JAVA Exceptions

Exception

- Exception: mécanisme de déroutement du programme, brise la séquentialité du programme (levée d'exception)

- Evènement « rare » qui perturbe le déroulement usuel des opérations

- Ce n'est pas nécessairement une erreur (ex : fin de fichier)
- Correspond souvent à une erreur possible (ex : saisie incorrecte)
- Peut être utilisée pour traiter des cas particuliers

Java n'oblige pas à traiter les exceptions

Vocabulaire

- try : essayer (d'exécuter des instructions)
- catch : attraper, intercepter et traiter le cas exceptionnel
- Java « lève » une exception, c.à.d. déclenche un évènement lorsqu'une telle situation se produit
- throw : lancer, diffuser, propager cet évènement

Intérêts : ce mécanisme va dans le sens de la sécurité, et permet d'obtenir des programmes plus lisibles

Syntaxe

```
try {
        // instructions susceptibles de lever (lancer) des exceptions
        // que l'on souhaite ou doit traiter
}
catch(Type1_d'exception exc) {
        // traitement de cette exception
}
catch(Type2_d'exception exc) {
        // traitement de cette exception
}
// ...
finally {   // bloc exécuté systématiquement s'il est présent
}
```

- Les blocs catch sont en nombre quelconque
- Le bloc finally est optionnel
- Exécution du bloc try : cas possibles:
  - Pas de lancement d'exception: on ignore les blocs catch
  - Une exception est lancée: on cherche le boc catch correspondant
    - S'il y en a un il est exécuté
    - Sinon l'exception « poursuit son chemin »
  - Si le bloc finally est présent il est exécuté

Ordre des clauses catch

- Les clauses catch sont traitées dans l'ordre de leur déclaration
- Il faut donc déclarer les clauses catch de la plus précise à la plus générale.

Exemple


```
int x, y, z;
      ....
try
{
      z = x / y;            // division éventuelle par 0
}
catch (ArithmeticException exc)  // un type d'exception prédéfini
{
      System.out.println("Division par 0");
}
```

Fonctionnement

Lorsqu'une exception apparaît, Java crée un objet du type du problème rencontré, et le diffuse.

- Cet objet peut ensuite être intercepté et traité par le catch approprié suivant le bloc try
- Si aucun catch ne convient, cet objet est diffusé vers le bloc englobant où est recherché un catch qui convient
- Et ainsi de suite jusqu'à un éventuel arrêt du programme

Délégation de la gestion d'une exception : l'instruction throws

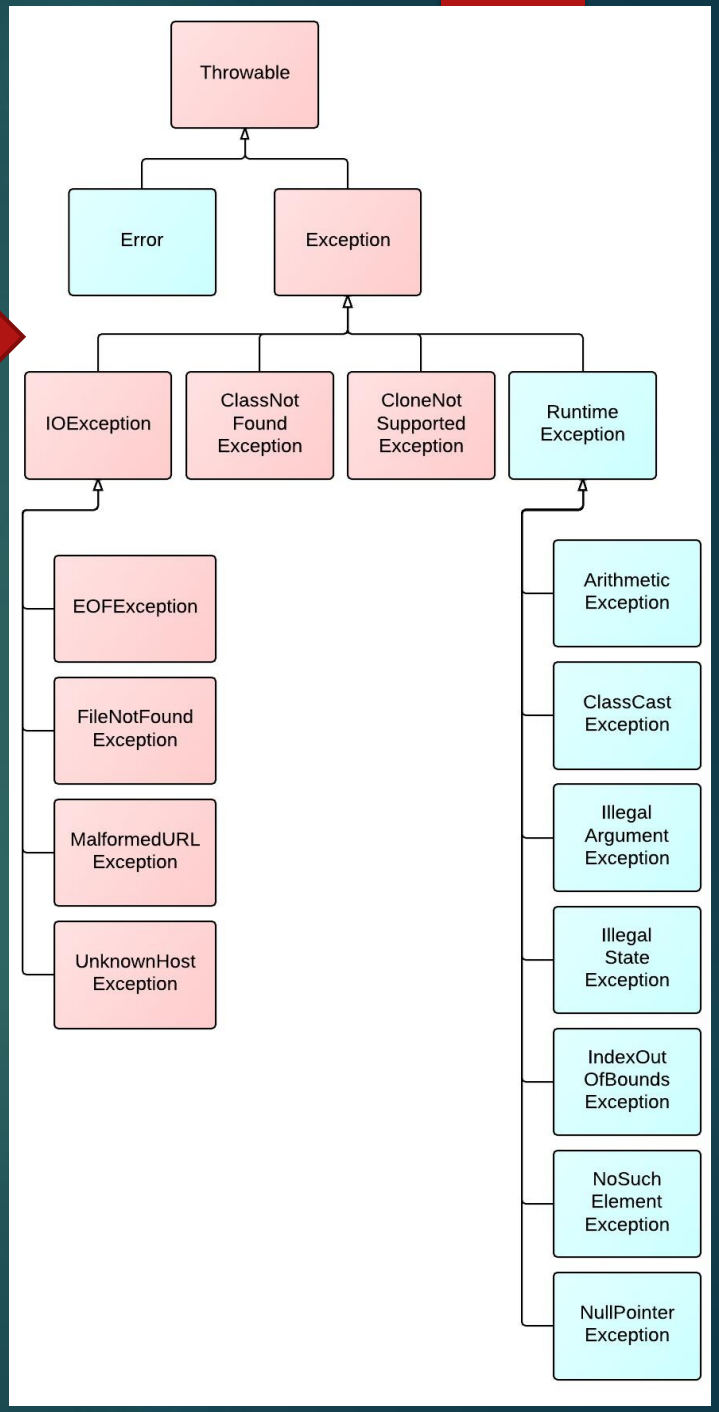
Lorsqu'une méthode ne traite pas (ou ne peut traiter) une exception qu'elle devrait traiter, elle doit déléguer ce traitement au bloc appelant (ou d'un niveau supérieur)

Pour ce faire, on utilise l'instruction throws

Exemple :

```
void maMethode (….) throws uneException
{
    // un bloc try dans lequel on lance uneException
            …
    // des blocs catch sans traitement d'objet de type uneException
}
```

Les exceptions prédéfinies



On peut créer ses propres classes d'exceptions

Un objet "exception" doit <u>toujours</u> être une instance d'une sous-classe de ***Throwable***, en général d'***Exception***

Créer ses propres exceptions (1)


```
public class monException extends Exception {
        ....
        public monException (String message) {
                super(message);        // par exemple
}
```

```
// pour créer et capturer l'exception :
try {
        if ( ... )              // l'exception se produit ici ...
                throw new monException ("mon message ");
}
catch (monException exc) {      // traitement
        // traitement approprié ...
}
```

Créer ses propres exceptions (2)

Sur l'exemple précédent, il faut savoir que les classes JAVA prédéfinies ***Exception*** (et ***Throwable***) disposent chacune d'un constructeur ayant en paramètre un *String* pour afficher un message d'erreur par défaut

Le constructeur de monException redéfinit le sens du message affiché

Le bloc catch permet de réaliser des actions complémentaires fonction du problème rencontré