DESIGN PATTERN

OBSERVATEUR : c’est un patron de conception de la famille des patrons comportementaux, on l’utilise pour limiter le couplage entre les modules aux seuls phénomènes à observer. Aussi, il permet une gestion simplifiée d’observateur multiples sur un même observable. Ces derniers effectuent l’action adéquate en fonction des informations qui parviennent depuis des modules qu’ils observent.

SINGLETON : il permet de restreindre l’instanciation d’une classe à un seul objet. Il est utilisé lorsqu’on a besoin exactement d’un objet pour coordonner les opérations dans un système. On implémente le singleton en écrivant une classe contenant une méthode qui crée une instance uniquement s'il n'en existe pas encore. Sinon elle renvoie une référence vers l'objet qui existe déjà. Dans beaucoup de langages de type objet, il faudra veiller à ce que le constructeur de la classe soit **privé**, afin de s'assurer que la classe ne puisse être instanciée autrement que par la méthode de création contrôlée.

STRATEGY : c’est un patron de conception de type comportementale grâce auquel des algorithmes peuvent être sélectionnés à la volée au cours du temps d'exécution selon certaines conditions. Il est utile pour des situations où il est nécessaire de permuter dynamiquement les algorithmes utilisés dans une application. Le patron stratégie est prévu pour fournir le moyen de définir une famille d'algorithmes, encapsuler chacun d'eux en tant qu'objet, et les rendre interchangeables. Il laisse les algorithmes changer indépendamment des clients qui les emploient.

BUILDER : Il est utilisé pour la création d'une variété d'objets complexes à partir d'un objet source. L'objet source peut consister en une variété de parties contribuant individuellement à la création de chaque objet complet grâce à un ensemble d'appels à l'interface commune de la classe abstraite monteur. Un exemple d'objet source est une liste de caractères ou d'images dans un message devant être codé. Un objet directeur est nécessaire pour fournir les informations à propos de l'objet source vers la classe Monteur.

FACTORY : Elle permet d'instancier des objets dont le type est dérivé d'un type abstrait. La classe exacte de l'objet n'est donc pas connue par l'appelant. Plusieurs fabriques peuvent être regroupées en une fabrique abstraite permettant d'instancier des objets dérivant de plusieurs types abstraits différents. Les fabriques étant en général uniques dans un programme, on utilise souvent le patron de conception singleton pour les implémenter.

DECORATOR : c’est le nom d'une des structures de patron de conception. Un décorateur permet d'attacher dynamiquement de nouvelles responsabilités à un objet. Ils offrent une alternative assez souple à l'héritage pour composer de nouvelles fonctionnalités.