IRT22 - Programmation orientée objet Java

Devoir. Vendredi 13/05/2022. Durée : 1h30.

Question de cours [5 pts]

1. C'est quoi la différence entre la surdéfinition et la redéfinition ? Donner un exemple.
2. En quoi les interfaces Java sont liées au concept du polymorphisme ?
3. Énumérer les différents cas possibles d'initialisation d'un attribut final.
4. Que signifie « `finally` » dans la gestion des exceptions.

Exercice 1 [9 pts]

Nous voulons concevoir une petite application de gestion des enseignants. Chaque enseignant a un ID (entier), nom et spécialité. Un enseignant peut être permanent ou vacataire. Un enseignant permanant est affilié à une structure (pôle ou département). Un vacataire a une catégorie (1 : universitaire, 2 : expert du milieu professionnel).

1. Créer la classe `Structure` qui a deux attributs : nom (chaîne de caractères) et type (entier, 1 : département et 2 : pôle). Ajouter un constructeur, getters et setters. Le constructeur lève une exception `ErrStructure` (à créer) si le type de la structure est incorrect.
2. Créer une classe abstraite `Enseignant`. Nous voulons avoir des ID auto-incrémentés pour les enseignants permanents et pour les enseignants vacataires de manière indépendante. La classe contient une méthode `affiche()` qui affiche les valeurs des attributs.
3. Créer les classes `EnseignantPermanent` et `EnseignatVacataire`. L'attribut `structure` de la classe `EnseignantPermanent` est un attribut de type objet. Redéfinir la méthode affiche pour chaque classe.
4. Écrire une classe `Test` qui crée et affiche un tableau d'enseignants qui contient deux enseignants permanents et deux vacataires. La méthode `main` doit gérer l'exception `ErrStructure`.

Exercice 2 [6 pts]

Nous avons une classe nommée `Entier` avec un attribut `val` de type `int`. Cette classe contient un constructeur et deux méthodes `getVal()` et `setVal()`.

Nous voulons créer une classe `Pile` qui a un attribut `elements` de type `ArrayList` contenant des objets de type `Entier`.

1. Écrire la classe `Pile` en ajoutant un constructeur qui initialise l'attribut `elements` à une liste vide. Ajouter les méthodes `empiler()`, `depiler()` et `estVide()`.
2. Créer une exception `PileVideException` et modifier la méthode `depiler()` pour lever cette exception si la pile est vide.
3. Nous voulons ajouter une méthode `tri()` à la classe `Pile` pour trier les éléments de la pile dans un ordre croissant selon les valeurs des objets Entier.
    a. Apporter les modifications nécessaires à la classe `Entier` pour pouvoir appeler la méthode `Collections.sort()` en passant une liste d'objets Entier comme paramètre.
    b. Écrire la méthode `tri()`.