Héritage vs Composition

Héritage

L'héritage est un principe propre à la programmation orientée objet, permettant de créer une nouvelle classe à partir d'une classe existante. Le nom d'"héritage" provient du fait que la classe enfant contient les attributs et les méthodes de la classe parent. L'intérêt majeur de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe enfant, qui viennent s'ajouter à ceux et celles héritées.

Composition

La composition est un cas particulier de l'agrégation, qui implique une inclusion plus forte de l'objet enfant dans l'objet parent. Alors qu'un même objet peut être agrégé par plusieurs objets différents, un même objet ne peut pas être « composé » par plusieurs objets. Il y a de plus une notion de durée de vie commune entre un composant et ses composés : les composés d'un composant sont en général détruits en même temps que le composant.

Comparaison

L'héritage et la composition permettent la réutilisation du code, la principale différence entre Composition et Héritage réside dans le fait que Composition permet la réutilisation du code sans hériter la classe mère, mais pour l'héritage vous devez hériter la classe mère pour toute réutilisation de code ou de fonctionnalité. Une autre différence qui découle de ce fait est que, en utilisant la Composition, vous pouvez réutiliser du code pour une classe finale, ce qui n'est pas extensible, mais Héritage ne peut pas réutiliser le code dans de tels cas. Également en utilisant Composition, vous pouvez réutiliser le code de nombreuses classes car elles sont déclarées comme une simple variable membre, mais avec Héritage, vous pouvez réutiliser le code d'une seule classe car en Java ou en PHP ou dans quelque d'autre langage de programmation, vous ne pouvez hériter qu'une seule classe. Vous pouvez le faire en C ++ car une classe peut hériter plusieurs classes. Ainsi, vous devriez toujours préférer la composition à l'héritage.

Nous pouvons voir que même si l'Héritage et la Composition ont le même objectif d'aider à réutiliser un objectif éprouvé, leur choix soulève différents défis. La composition offre un meilleur moyen de réutiliser le code et protège en même temps la classe que vous réutilisez de n'importe quel client, mais Héritage n'offre pas cette garantie. Parfois l'héritage est nécessaire, principalement quand vous créez une classe de la même famille.