Java Définitions

Le **polymorphisme** dans **java** veut simplement dire qu'une classe peut prendre plusieurs formes et c'est d'autant plus vrai avec les classes qui hérite d'une classe supérieure.

* Une méthode **surchargée** (overload en anglais) diffère de la méthode originale par le nombre ou le type des paramètres qu'elle prend en entrée (**même nom MAIS signature différente**).
* Une méthode **polymorphe** ou **redéfinie (réécrite)** (override en anglais) a un squelette identique à la méthode de base, mais traite les choses différemment (**même signature**). Cette méthode se trouve dans une autre classe (classe fille).
* Classe abstraite lorsque on peut pas instancier la classe (Pas de new ), des attributs et méthodes qui peuvent être utilisés par les instances des **classes** qui en héritent.
* Interface = **une interface** est un ensemble de signatures de méthodes publiques d'un objet. Il s'agit donc d'un ensemble de méthodes accessibles depuis l'extérieur d'une classe, par lesquelles on peut modifier un objet, ou plus généralement communiquer avec lui.
* public : classe / variable d’instance / méthode accessible par n’importe qui et n’importe où
* protected : variable d’instance / méthode accessible par les classes du même package ET ses classes filles (héritage) quel que soit leur package
* ***default*** (***package***) : classe / variable d’instance / méthode accessible uniquement par les classes du même package
* private : variable d’instance / méthode accessible par personne (uniquement l’objet lui-même)

En résumé

* Une classe est définie comme abstraite avec le mot clé abstract.
* Les classes abstraites sont à utiliser lorsqu'une classe mère ne doit pas être instanciée.
* Une classe abstraite ne peut donc pas être instanciée.
* Une classe abstraite n'est pas obligée de contenir de méthode abstraite. Si une classe contient une méthode abstraite, cette classe doit alors être déclarée abstraite.
* Une méthode abstraite n'a pas de corps.
* Une interface est une classe 100 % abstraite. Aucune méthode d'une interface n'a de corps.
* Une interface sert à définir un supertype et à utiliser le polymorphisme.
* Une interface s'implémente dans une classe en utilisant le mot clé implements.
* Vous pouvez implémenter autant d'interfaces que vous voulez dans vos classes.
* Vous devez redéfinir toutes les méthodes de l'interface (ou des interfaces) dans votre classe.
* Le pattern strategy vous permet de rendre une hiérarchie de classes plus souple.
* Préférez encapsuler des comportements plutôt que de les mettre d'office dans l'objet concerné.

Enum

* Une énumération est une classe contenant une liste de **sous-objets**.
* Une énumération se construit grâce au mot clé enum.
* Les enum héritent de la classe java.lang.Enum.
* Chaque élément d'une énumération est un objet à part entière.
* Vous pouvez compléter les comportements des objets d'une énumération en ajoutant des méthodes.

Classe enveloppe

des classes qui encapsulent les données de type primitif, afin de pouvoir les utiliser comme des objets ordinaires.