# FICHE 4 : EXCEPTIONS

## Objectifs

- Pouvoir lancer une exception et être capable de traiter les exceptions.

## Vocabulaire

| | | |
|---|---|---|
| Exception | throw | try {…} catch(…) {…} |

## Exercices

Dans les exercices qui suivent, un objet `null` passé en paramètre sera considéré comme invalide. Pour savoir si un objet est `null`, cela se fait de la manière suivante :
if (objet == null) …

### 1. Article

Modifiez votre classe Article afin que les constructeurs et les méthodes testent leurs paramètres et lancent une `IllegalArgumentException` en cas de paramètre invalide sachant que :

- la référence et le nom d'un article ne peuvent pas être `null ;`
- 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. Etudiant et Serie

Modifiez la méthode `changerSerie` de votre classe `Etudiant` pour qu'elles renvoient `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 `elireDelegue` de votre classe `Serie` pour qu'elles renvoient `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.

### 3. Habitation – Adresse

1. Récupérez votre classe `Adresse` et modifiez-la afin que le constructeur teste ses paramètres et lance une exception en cas de problème.
2. On veut maintenant définir une classe `Habitation` qui doit garder les informations suivantes :
   - son propriétaire ;
   - son adresse ;
   - son revenu cadastral (un réel strictement positif);
   - son année de construction ;
   - sa surface (un entier strictement positif).

Le constructeur recevra en paramètre toutes les valeurs des attributs dans l'ordre ci-dessus.

Il faudra également un setter pour le propriétaire, tous les getters et la méthode `toString()` qui tient compte de tous les attributs.

De plus, il faudra qu'on puisse faire les opérations suivantes:

- mettre à jour le revenu cadastral en le multipliant par un coefficient donné. Ce coefficient doit être strictement supérieur à 1 ;
- calculer le précompte immobilier. Celui-ci est égal à un certain pourcentage du revenu cadastral. Ce pourcentage doit être un réel strictement positif et inférieur à 100.

a) On vous demande de réaliser le diagramme UML correspondant à une `Habitation`.
b) Lorsque votre diagramme est valide (demandez-le à un professeur), implémentez votre classe en Java en veillant à bien tester les paramètres et à lancer, le cas échéant, une `IllegalArgumentException`.
c) Écrivez un programme qui teste toutes les situations exceptionnelles et affiche des messages d'erreur adéquats.

## 4. 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.