

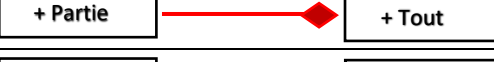
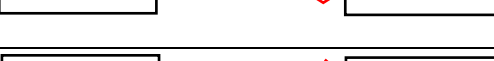
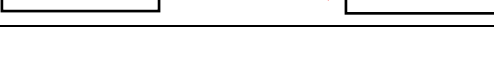


Définition Java POO

Implémentation (interface)		
Héritage		
Compositions		La destruction du tout détruit les parties
Agrégation		La destruction du tout ne détruit pas les parties
Énumération		

COO : Conception Orientée Objet :

- Analyser les différents types d'éléments en présence
- Définir leurs caractéristiques (les données les caractérisant)
- Expliciter l'ensemble des traitements qu'ils peuvent réaliser / subir

Instanciation : nom donné à l'opération de création d'un objet. On utilise l'opérateur **new** pour instancier

Diagramme de classes (UML) : représente les membres (attributs et méthodes) du modèle au sein du module

Nom
(+ / -) Les attributs
(+ / -) Les méthodes

Classe : modèle décrivant les caractéristiques communes et les comportements communs d'un ensemble d'éléments (module décrit selon le langage choisi)

Objet : représentant d'une classe :

- Une classe constitue un générateur d'objet / un modèle d'objet
- Un objet est une instance de cette classe

Les constructeurs :

- **Par défaut** : On ne précise rien (aucun paramètre), les valeurs sont fixées par défaut, selon une convention.
- **Par initialisation** : On précise toutes les valeurs de tous les attributs en paramètres. (On vérifie les valeurs).
- **Par copie** : On passe un objet de même type en paramètre, le nouvel objet aura les mêmes valeurs que celui passé en paramètre.

Membres :

- **Attributs** : données membres
- **Méthodes** : comportement des objets de la classe

Encapsulation : Les données et les méthodes sont rassemblées au sein de l'objet

Interface [partie publique] :

En Java : Attributs publics et signature des méthodes publiques.

En POO : Ensemble de méthodes accessibles depuis l'extérieur de la classe. Le monde extérieur n'a accès aux données que par l'intermédiaire de cet ensemble de méthodes.

Implémentation [partie privée] :

C'est le détail non visible de la représentation des données et du codage des méthodes

Enumérations :

- Type de données particulières
- Ne peut prendre qu'un nombre restreint de valeurs
- Ces valeurs sont des constantes nommées

Exceptions :

Chargée de signaler un comportement exceptionnel (mais prévu !) d'une partie spécifique du code.

Souvent intégré aux langages de programmation actuels

- **Vérifiée** : hérite de l'une des deux classes Error ou RuntimeException **throws**

- **Non vérifiée** : toutes les autres exceptions **throw**

Public : accessible par toutes les classes

Protected : accessibles par toutes les classes héritées et les classes du même paquetage, inaccessibles par les autres

Private : inaccessible par toute autre classes.

Static : membre de classe (souligné dans diagramme UML)

Non-Static : membre d'instance

Final : non modifiable

Non-Final : modifiable

Surcharge : permet de définir plusieurs fois une même méthode/constructeur avec des arguments différents.

Agrégation : permet de définir qu'un objet est l'assemblage d'un ou de plusieurs sous-objets. (Une entreprise peut exister sans employer car si l'entreprise n'existe plus l'employé lui est encore là)

Composition : est un cas particulier de l'agrégation qui implique une dépendance plus forte de l'objet agrégé dans l'objet agrégeant. (Une salle ne peut pas exister sans bâtiment en effet si le bâtiment est détruit la salle est détruite avec)

Copie de surface : recopie de la référence (objet partagé). En général on l'utilise avec une **agrégation**.

Copie profonde : clonage de l'objet (avec le constructeur par copie de l'objet utilisé). En général on l'utilise avec une **composition**.

Héritage : Evite la réécriture : factorisation + spécialisation

Classe base (mère) : Plus générique => Partie commune

Classe dérivée (fille) :

Plus spécifique(s) => Hérite(nt) de la partie commune + traitements spécifiques + attributs propres

Constructeur d'une classe fille : Il faut appeler un constructeur de la classe mère avec l'instruction « **super ()** ». Si on ne la déclare pas elle est mise de base par java. Attention la classe mère doit contenir un constructeur par défaut.

Typage statique : type de l'objet = type de la référence (variable)

Typage dynamique : type de l'objet plus spécifique que celui de la référence. Imaginons qu'il y a une classe capitale qui hérite de la classe ville alors

On peut : `Ville v = new Capitale () ;`

Mais PAS ! : `Capitale c = new Ville () ;`

Classe abstraite :

- ne peut pas être instanciée
- Contient des attributs et méthode concret

Introspection : besoin de connaître le type d'un objet avec le mot « **instanceof** » cela permet de déterminer si un objet est d'une classe donnée

Classe object :

- classe fondamentale
- Toutes les classes java en hérite

toString() : permet d'afficher ou non la description d'un objet

clone() : copie de bas niveau efficace / duplique un objet en dupliquant la zone mémoire

/ ! \ Le clonage par défaut est interdit il faut le rendre public et le redéfinir

equals() : compare 2 objets => égalité sémantique

Les collections :

- Structure les données efficacement
- Permet de gérer un ensemble d'objets (tri, stocker, manipuler ...)

⇒ Car jusqu'à-là on utilisait les tableaux mais manque de fonctionnalités (taille fixe ...)

Classe paramétrée (type générique) : Spécialise les classes qui gèrent les collections en fonction du type des objets qu'elles vont manipuler

Classe Vector<E> : gestion d'une collection d'objets E dans un tableau dynamique (la taille varie)