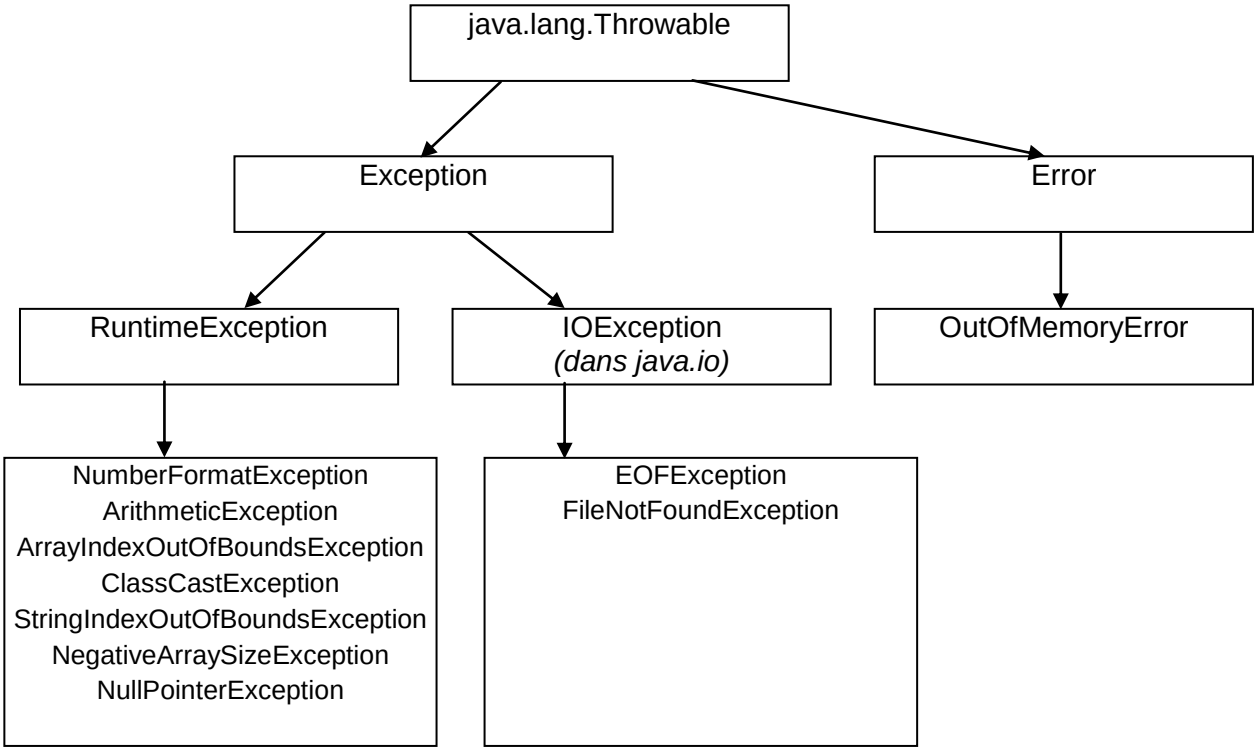
Principe des exceptions

Une exception correspond à un évènement anormal ou inattendu dans un programme qui conduit à l'arrêt du programme.
En java, lorsqu'une erreur survient à l'exécution d'un programme, un objet exception est généré.

Exemple :

Une instruction qui fait la division d'un nombre par 0 provoque une erreur, son exécution est interrompue et une exception représentée par un objet de la classe **ArithmeticException** est générée.

Hiérarchie et types d'exceptions



java.lang.Error : représente les erreurs graves de la machine virtuelle (erreur de type « out of memory »). En général, ces erreurs ne sont pas gérées par le programmeur.

java.lang.Exception : représente les erreurs liées à des évènements inattendus, qui sont souvent traitées de sorte qu'elles ne provoquent pas l'arrêt du programme.

Il existe deux types d'exceptions :

Checked :	Le compilateur impose de prendre en charge ces exceptions (comme IOException).
Unchecked :	Le compilateur n'impose aucun traitement particulier (classes dérivées de RuntimeException).

Gestion des exceptions

Deux façons de gérer les exceptions :

1. **try et catch** : Entourer la section de code sur laquelle on s'attend qu'une exception (une erreur) soit levée par **try** et **catch**.

2. **throws** : L'instruction **throws** est utilisée uniquement avec les méthodes. Elle permet de propager l'erreur à la méthode appelante.

Utilisation de try/catch

```
try {
    code pouvant provoquer une erreur
}
catch(Type1Exception e1) {
    code à exécuter en cas d'erreur1
}
catch(Type2Exception e2) {
    code à exécuter en cas d'erreur2
}
```

- Lorsqu'une erreur survient, java vérifie si le type d'erreur est du type indiqué dans la commande catch. Dans ce cas, le code dans le catch est exécuté.
- Placer toujours les exceptions les plus spécifiques d'abord.

Exemple 1 :

```
import javax.swing.*;
public class Exemple1{
    public static void main(String[] args)
    {
        int choix;
        try {
            choix = Integer.parseInt(JOptionPane.showInputDialog(
                        "entrez votre choix"));
            JOptionPane.showMessageDialog(null,"vous avez saisi " + choix);
        }
        catch(NumberFormatException e) {
            System.out.println("votre choix n'est pas un nombre");
        }
        System.exit(0);
    }
}
```

Saisie	Affichage
d	**votre choix n'est pas un nombre** *(sur la console)*
6	**vous avez saisi 6**

Exemple 2 :

```
import javax.swing.*;
public class Exemple2{
    public static void main(String[] args)
    {
        int choix = 0;
        boolean ok;
        String msgErr = "";
        // tant que la donnée n'est pas valide
        do {
            ok = true;
            try {
                choix = Integer.parseInt(JOptionPane.showInputDialog(
                            msgErr + "entrez votre choix"));
            }
            catch(NumberFormatException e) {
                msgErr = "Entrée invalide!\n";
                ok = false;
            }
        } while(!ok);
        JOptionPane.showMessageDialog(null,"vous avez saisi " + choix);
        System.exit(0);
    }
}
```

Utilisation de throws

La propagation d'une exception provoque la remontée dans l'appel des méthodes jusqu'à ce qu'un bloc catch acceptant cette exception soit trouvé.

Si aucun catch n'est trouvé, l'exception est capturée par l'interpréteur et le programme s'arrête.

Exemple 3 :

```
import javax.swing.*;
public class Exemple3{
    public static void main(String[] args) throws NumberFormatException
    {
        int choix;
        choix = Integer.parseInt(JOptionPane.showInputDialog(
                    "entrez votre choix"));
        JOptionPane.showMessageDialog(null,"vous avez saisi " + choix);
        System.exit(0);
    }
}
```

Exemple 4 :

```
// exemple exception arithmétique
public class Exemple4 {
    static void methode1() {
        int i = 1;
        int j = 0;
        i = i / j;
    }

    public static void main (String args[]) {
        methode1();
        System.exit(0);
    }
}
```

Résultat à l'exécution :

```
Fatal Exception occurred. Program will exit
```

REMARQUE :

Pour savoir si une méthode génère une exception, consulter la documentation de l'API de java. Par exemple :

public parseInt(String s) **throws NumberFormatException**

Exemple 4 corrigé :

```
// exemple exception arithmétique
class Exemple4Revu {
    static void methode1() {
        int i = 1;
        int j = 0;
        try {
            i = i / j;
        } catch(ArithmeticException e) {
            e.printStackTrace();     // affiche la trace d'exécution
            System.out.println("exception attrapée");
        }
    }

    public static void main (String args[]) {
        methode1();
        System.exit(0);
    }
}
```

Résultat à l'exécution :

```
java.lang.ArithmeticException: / by zero
    at Exemple4Revu.methode1(Exemple4Revu.java:7)
exception attrapée
at Exemple4Revu.main(Exemple4Revu.java:15)
```