TP La programmation par objet

Exercice 1 — *Les piles*

Dans cette exercice, on vous demande d'implémenter une classe `Pile` en java, pour manipuler des entiers, **en utilisant un tableau**. Il faut ainsi :

(1) Définir un constructeur qui, à partir de la donnée d'un entier `n` représentant le nombre maximum d'éléments que peut contenir la pile, initialise l'instance à l'aide d'un tableau de taille `n`.

(2) Implémenter les méthodes de base d'une telle classe qui :
   (a) teste si la pile est vide ou non,
   (b) teste si la pile est pleine ou non,
   (c) récupère la valeur de l'élément situé sur le haut de la pile,
   (d) empile un élément,
   (e) dépile un élément.

(3) Ecrire une programme de test qui permette à l'utilisateur de créer une pile vide, et de tester ensuite chacune des opérations disponibles.

Exercice 2 — *Les files*

Dans cette exercice, on vous demande d'implémenter une classe `File` en java, pour manipuler des entiers, **en utilisant un tableau**. Il faut ainsi :

(1) Définir un constructeur qui, à partir de la donnée d'un entier `n` représentant le nombre maximum d'éléments que peut contenir la file, initialise l'instance à l'aide d'un tableau de taille `n`.

(2) Implémenter les méthodes de base d'une telle classe qui :
   (a) teste si la file est vide ou non,
   (b) teste si la file est pleine ou non,
   (c) récupère la valeur de l'élément situé en tête de file,
   (d) enfile un élément,
   (e) défile un élément.

(3) Ecrire une programme de test qui permette à l'utilisateur de créer une file vide, et de tester ensuite chacune des opérations disponibles.

Exercice 3 — *La classe `Personne`*

1. Écrire une classe `Personne` contenant comme attributs : ses nom et prénom, son année de naissance, son sexe et son statut matrimonial (célibataire ou en couple)
2. Ajouter un constructeur à la classe `Personne`,
3. Écrire une méthode `afficheInfos` qui affiche les chaînes de caractère suivantes :
   "M. Gaston Martin est né en 1975, il est célibataire."
   "Mme. Léonie Dupond est née en 1985, elle est en couple."
4. Écrire une fonction `main` qui déclare 3 variables de type `Personne`, crée les 3 instances associées et affiche les information les concernant.
5. Ajouter une méthode `age` qui renvoie l'âge de l'individu en fonction d'une année donnée en paramètre.
6. Modifier la classe et le constructeur pour compter le nombre d'instances de la classe `Personne` créées.

Exercice 4 — *Animal de compagnie*

Soit la classe `AnimalDeCompagnie` qui a les deux propriétés suivantes : `String nom` et `int age`.

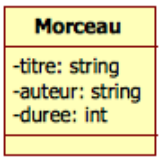
(1) Ecrire la classe `AnimalDeCompagnie`, elle comportera un constructeur avec deux paramètres et une méthode `Affichage` qui à partir d'un `AnimalDeCompagnie` dont les attributs `nom` et `age` valent respectivement `"bill"` et `3` et affiche le message suivant :
`L'animal s'appelle bill et il a 3 ans`

(2) Ecrire la méthode main qui instancie un `AnimalDeCompagnie` s'appellant `"bill"` et ayant `3` ans, et qui en affiche les caracteristiques.

Exercice 5 — *Lecteur de Playlistes*

(1) Nous allons d'abord représenter un ensemble de morceaux musicaux.

(a) A partir du diagramme de classe qui suit, définir le schéma de classe java `Morceau` correspondant :



(b) La compléter avec les `getters` et les `setters` (accesseurs aux variables d'instance).

(c) Ajouter un constructeur permettant d'initialiser toutes les variables d'instance de cette classe.

(d) On introduit maintenant 4 constantes publiques pour représenter 4 genres musicaux (nommées `GENRE1`, `GENRE2`, `GENRE3`, `GENRE4`) : `{ JAZZ, REGGAE, METAL, SKA }`.
Ajouter les `getters` et les `setters` pour la nouvelle variable d'instance genre de type `String`.
Le `setters` affichera un message d'erreur si le `genre` proposé n'appartient pas à l'ensemble `{ JAZZ, REGGAE, METAL, SKA }`.

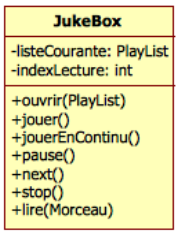
(2) Une `PlayList` rassemble une suite de `Morceau` sélectionnés.

(a) La classe java `PlayList` qui les représente stocke les `Morceau` sous la forme d'un tableau dont la taille maximale sera passée en paramètre du constructeur.
En plus des `getters` sur la variable `taille`, et sur le `Morceau` d'indice `i`, cette classe fournit deux méthodes :
— `ajouter` : qui permet d'ajouter un nouveau `Morceau` de la `PlayList`.
— `supprimer` : qui permet de supprimer un `Morceau` de la `PlayList`.
Son modèle est représenté par le diagramme UML :

(b) Ajouter une variable de classe `compteurPlayList` pour comptabiliser à tout moment le nombre d'instances de `PlayList` existantes.

(3) Un `JukeBox` est associé à une `PlayList`. Il permet l'écoute de tout ou partie d'une `PlayList`. La variable d'instance `indexLecture` représente l'index dans la `PlayList` du `Morceau` en lecture. Les opérations disponibles sont :
— `ouvrir(PlayList)` : qui ouvre une `Playlist`.

— `jouer` : qui joue un `Morceau` en affichant le titre du `Morceau` courant et incrémente l'indice de lecture. Notons que lorsque la fin de la `PlayList` est atteinte, c'est le premier `Morceau` qui est joué.
— `jouerEnContinu` : qui joue les `Morceau` de la `Playlist` de l'indice `indexLecture` jusqu'à la fin de la `Playlist`.
— `next` : qui joue le `Morceau` suivant de la `PlayList`. La lecture se fait circulairement.
— `pause` : qui affiche le message : "–pause–" .
— `stop` : qui remet l'`indexLecture` à 0 ;
Son modèle est représenté par le diagramme UML :



Implanter la classe `JukeBox`.

(4) Il s'agit maintenant d'écrire un programme (classe contenant la méthode main) dont l'algorithme est le suivant :
   (a) construire 4 instances de `Morceau`
   (b) construire une `PlayList`
   (c) ajouter les 4 morceaux à la `PlayList`
   (d) construire un `JukeBox`
   (e) jouer la `PlayList` en continu
   (f) appuyer sur pause
   (g) jouer le morceau courant
   (h) jouer le morceau suivant
   (i) jouer le morceau courant
   (j) supprimer un morceau
   (k) jouer la `PlayList` en continu
   (l) stop
   (m) construire une nouvelle `PlayList`
   (n) y ajouter 2 morceaux
   (o) jouer le suivant, encore le suivant et encore le suivant.
   (p) afficher le nombre d'instances de `PlayList`