Développement informatique avancé orienté application - TP

Classes abstraites, interfaces et exceptions

Virginie Van den Schrieck

Ce TP se base sur les classes réalisées pendant les TPs précédents. Si vous n'avez pas fini ce dernier, mettez-vous en ordre au plus vite, et soyez sûrs de rattraper votre retard avant la prochaine séance !

Les notions illustrées par les exercices ci-dessous sont expliquées dans les section 2.6 à 2.9 du site INGInious.

1 Classes abstraites

1.1 Des formes

1. Ecrivez, documentez et testez une classe abstraite `Shape`. Les objets de type `Shape` possède un identifiant, qui est leur numéro d'ordre de création (le premier possède l'identifiant 1, le second 2, etc.). Ils possèdent également une absisse et une ordonnée (entiers), qui identifient leur emplacement dans le plan. La classe `Shape` offre deux méthodes abstraites : `surface()` et `perimetre()`. La classe `Shape` offre aussi la méthode `getNom()`, qui renvoie le nom de la forme suivie de son identifiant. Ainsi, si on crée un carré comme troisième forme, elle renverra Carre-3 (conseil : Utilisez la méthode `getClass().getSimpleName()` de la classe `Object`). La classe implémente également la méthode `toString()`. Cette méthode renvoie une chaîne de caractère composée de quatre lignes : Le nom de la forme, son emplacement, son périmètre et sa surface.
2. Ecrivez, documentez et testez trois classes concrètes, `Point`, `Cercle` et `Carre`, qui étendent `Shape`. Un point a une surface et un périmètre nul. Un cercle est caractérisé par son rayon et sa coordonnée représente le centre, tandis qu'un carré est caractérisé par la longueur de son coté et la coordonnée représente son point inférieur gauche. Implémentez les constructeurs ad-hoc pour chacune des formes, en n'oubliant pas de configurer correctement l'identifiant. Implémentez les méthodes `surface()` et `perimetre()` pour chacune.
3. Représentez le diagramme UML de ces classes

1.2 Gestion d'école

Vous devez écrire le logiciel de gestion d'une école. Celui ci doit gérer les personnes, à savoir leur nom, prénom, date de naissance et date d'arrivée dans l'établissement. On doit pouvoir obtenir l'âge d'une personne. Il y a deux types de personnes : des professeurs et des étudiants. Les étudiants ont un matricule.

On souhaite calculer l'ancienneté d'une personne :

— Dans le cas d'un professeur, l'ancienneté est le nombre d'années de travail dans l'établissement. Si l'employé a été engagé avant ses 23 ans, l'ancienneté ne commence à courir qu'à partir du moment où il a 23 ans résolus.

— Pour les étudiants, il s'agit du nombre d'années depuis la première inscription dans l'établissement.

On souhaite également obtenir le matricule des étudiants. Ce matricule est constitué de l'année d'arrivée dans l'établissement, puis des initiales du nom et du prénom. Par exemple, David Dupont, inscrit en 2012, aura comme matricule : `2012DD`.

Pour toutes les personnes, la méthode `toString()` renvoie le nom, le prénom, l'âge et l'ancienneté. Pour les étudiants, elle imprime en plus le matricule.

Il vous est demandé de dessiner le diagramme UML, spécifier, tester et implémenter ce logiciel.

2 Interfaces

2.1 Définition d'interfaces

Définissez deux interfaces, en définissant soigneusement la spécification de chaque méthode :

1. Une interface `Affichable` avec une méthode affiche(), qui a pour effet d'imprimer la représentation textuelle de l'Affichable à la console.
2. Une interface `Transformable`, qui contient deux méthodes : `deplace(int deltaX, int deltaY)` et `agrandit(int facteur)`, qui a pour effet de multiplier les dimensions de l'objet du facteur indiqué.

Ensuite, modifiez les diagrammes UML et les classes précédemment créées pour que :

— Les personnes et les formes soient Affichables

— Les formes soient Transformables

2.2 Exploration de l'API Java

Parcourez l'API Java et identifiez trois interfaces différentes. Pour chacune, décrivez à quoi elle sert et listez les méthodes qu'elle contient.

3 Exceptions

3.1 Etudiant

Reprenez la classe Etudiant créée au début de ce TP. Créez une exception DateArriveeInvalideException qui sera lancée lorsqu'on tente d'encoder un étudiant avec une année de première inscription antérieure à l'an 2000, et postérieure à l'année en cours.

Testez le lancement de votre exception avec un test JUnit.

Ecrivez également une méthode `main()` qui crée un étudiant sur base des arguments de la ligne de commande, et renvoie un message propre en cas d'année d'inscription incorrecte.