POO Résumé

Lorsque l'interpréteur Java rencontre le mot-clé **static** devant une variable (**variable de classe**) il réserve un seul et unique emplacement mémoire pour cette variable.

Si ce mot-clé est absent, l'interpréteur peut construire en mémoire la variable déclarée non static (**variable d'instance**) en plusieurs exemplaires.

Cette présence ou cette absence du mot-clé static permet de différencier les variables des objets.

Les objets sont définis en mémoire par l'intermédiaire d'une **adresse** (**référence**).

Lorsqu'un objet est passé en paramètre d'une fonction, la valeur passée au paramètre formel est l'adresse de l'objet.

De cette façon, si la méthode transforme les données du paramètre formel, elle modifie aussi les données de l'objet effectivement passé en paramètre.

Ainsi, tout objet passé en paramètre d'une méthode voit, en sortie de la méthode, ses données transformées par la méthode.

Ce mode de transmission des données est appelé **passage** de paramètres **par référence**.

L'objectif principal de la programmation objet est d'écrire des programmes qui contrôlent par eux-même le bien-fondé des opérations qui leur sont appliquées.

Ce contrôle est réalisé grâce au principe d'**encapsulation** des données. Par ce terme, il faut comprendre que les données d'un objet sont protégées, de la même façon qu'un médicament est protégé par la capsule qui l'entoure.

L'encapsulation passe par le contrôle des données et des comportements de l'objet à travers les niveaux de **protection**, l'**accès** contrôlées et la notion de **constructeurs** de classe.

Le langage Java propose trois niveaux de protection :

- `public`
- `private`
- `protected`

Lorsqu'une donée est totalement protégée (`private`), elle ne peut être modifiée que par les méthodes de la classe où la donnée est définie.

On distingue les méthodes qui consultent la valeur d'une donnée sans pouvoir la modifier (**accès en consultation**) et celles qui modifient après contrôle et validation la valeur de la donnée (**accès en modification**).

Les constructeurs sont des méthodes particulières, déclarées uniquement `public`, qui portent le même nom que la classe où ils sont définis. Ils permettent le contrôle et la validation des données dès leur initialisation.

Par défaut, si aucun constructeur n'est défini dans une classe, le langage Java propose un constructeur par défaut, qui initialise toutes les données de la classe à 0 ou *null*, si les données sont des objets.

Si un constructeur est défini, le constructeur par défaut n'existe plus.

L'**héritage** permet la réutilisation des objets et de leur comportement, tout en apportant de légères variations Il se traduit par le principe suivant : on dit qu'une classe B hérite d'une classe A (B étant une sous-classe de A) losrqu'il est possible de mettre la relation "est un " entre B et A.

De cette façon, toutes les méthodes, ainsi que les données déclarées `public` ou `protected`, de la classe A sont applicables à la classe B.