**GESTION DES EXCEPTIONS\_JAVA**

Table des matières

[Chapitre 1 INTRODUCTION 2](#_Toc102921862)

[Chapitre 2 : ARBORESCENCE (HIERARCHIE) DES EXCEPTIONS 2](#_Toc102921863)

[Chapitre 3 TRAITEMENT DES EXCEPTIONS : 3](#_Toc102921864)

# 

# INTRODUCTION

Le traitement des erreurs désigne gérer des situations où notre programme se retrouve dans des situations illogiques : par exemple la division par zéro, consommer un fichier csv dans un répertoire vide, etc…

Certains cas d’erreurs peuvent être prévus :

Exemple :

* Erreurs d’entrée et sortie de fichiers
* Erreurs de saisie de données par l’utilisateur

Le programme peut alors :

* Laisser planter le code à l’endroit où l’erreur est détectée
* Manifester explicitement le problème à la couche supérieure
* Tenter une correction

Notion d’exception : est un objet java (une instance de la classe Exception) correspond à un type d’erreurs dans le code,

Un objet Exception est une bulle logicielle produite dans une situation d’exceptions qui va ***remonter la pile d’exceptions*** pour trouver une portion de code qui la traite. Si cette portion n’existe pas alors le code crash.

***La pile d’exceptions :*** dans le code il y a plusieurs méthodes (meth1, meth2, meth3, …methn), la meth1 appelle la méth2 qui appelle la meth3 qui appelle la meth4…

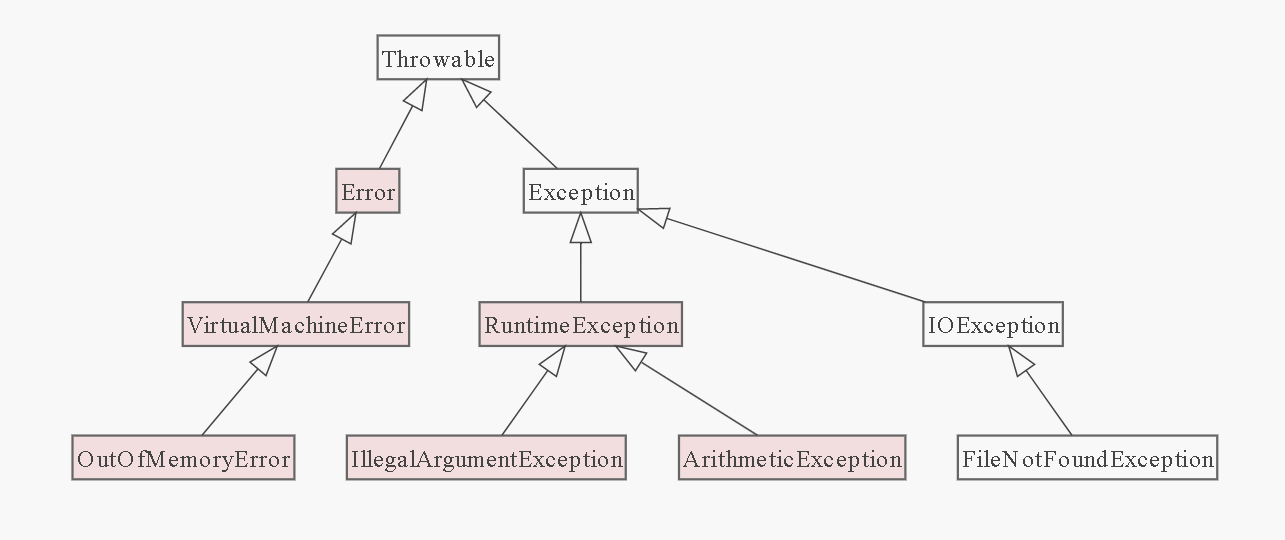
Si on détecte un problème au niveau de la methn alors la bulle remonte toute la pile d’exécution (jusqu’à la meth1) (la pile d’exception dans ce cas) en cherchant une méthode qui la traite.

***NB :*** Lever une exception consiste à signaler quelque chose d’exceptionnel.

Capturer l’exception consiste à essayer de la capturer.

La plupart des API java soulèvent des exceptions.

# : ARBORESCENCE (HIERARCHIE) DES EXCEPTIONS



En Java, la notion d'exception est définie en pratique par la classe ***Throwable*** (que les anglicistes parmi vous auront traduit approximativement par « qui peut être lancé »). Tout objet étant une instance d'une classe qui hérite de ***Throwable*** sera considéré, en Java, comme une exception. Ainsi, par exemple, la classe java.lang.ArithmeticException, a pour arbre d'héritage : java.lang.ArithmeticException → java.lang.RuntimeException → java.lang.Exception → java.lang.Throwable (puis java.lang.Object, )

***La classe Error*** est la classe des exceptions « graves », qui ne devraient pas être récupérées (erreurs système, plus de mémoire... etc.)

***La classe Exception*** correspond aux exceptions qu'on peut vouloir récupérer. Celle-ci a une sous-classe RuntimeException

La classe Exception a beaucoup de descendants définis dans les paquets Java. Ces descendants indiquent de différents types d'Exception qui peuvent se produire. Par exemple, NegativeArraySizeException est jetée lorsqu’on essaie de créer un tableau qui contient des éléments négatifs. Une sous-classe Exception a une signification particulière dans le langage Java, c'est RuntimeException.

La classe RuntimeException représente des exceptions qui se produisent lors de l'exécution du programme. Un exemple d'exécution d'une exception est NullPointerException, qui se produit lorsqu’on accède à une méthode ou au champ d'un objet via une référence null. Une NullPointerException peut se produire lorsqu’un programme essaie de faire la référence à un objet. Le coût de la vérification de l'exception dépasse souvent l'avantage de la détecter.

Comme des exceptions Runtime sont très omniprésentes donc lorsqu’on essaie de les attraper ou les spécifier, ce sera un exercice infructueux. Le compilateur Java ne vérifie pas ces exceptions dans le processus de compilation du code.

# TRAITEMENT DES EXCEPTIONS :

Comprendre la notion « d’erreur » en Java :

Avant d’entrer dans la gestion d’erreur proprement dite, il faut commencer par comprendre la notion d’erreur et leurs potentielles sources. Les erreurs lors de l’exécution d’une application peuvent être divisées en 2 :

1. **Les erreurs de bas niveau** qui impliquent la violation de contraintes, telles que :

* Le déréférencement d’un pointeur nul
* L’accès hors limite à un tableau
* La division par zéro
* La tentative d’ouverture d’un fichier inexistant
* La mauvaise conversion (par exemple, conversion d’un booléen en un nombre entier).

1. Et des **erreurs logiques de plus haut niveau**, telles que la violation de la précondition d’une fonction :

* Appel de la méthode « pop » de la classe Stack avec une pile vide
* Appel d’une fonction « factorielle » avec un nombre négatif

Les erreurs logiques peuvent entraîner des erreurs de bas niveau si elles ne sont pas détectées. Souvent, il est préférable de les détecter pour fournir un meilleur retour d’information à l’utilisateur de votre application.

En Java, toute exception sous contrôle lancée doit être :

* Soit rattrapée et traitée localement dans la méthode dans laquelle elle a été lancée ;
* Soit déclarée explicitement dans la signature de la méthode comme pouvant être lancée.

L’intérêt du traitement du code est :

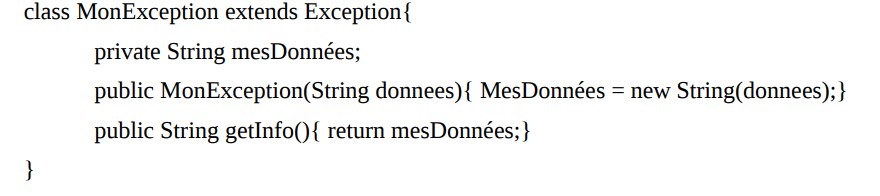
1. Cette syntaxe permet de séparer très clairement le code exécuté lors d'un comportement « nominal » du programme, et le code dédié au traitement d'erreur.
2. Cette syntaxe permet de classifier clairement le code dédié à la gestion de chaque type d'erreur. De même, le code dédié à un type particulier d'erreur est factorisé à un seul endroit (alors que cette erreur peut survenir à plusieurs endroits), évitant ainsi la duplication

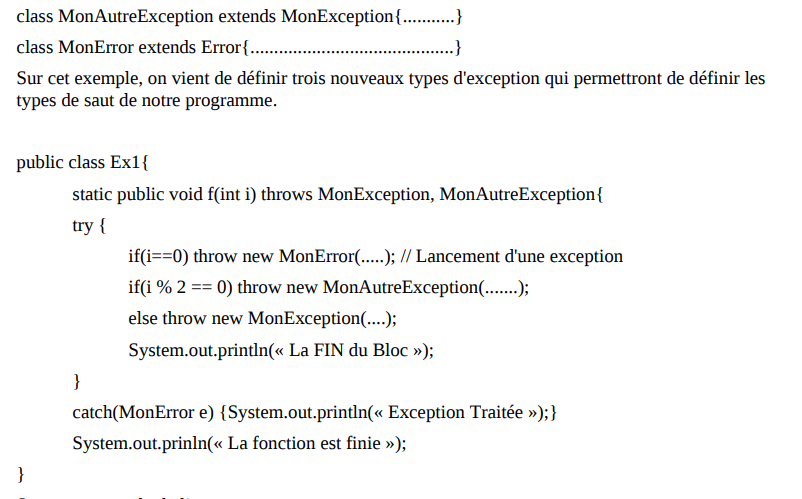
**LA SYNTAXE DE LA GESTION DES EXCEPTIONS :**

La syntaxe des exceptions.

La gestion des exceptions en Java se fait par l'utilisation de trois mots clefs **try, catch, throw**.

* **Le mot clef try** sert à définir un bloc dans lequel les exceptions sont susceptibles d'être capturées. On parle aussi de sensibilisation d'un bloc à un ensemble d'exceptions. La capture des exceptions est confiée à des gestionnaires d'exception.
* Après la définition de ce bloc se trouve la liste des gestionnaires d'exception. Un gestionnaire d'exception est défini en utilisant le **mot réservé catch**. La liste des gestionnaires d'exception ressemble à un ensemble de fonctions surchargées.
* Une exception est levée en utilisant le **mot clef throw**. L'identification de l'exception, se fait en utilisant le type d'un objet.





**Ordre de déclaration des clauses catch**.

Comme déjà vu, les clauses catch d'un bloc try sont traitées les unes après les autres dans l'ordre de leur déclaration. La première clause catch qui est compatible avec le type de l'exception est exécuté et le programme continue son exécution après la dernière clause catch du bloc try. **Il faut donc déclarer les clauses catch de la plus précise à la plus générale**