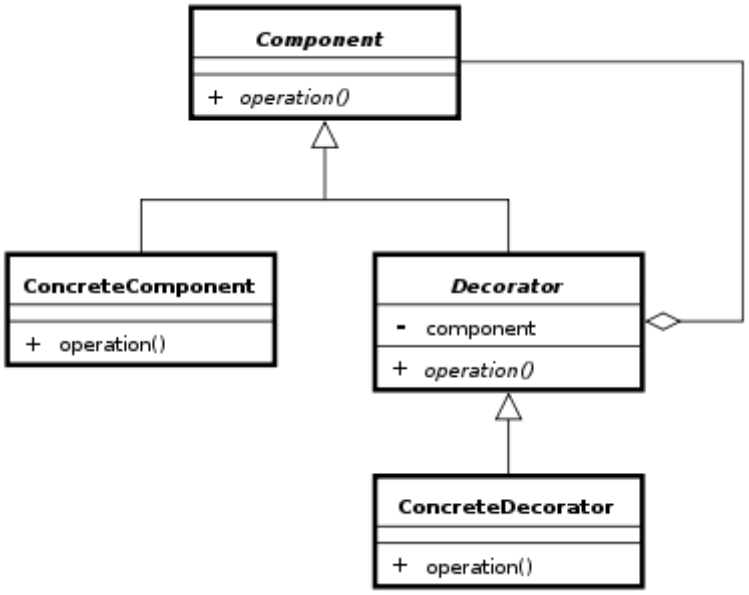
GROUPE 6

DESIGN PATTERN

❖ DECORATOR PATTERN :

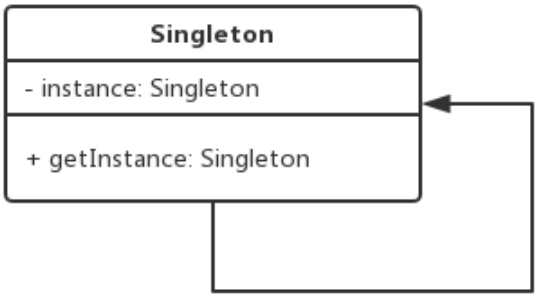
C’est le nom d'une des structures de patron de conception. Un décorateur permet d'attacher dynamiquement de nouvelles responsabilités à un objet. Ils offrent une alternative assez souple à l'héritage pour composer de nouvelles fonctionnalités.



❖ SINGLETON PATTERN :

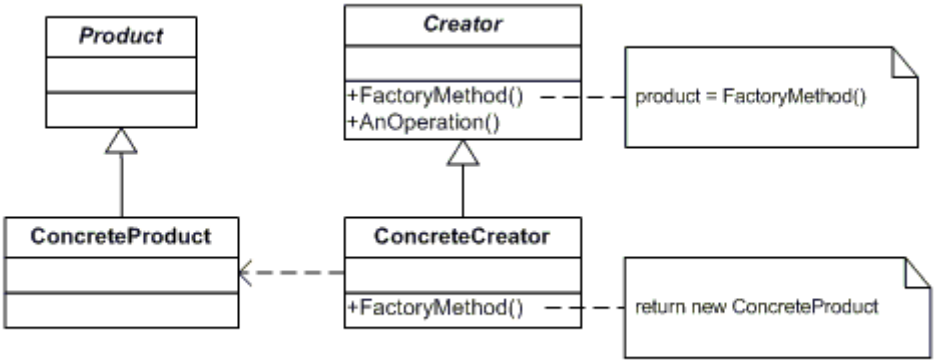
C’est un patron de conception de type comportementale grâce auquel des algorithmes peuvent être sélectionnés à la volée au cours du temps d'exécution selon certaines conditions. Il est utile pour des situations où il est nécessaire de permuter dynamiquement les algorithmes utilisés dans une application. 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.

GROUPE 6



❖ **FACTORY PATTERN :**

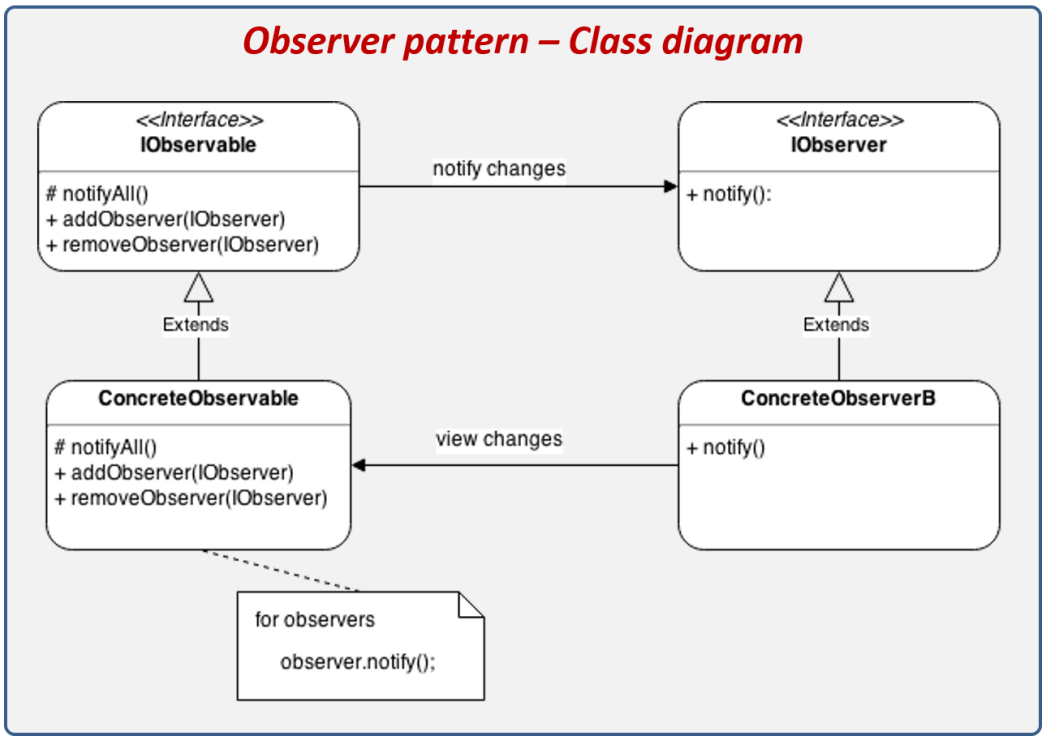
Elle permet d'instancier des objets dont le type est dérivé d'un type abstrait. La classe exacte de l'objet n'est donc pas connue par l'appelant. Plusieurs fabriques peuvent être regroupées en une fabrique abstraite permettant d'instancier des objets dérivant de plusieurs types abstraits différents.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



❖ **OBSERVER PATTERN :**

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.

GROUPE 6

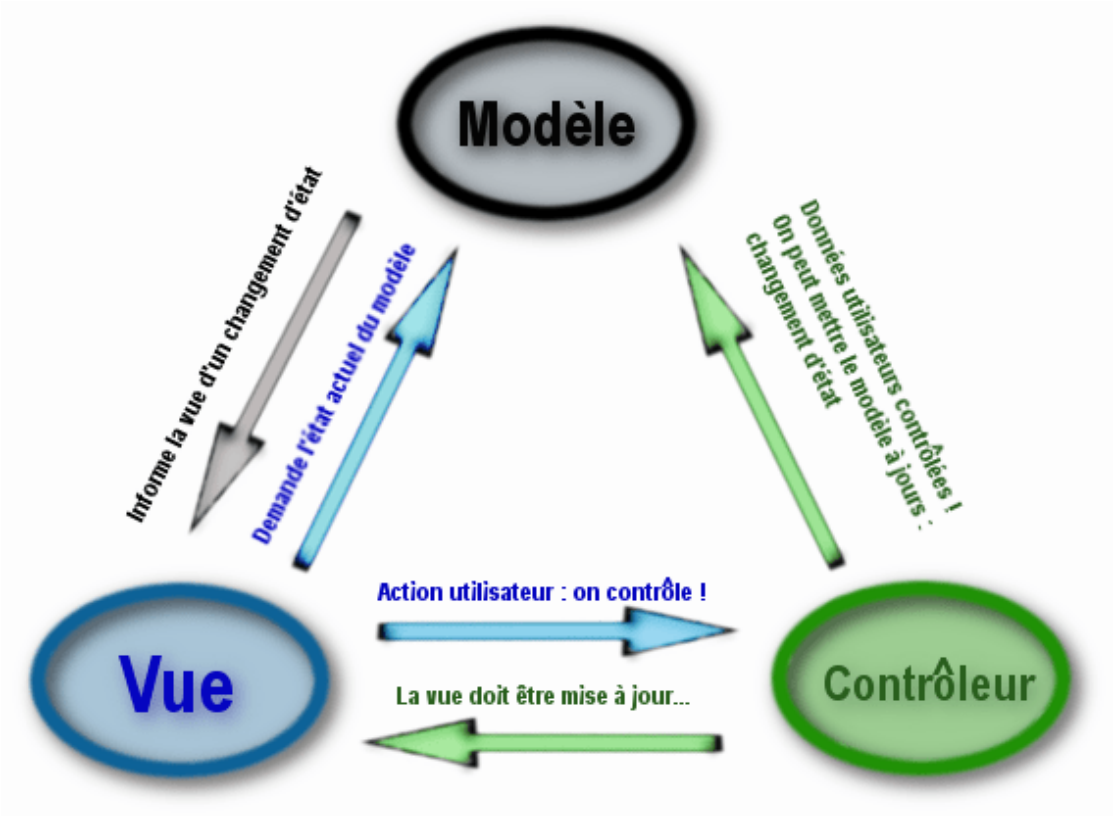


❖ **MVC PATTERN :**

Le pattern MVC permet de bien organiser son code source. Il va nous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

- **Modèle** : cette partie gère les *données* du site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.
- **Vue** : cette partie se concentre sur l'*affichage*. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
- **Contrôleur** : cette partie gère la logique du code qui prend des *décisions*. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

GROUPE 6



❖ **STRATEGY PATTERN :**

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.