# La gestion des exceptions

La gestion des exceptions représente le mécanisme de gestion des erreurs.

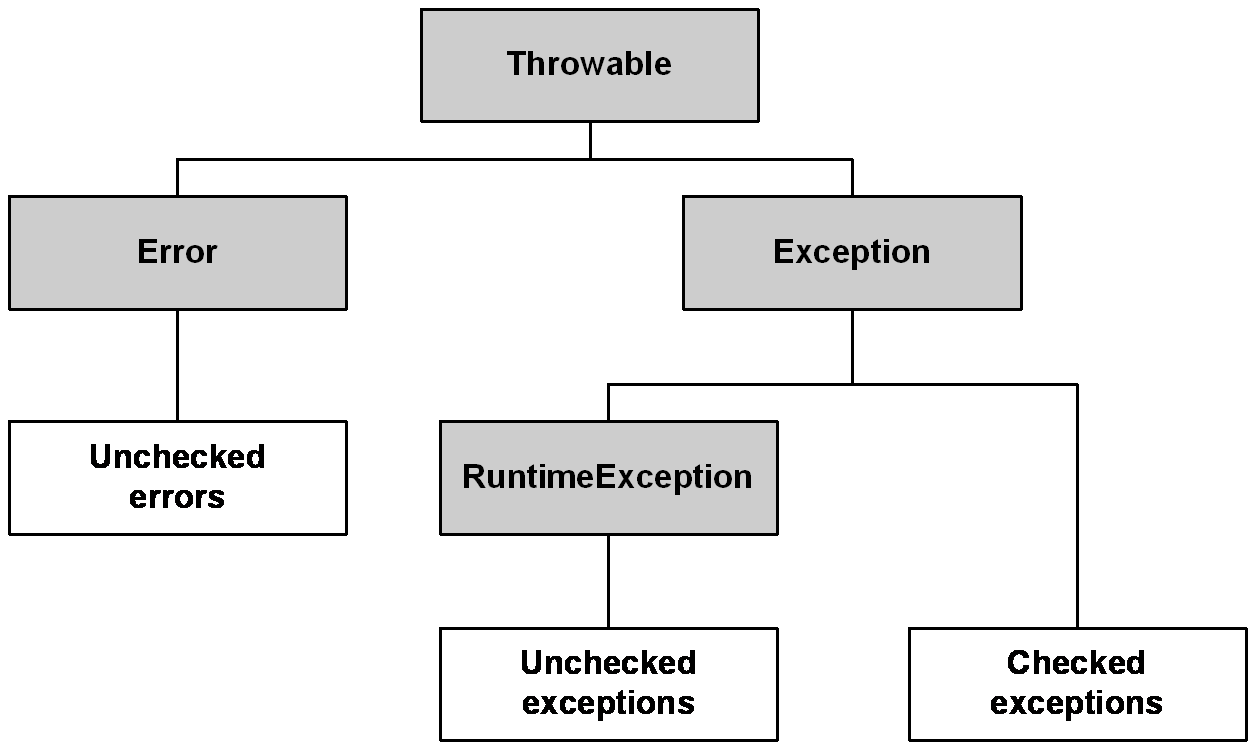
On va pouvoir :

* Lancer ou propager une exception
* Identifier une partie de code susceptible de générer une exception et intercepter celle-ci afin de la traiter et éventuellement d’afficher un message d’erreur personnalisé

En Java, on distingue trois types d'erreurs, qui sont de degrés de gravité différents, à savoir :

* Les erreurs graves qui sont représentées par la classe **java.lang.Error**, ces erreurs ne sont généralement pas interceptées dans le code, il s’agit généralement d’un problème grave, elles doivent donc causer l’arrêt du programme
  + **VirtualMachineError :** ces erreurs empêchent le bon fonctionnement de la JVM
    - **StackOverflowError** en cas de boucle infinie
    - **OutOfMemoryError** quand la JVM manque de mémoire
* Les erreurs qui doivent généralement être traitées et qui sont représentées par la classe **java.lang.Exception**. Elles doivent être déclarées dans la signature de la méthode avec le mot clé **throws**, ces exceptions sont donc **explicites**.  
  Il s’agit d’erreurs au niveau **applicatif** qui vont se **propager** dans le code en ignorant les instructions suivantes.
* Les erreurs qui peuvent ne pas être traitées et qui sont des objets de la classe **java.lang.RuntimeException** qui hérite de **java.lang.Exception**. Ces exceptions sont **implicites**, elles peuvent être levées sans déclaration préalable ;
  + **ArrayIndexOutOfBoundsExceptions** quand on essaie d’accéder à un élément d’un tableau hors limite

Toutes ces classes héritent directement ou indirectement de la classe j**ava.lang.Throwable**.



### Exception personnalisée

Pour créer son propre type d'exception, il faut écrire une classe héritant de la classe **Exception**.

La classe **Exception** contient 4 constructeurs publics et il est souvent préférable d'utiliser les mêmes constructeurs, afin de simplifier leurs créations et l'encapsulation d'exception.

Exemple :

**public** **class** MonException **extends** Exception {

/\*\*

\* Crée une nouvelle instance de MonException.

\*/

**public** MonException () {

**super**();

}

/\*\* Crée une nouvelle instance de MonException.

\* **@param** *message* Le message détaillant l’exception

\*/

**public** MonException(String message) {

**super**(message);

}

/\*\*

\* Crée une nouvelle instance de MonException.

\* **@param** *cause* L’exception à l’origine de cette exception

\*/

**public** MonException (Throwable cause) {

**super**(cause);

}

/\*\* Crée une nouvelle instance de MonException.

\* **@param** *message* Le message détaillant l’exception

\* **@param** *cause* L’exception à l’origine de cette exception

\*/

**public** MonException (String message, Throwable cause) {

**super**(message, cause);

}

}

### Lancer une exception : *throw*

On lance une exception grâce au mot clé **throw.**

L’exception lancée doit hériter de la classe exception.

**public** **class** MyClass {

**public** **void** maMethode(String s) **throws** MyException{

**if**(s==**null**){

**throw** **new** MyException();

}

}

**public** **class** MyException **extends** Exception {

}

Si une méthode lance une ou plusieurs exceptions différentes de java.lang.RuntimeException (et des ses sous-classes), elle doit le mentionner dans une clause **throws** figurant dans son en-tête, les différentes exceptions seront séparées par des virgules.

### Gestion des exceptions : try catch finally

**try**{

//code susceptible de générer une exception

maMethode1() ;

maMethode2() ;

}**catch**(Exception1 e){

//on intercepte et traite le type Exception1

} **catch**(Exception2 | Exception3 e){

//on intercepte et traite communément le type Exception2 et Exception3

}**finally**{

//

}

* Lors du déclenchement d’une exception dans le bloc **try,** les instructions suivant le code qui a déclenché l’exception ne seront pas exécutées et le code du premier bloc **catch** interceptant le type d’exception généré sera exécuté.

Ainsi, si **maMethode1** génère une exception de type **Exception1**, le code qui suit ne sera pas exécuté (maMethode2 ne sera donc pas exécutée) et le code du premier bloc catch sera exécuté.

* Il faut définir au minium 1 bloc catch.
* L'ordre des clauses catch a son importance, un type d'exception ne doit pas venir après un type de sa super classe. Il faut faire attention à l'ordre des clauses catch pour traiter en premier les exceptions les plus précises (sous-classes) avant les exceptions plus générales. Un message d'erreur est émis par le compilateur dans le cas contraire.
* Le bloc **finally** est facultatif. L'intérêt d'un tel bloc repose sur le fait que les instructions qu'il comporte **seront toujours exécutées**, qu'une exception soit levée ou pas.