FICHE 5 : VARIABLES ET MÉTHODES D'INSTANCE ET DE CLASSE, EXCEPTIONS

Objectifs

- Distinguer les méthodes et variables d'instance et de classe
- Comprendre l'importance du contexte d'invocation (static ou non)
- Manipuler en UML et en Java les variables et méthodes d'instance et de classe.
- Pouvoir lancer une exception et être capable de traiter les exceptions.

Vocabulaire

static	final	constante	
variable de classe	variable d'instance	méthode de classe	méthode d'instance
Exception	throw	try {…} catch(…) {…}	

Exercices

Créez, dans le répertoire APOO, un projet intitulé **APOO_fiche5** afin d'y implémenter les classes demandées dans cette fiche.

1. Article

Reprenez votre classe `Article`. Si vous n'avez pas cette classe, vous pouvez récupérer celle fournie sur moodle.

a) Modifiez le code afin de faire ce qui suit :
   - Ajoutez une constante correspondant au taux de TVA à mettre par défaut lorsque celui-ci n'est pas passé en paramètre à la création et modifiez le constructeur adéquat afin d'utiliser cette constante
   - Il faut pouvoir déterminer le nombre d'articles créés et il faut ajouter une méthode de classe qui renvoie ce nombre.

   Testez votre code ! Écrivez un programme qui crée quelques articles en utilisant les deux constructeurs. Affichez le nombre d'articles créé avant de créer un article, entre chaque article créé et après avoir créé tous vos articles. Affichez aussi les articles créés.

b) Il faut maintenant ajouter le test des paramètres des constructeurs et des méthodes. Il faut lancer une `IllegalArgumentException` en cas de paramètre invalide sachant que :
   - la référence et le nom d'un article ne peuvent pas être null ;
   - la référence et le nom d'un article ne peuvent pas être une chaîne de caractères vide ("") ;
   - le prix d'un article doit être strictement positif ;
   - le taux de TVA doit être compris entre 0 et 100 (bornes incluses) ;
   - une réduction doit être comprise entre 0 et 100 (bornes exclues).

   Écrivez un programme qui teste toutes les situations exceptionnelles et affiche des messages d'erreur adéquats.

2. Appel Téléphonique

On vous demande de développer une classe qui permettra de calculer le prix d'un appel téléphonique fait à partir d'une ligne de téléphone fixe. Il faut garder les informations suivantes concernant un appel :

- la date et l'heure de l'appel (utilisez la classe `LocalDateTime !`);

- le numéro appelé ;
- la durée de l'appel (utilisez la classe `Duration` !) ;
- le tarif à la minute.

Il faudra deux constructeurs : le premier prendra en paramètres toutes les valeurs nécessaires pour initialiser tous les attributs, le deuxième prendra les mêmes paramètres à l'exception du tarif qui sera initialisé à une valeur par défaut (actuellement cette valeur est de 0.15 euro par minute). Cette dernière est la même pour tous les appels téléphoniques et doit aussi être gardée. De plus, il faut pouvoir récupérer et aussi modifier la valeur par défaut du tarif.

Il faut aussi :

- un getter pour chaque attribut ;
- pouvoir calculer le coût d'un appel téléphonique (la minute entamée compte) ;
- pouvoir récupérer les informations d'un appel téléphonique sous forme de texte.

a) Donnez la représentation en UML de la classe `AppelTelephonique`.
b) Une fois votre diagramme validé, implémentez cette classe en java. N'oubliez pas de tester les paramètres sachant que :
   - les paramètres, correspondant à des objet, ne peuvent être `null` ;
   - un tarif doit être strictement supérieur à 0 ;
   - la durée ne peut pas être inférieure ou égale à 0 ;
   - le moment de l'appel (date et heure) doit être passé.

   Remarque : le début du toString contenant le formatage de la date est fourni sur moodle.
c) Écrivez une classe permettant de tester la classe `AppelTelephonique`.

3. Etudiant et Serie

Récupérez vos classes `Etudiant` et `Serie` de la fiche 3. Modifiez la méthode permettant de changer la série d'un `Etudiant` pour qu'elle renvoie `void` plutôt qu'un `boolean`. À la place de ce `boolean`, lancez des `IllegalArgumentException` si le paramètre est invalide ou si l'étudiant est déjà dans la série passée en paramètre et lancez une `IllegalStateException` si l'étudiant est délégué de sa série.
Modifiez la méthode qui permet d'élire le délégué de votre classe `Serie` pour qu'elle renvoie `void` plutôt qu'un `boolean`. À la place de ce `boolean`, vous lancerez une `IllegalArgumentException` en cas de paramètre invalide et une `IllegalStateException` si la série a déjà un délégué.
Adaptez la classe `TestEtudiantSerie` afin que le programme s'exécute jusqu'au bout.

4. Bonus : personne

Cet énoncé a pour objet de représenter une personne avec ses deux parents. Une personne possède un nom de famille, un prénom, un genre ('M' ou 'F') et un numéro de registre national. Ces informations ne sont pas modifiables mais doivent être consultables

Elle possède également un domicile de type `Adresse` qui est accessible et modifiable.
Une personne possède maximum un père et maximum une mère. Ils sont accessibles mais non modifiables.

Il faut aussi pouvoir signifier si une personne est un descendant (de degré 2 maximum) d'une autre personne.

Il faut aussi une méthode toString prenant en compte le prénom, le nom, le genre, le numéro de registre national et des parents (seulement leur prénom et nom ou inconnu quand il n'y en a pas) de la personne.

1. Donnez le diagramme de classes en UML. Dans le comportement de cette classe, n'oubliez pas qu'il faut pouvoir l'instancier. L'adresse n'est pas fournie lors de l'instanciation. Il doit être possible de créer un une personne en précisant 0, 1 ou 2 parents. Il faut s'assurer de mettre le bon parent comme père ou comme mère. Si un paramètre est spécifié, il ne peut être null. **Soyez attentif aux associations !**

2. Implémentez cette classe en Java. Le constructeur et les méthodes doivent tester leurs paramètres.

3. Construire un programme de tests qui correspond au scénario suivant : Elizabeth Bonte (numéro de registre national : 01.10.25-004.16) a deux parents : Philippe Bonte (numéro de registre national :70.04.15-001.61) et Julie Maes (numéro de registre national : 73.01.20-002.65). Ils habitent tous les trois à l'adresse suivante : 142 Grand rue, 7000 Mons. Albert Bonte (numéro de registre national 44.06.06-001.90) et Marie Leclercq (numéro de registre national 47.09.11-002.23) sont les parents de Philippe. On ne connait pas leur adresse. André Bonte (numéro de registre national : 12.11.03-001.07) est un parent d'Albert, sans adresse connue. Affichez les personnes créées et utilisez-les également afin de tester votre méthode qui indique la descendance.