

LES EXCEPTIONS

Les exceptions

- **Le programme idéal est un programme sans erreur...**
- Le compilateur est le premier niveau de contrôle des erreurs dites de syntaxe, de type, voire de sémantique.
- Le deuxième niveau de contrôle fait intervenir la notion d'exception.
- Lors du développement, on ne peut jamais savoir ce que l'utilisateur de la classe ou du programme va faire.
 - Par exemple, lors des spécifications, le concepteur a imposé que le paramètre **x** d'une fonction **f** ne soit jamais nul.
 - Si néanmoins l'utilisateur appelle cette fonction avec **0**, sans gestion du problème, le programme se plante et s'arrête définitivement !!!

Les exceptions

- Les exceptions permettent de séparer le bloc d'instructions de la gestion des erreurs pouvant survenir dans un bloc.

```
try {  
    // Code pouvant lever des exceptions  
} catch (Exception e) {  
    // Gestion de toutes les exceptions  
}
```

Le mot clé try indique que le code du bloc peut soulever des exceptions, et que si c'est le cas alors il y aura exécution d'un bloc marqué par le mot clé catch.

Un bloc catch prend en paramètre une instance d'une classe héritant de la classe Exception.

En définissant plusieurs blocs catch (prenant différentes classes d'exception), on peut réaliser un traitement différent selon l'exception soulevée.

Les exceptions

- La levée d'une exception provoque une remontée dans l'appel des méthodes jusqu'à ce qu'un bloc **catch** acceptant cette exception soit trouvé.
- Si aucun bloc **catch** n'est trouvé, l'exception est capturée par l'interpréteur et le programme s'arrête.
- L'appel à une méthode pouvant lever une exception doit :
 - soit être contenu dans un bloc **try/catch**
 - soit être situé dans une méthode propageant (**throws**) cette classe d'exception (-> la méthode ne traite pas l'exception, mais laisse la méthode appelante le faire).

Les exceptions

- Un bloc (optionnel) **finally** peut-être posé à la suite des **catch**
- Son contenu est exécuté après un **catch** ou après un **break**, un **continue** ou un **return** dans le bloc **try**

Les exceptions

Method Detail

```
Object getContent()
```

writeObject

```
try
void writeObject(Object obj)
throws IOException
    openConnection();
```

Write an object to the underlying storage or stream. The class that implements this interface defines how the object is written.

Parameters: **catch** (IOException e) {

obj - the object to be written

Throws:

finally {
IOException - Any of the usual Input/Output related exceptions.
...
}

...

}

```
void openConnection()
    throws IOException
```

```
{
    openSocket();
    sendRequest();
    receiveResponse();
}
```

```
void sendRequest()
    throws IOException
```

```
{
    write(header);
    write(body); // ERROR
}
```

Les exceptions

- La levée d'une exception passe par la création d'une instance d'une classe dérivant de **Exception** et l'utilisation du mot clé **throw**.

```
if(t == null) throw new NullPointerException();
```

- Deux constructeurs : un par défaut, l'autre prenant une chaîne de caractères en paramètre

```
if(t == null) throw new NullPointerException("t = null");
```

```

public class Fraction {

    private int numerateur, denominateur = 1 ;

    public Fraction(int numerateur) {
        this.numerateur = numerateur;
    }

```

```

    public Fraction(int numerateur, int denominateur) throws ZeroDivideException {
        this.numerateur = numerateur;
        // this.denominateur = denominateur;
        setDenominateur(denominateur);
    }

```

```

    public void setNumerateur(int numerateur) {
        this.numerateur = numerateur;
    }

```

```

    public void setDenominateur(int denominateur) throws ZeroDivideException{
        if(denominateur == 0) throw new ZeroDivideException() ;
        this.denominateur = denominateur;
    }

```

```

}

```

```

public class ZeroDivideException extends Exception{

    @Override
    public String toString() {
        return "DIVISION PAR ZERO !!!" ;
    }

}

```

```

public static void main(String[] args) {

    Fraction f1 = new Fraction(10) ;
    try {
        f1.setDenominateur(0);
    } catch (ZeroDivideException e) {
        System.err.println("Problème : " + e);
    }

    System.out.println();

    Fraction f2 ;
    try {
        f2 = new Fraction(20, 0) ;
    } catch (ZeroDivideException e) {
        e.printStackTrace();
    }

}

```