**POO**

**Argumentaire sur l’héritage et la composition**

**Héritage :**

L’héritage est une technique très puissante et extrêmement pratique, son principe est simple, une **classe B** hérite d'une **classe A**. La classe A est donc considérée comme la **classe mère** et la classe B est considérée comme la **classe fille**.

La **classe fille** hérite de tous les attributs et méthodes de la **classe mère** de type public. Les types protected et private restreignent leur accès.

L'héritage est une relation *unidirectionnelle,* Mère vers la fille.



**Avantage :**

- Facilité pour créer des classes d'enfants avec la possibilité d’abstraire des classes grâce à la méthode « abstract » afin d’améliorer la logique du code.

Création de hiérarchies de classes reflétant précisément le domaine d'étude.

- Grâce à la relation d'héritage, une classe fille peut utiliser une méthode de la classe mère sans avoir à la redéfinir. C’est un gain de temps de développement.

Ex : un objet de la classe CompteEpargne peut utiliser les fonctionnalités de la classe CompteBancaire sans avoir à les redéfinir.

**Composition :**

Une composition c’est une combinaison d’objet simple, transparents et indépendants en un tout complexe. La composition sera plus flexible et ne sera pas liée directement à la classe qui l’accueille contrairement à l’héritage.



Un autre exemple concret : un mariage est une composition de 2 individus.

Représentation UML d’une composition.

****

**Avantage :**

* Une indépendance structurelle.
* La composition ouvre la possibilité de modifier le comportement des objets dynamiquement ce qui offre une grande flexibilité.
* La composition offre un gain de temps considérable lors de l’ajout de nouvelle fonctionnalité.

**Cas d’utilisation**

Pour savoir à quel moment utiliser l’héritage ou la composition il faut définir les relations entre les différentes classes.

Si la classe B a toutes les caractéristiques de la classe A, mais qu'elle est susceptible, durant son cycle de vie, de modifier la classe A, il faut s’orienter vers une composition...  
  
Si, par contre, les caractéristiques de A sont définies une fois pour toutes dans B, sans modification, l’héritage sera la solution.

L’héritage multiple n’est pas supporté par tous les langages de programmation.

**Conclusion :**

**L'héritage correspond au verbe "être" et la composition au verbe "avoir".**

L’héritage c’est lorsque que vous concevez vos classes en fonction de ce qu'ils sont.

La composition lorsque vous concevez des types en fonction de ce qu'ils font.

Si **l’Héritage** et **la Composition** ont le même objectif d’aider à réutiliser du code, leur choix soulève différents défis. La **composition** offre un meilleur moyen de réutiliser le code

**Composition – a-une** relation entre **les objets**.  
**Héritage – est-une** relation entre **les classes**.

**Composition** – L’objet de composition contient une référence aux classes de composition et la relation est donc vaguement liée.  
**Héritage** – L’objet de la classe fille porte la définition de la classe mère en soi et donc étroitement liée.

**Composition** – Utilisé en injection de dépendance  
**Héritage** – Utilisé dans le polymorphisme

**Composition** – Les objets peuvent être composés au sein de plusieurs classes.  
**Héritage** – Une classe ne peut hériter que d’une seule classe.