Cours 3 POO : try/catch

Gérer les situations problématiques

Une situation qui ne correspond pas au fonctionnemment normal d'une méthode

Il y a deux manières de gérer ce genre de situations :

- Un code de retour, pas simple à traiter
- Les exceptions
  - Signale là où le problème se pose dans le code
  - Sépare le traitement des cas normaux de celui des cas exceptionnels
  - Les cas particuliers sont transmis au gestionnaire d'exception

A quoi ça sert ?

Des portions de code peuvent lancer des exceptions Les classes d'exceptions se nomment par convention `NatureDuProblèmeException`

Les exceptions se documentent avec `@throws/@exception`

Concrètement

Lever une exception

Les exceptions ont la syntaxe suivante :

```java
try {
    ... /** code normal **/
} catch (classeDException e) {
    ... /** code en cas de problème **/
}
```

`catch` n'est pas bloquant : le flux d'exécution reprend normalement après (un peu comme un if/else)

Si une méthode peut lancer une exception, il faut l'expliciter dans la signature de la fonction

```java
public int method( ... ) throws
    FileNotFoundException (, Exception2, ... ,
    ExceptionN)
```

Déclencher une exception

Pour produire une exception, on utilise le mot clé `throw` de la manière suivante :

```java
throw new IllegalArgumentException("index" + index " is not valid")
```

> ⚠ Warning
>
> Les exceptions se testent !! On utilise l'assertion `assertThrows(ExpectedException.class, tested_expresssion)` Attention, tested_expression est un callback `() -> {}` équivalent à `() => {}` en js