

Filière : Techno de l'info et de communication

Je suis l'étudiant Alaoui Manal ma date de naissance est : 2011/03/02

Filière : Génie civil

Je suis l'étudiant Safi Meriem ma date de naissance est : 2000/11/02  
Je suis l'étudiant Rami Mouad ma date de naissance est : 2013/01/02