

"Design Pattern"
Catalogue de modèles de conception réutilisables
Erich Gamma - International Thomson Publishing

Nous connaissons tous le poids, la valeur, l'importance de l'expérience (en matière de conception objet ou autre). Combien de fois avons-nous eu cette sensation de déjà vu ? Cette impression d'avoir résolu ce problème, mais sans se rappeler quand et comment et surtout comment réécrire la solution ?

L'objectif des "Design Patterns" est de tenter une expertise, en matière de conception orientée objets, sous la forme de 'Modèles de Conception'. Chaque modèle de conception nomme, explique et évalue un concept important qui figure fréquemment dans les systèmes orientés objets.

1 – Organisation du catalogue

Le Rôle traduit ce que fait le modèle :

- Créateur : concernent le processus de création d'objets,
- Structurel : s'occupent de la composition de classes ou d'objets,
- Comportemental : spécifie les façons d'interagir de classes et d'objets et de se répartir les responsabilités.

Le Domaine précise le modèle s'applique aux :

- classes : traitent de relations statiques, définies par héritage et donc lors de la compilation,
- objets : traitent de relations dynamiques, définies au moment de l'exécution.

		Rôle		
		Créateur	Structurel	Comportement
Domaine	Classe	Fabrication	Adaptateur (classe)	Interprète Patron de méthode
	Objet	Fabrique abstraite Monteur Prototype Singleton	Adpatateur (objet) Pont Composite Décorateur Façade Poids mouche Procuration	Chaîne de responsabilités Commande Itérateur Médiateur Memento Observateur Etat Stratégie Visiteur

Il est à noter que la plupart des modèles sont du domaine Objet

Rôle	Modèle de conception	Elément(s) modifiable(s)
Créateur	Fabrique abstraite	Les objets "familles de produits"
	Monteur	La façon de fabriquer un objet composite
	Fabrication	Une sous-classe instanciable d'objet
	Prototype	Une classe instanciable d'objet
	Singleton	L'instance unique d'une classe
Structurel	Adaptateur	Une super-interface d'objet
	Pont	L'implémentation d'un objet
	Composite	La structure et la composition d'un objet
	Décorateur	Les responsabilités d'un objet sans avoir à recourir à la dérivation de classes

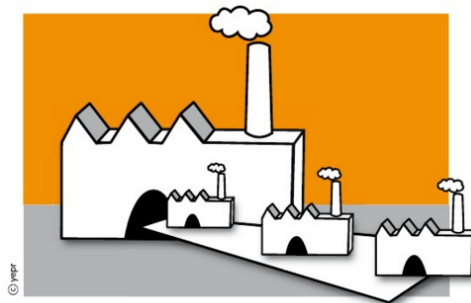
Rôle	Modèle de conception	Élément(s) modifiable(s)
Comporte- mental	Poids mouche	Le coût de stockage des objets
	Procuration	La forme d'accéder à un objet ; son emplacement
	Chaîne de responsabilité	L'objet capable de satisfaire une requête
	Commande	L'occasion et la manière de satisfaire une requête
	Interprète	La grammaire et l'interprétation d'un langage
	Médiateur	La nature des objets en interaction et les modalités de cette interaction
	Memento	Quelle information privée est extraite d'un objet ; à quelles occasions
	Observateur	Le nombre des objets dépendants d'un autre objet ; les moyens de tenue à jour des objets dépendants
	Etat	Les états d'un objet
	Stratégie	Un algorithme
	Méthode du patron	Des étapes d'un algorithme
	Visiteur	Les opérations qui s'appliquent aux objets sans avoir à modifier leurs classes

Éléments de conception à variabilité facilitée par les modèles de conception.

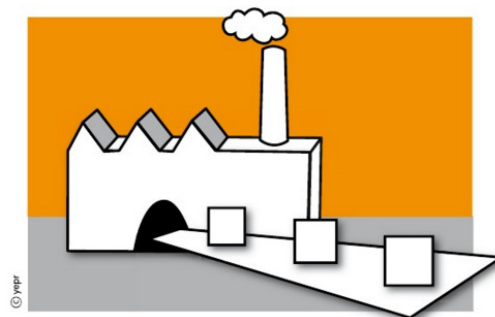
2 – Le catalogue des modèles de conception

CREATEURS

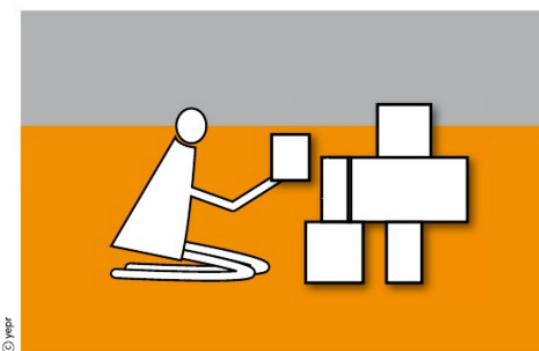
Fabrique Abstraite (Abstract Factory) : Fournit une interface pour créer des familles d'objets apparentés ou dépendants, sans avoir à spécifier leurs classes concrètes.



Fabrication (Factory method) : Définit une interface pour la création d'un objet, tout en laissant à des sous-classes le choix de la classe à instancier. Une fabrication permet de déléguer à des sous-classes les instanciations d'une classe.



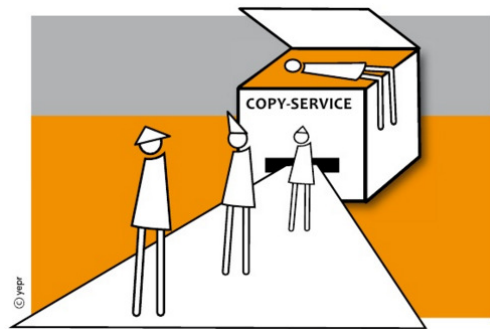
Monteur (Builder) : Dans un objet complexe, dissocie sa construction de sa représentation, de sorte que, le même procédé de construction puisse engendrer des représentations différentes.



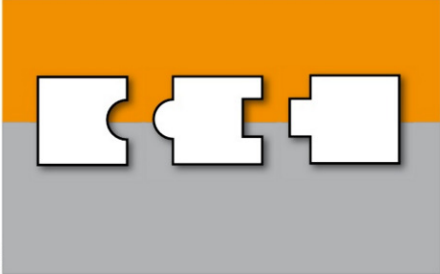
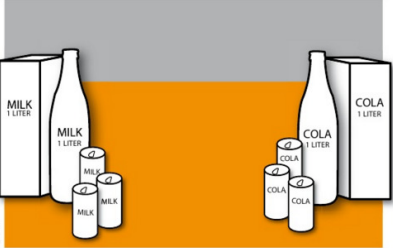
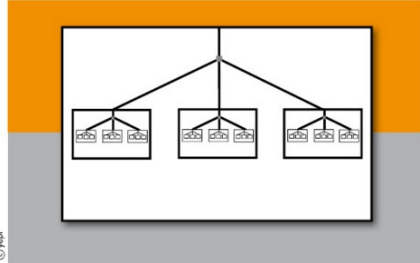

Singleton (Singleton) : Garantit qu'une classe n'a qu'une seule instance, et fournit à celle-ci un point d'accès de type global.



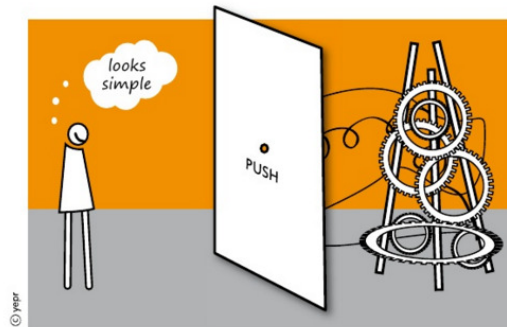
Prototype (Prototype) : Spécifie les espèces d'objets à créer, en utilisant une instance de type prototype, et crée de nouveaux objets par copies de ce prototype.



STRUCTUREL

<p>Adaptateur (Adapter) : Convertit l'interface d'une classe en une interface distincte, conforme à l'attente de l'utilisateur. L'adaptateur permet à des classes de travailler ensemble, qui n'auraient pu le faire autrement pour causes d'interfaces incompatibles.</p>	
<p>Pont (Bridge) : Découpe une abstraction de son implémentation associée, afin que les deux puissent être modifiés indépendamment.</p>	
<p>Composite (Composite) : Organise les objets en structure arborescente représentant la hiérarchie de bas en haut. Le composite permet aux utilisateurs de traiter des objets individuels, et des ensembles organisés de ces objets de la même façon.</p>	
<p>Décorateur (Decorator) : Attache des responsabilités supplémentaires à un objet de façon dynamique. Il permet une solution alternative pratique pour l'extension des fonctionnalités, à celle de dérivation de classe.</p>	

Façade (Facade) : Fournit une interface unifiée pour un ensemble d'interfaces d'un sous-système. Façade définit une interface de plus haut niveau, qui rend le sous-système plus facile à utiliser.



Poids mouche (Flyweight) : Assure en mode partagé le support efficace d'un grand nombre d'objets à fine granularité.



Procuration (Proxy) : Fournit un remplaçant d'un objet pour en contrôler l'accès.

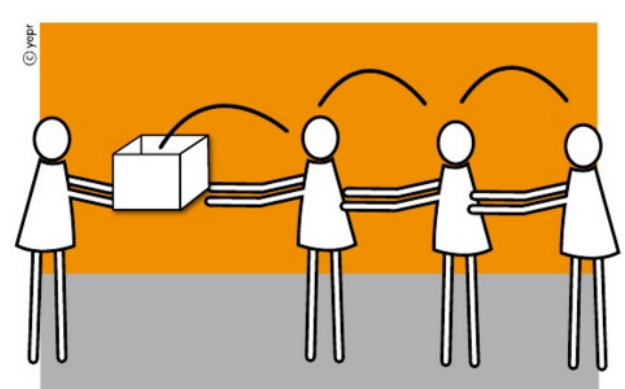


COMPORTEMENTAL

Commande (Command) : Encapsule une requête comme un objet, ce qui permet de faire un paramétrage des clients avec différentes requêtes, files d'attente, ou historiques de requêtes, et d'assurer le traitement des opérations réversibles.

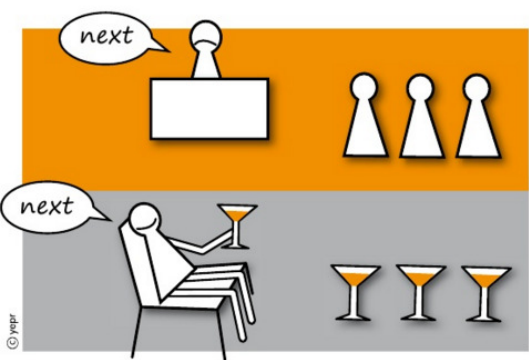
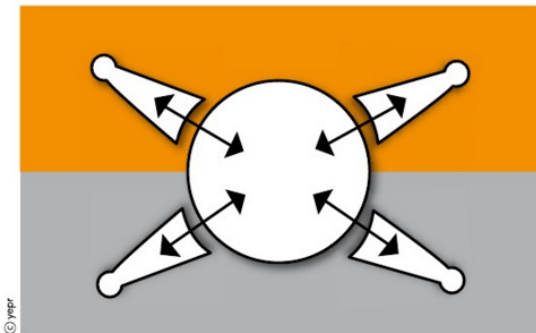
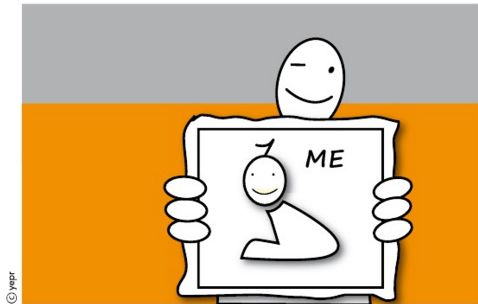
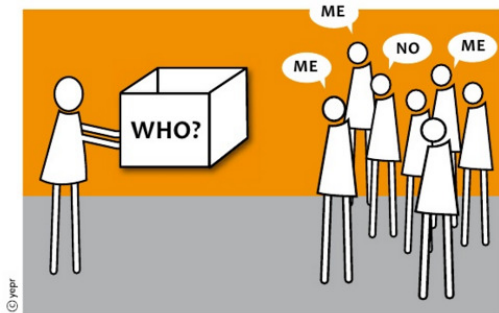




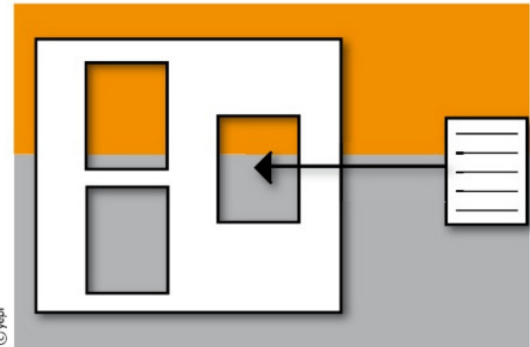

Chaîne de responsabilités (Chain of responsibility) : Permet d'éviter de coupler l'expéditeur d'une requête à son destinataire, en donnant la possibilité à plusieurs objets de prendre en charge la requête. Pour ce faire, il chaîne les objets récepteurs, et fait passer la requête tout au long de cette chaîne jusqu'à que l'un de ces objets la prenne en charge.



Interprète (Interpreter) : Dans un langage donné, il définit une représentation de sa grammaire, ainsi qu'un interprète qui utilise cette représentation pour analyser la syntaxe du langage.



<p>Itérateur (Iterator) : Fournit un moyen pour accéder en séquence aux éléments d'un objet de type agrégat sans révéler sa représentation sous-jacente.</p>	
<p>Médiateur (Mediator) : Définit un objet qui encapsule les modalités d'interaction de divers objets. Le médiateur favorise les couplages faibles, en dispensant les objets d'avoir à faire référence explicite les uns aux autres ; de plus, il permet de modifier une relation indépendamment des autres.</p>	
<p>Memento (Memento) : Sans violer l'encapsulation, acquiert et délivre à l'extérieur une information sur l'état interne d'un objet, afin que celui-ci puisse être rétabli ultérieurement dans cet état.</p>	
<p>Observateur (Observer) : Définit une corrélation du type un à plusieurs, entre objets, de façon que lorsqu'un objet change d'état, tous ceux qui en dépendent en soient notifiés et mis à jour automatiquement.</p>	

<p>Etat (State) : Permet à un objet de modifier son comportement lorsque son état interne change. L'objet paraîtra changer de classe.</p>	
<p>Stratégie (Strategy) : Définit une famille d'algorithmes, encapsule chacun d'entre eux et les rend interchangeables. Une stratégie permet de modifier un algorithme indépendamment de ses clients.</p>	
<p>Patron de méthode (Template method) : Définit le squelette de l'algorithme d'une opération, en déléguant le traitement de certaines étapes à des sous-classes. Le patron de méthode permet aux sous-classes de redéfinir certaines étapes d'un algorithme sans modifier la structure de l'algorithme.</p>	
<p>Visiteur (Visitor) : Représente une opération à effectuer sur les éléments d'une structure d'objet. Le visiteur permet de définir une nouvelle opération sans modifier les classes des éléments sur lesquels il opère.</p>	

<http://fr.slideshare.net/HermanPeeren/design-patterns-illustrated>