Janssen
Ould Slimane
Troadec
Tedongap Tsapi

Haute Ecole de la Province de Liège

## Cas n°1

Vous développez dans le cadre de vos activités une application utilisée pour la vente de PCs personnalisés. Les ordinateurs qui seront vendus peuvent être configurés au niveau de la quantité de mémoire (nombre de « barrettes »), un CPU et des cartes additionnelles (graphique, réseau, …). Les cartes additionnelles possèdent elles -mêmes, en plus de ports d'entrées-sorties, de la mémoire et un CPU. Dans la solution, il doit être possible de calculer le prix de l'ordinateur ainsi configuré en additionnant les prix de chaque élément qui le constitue.

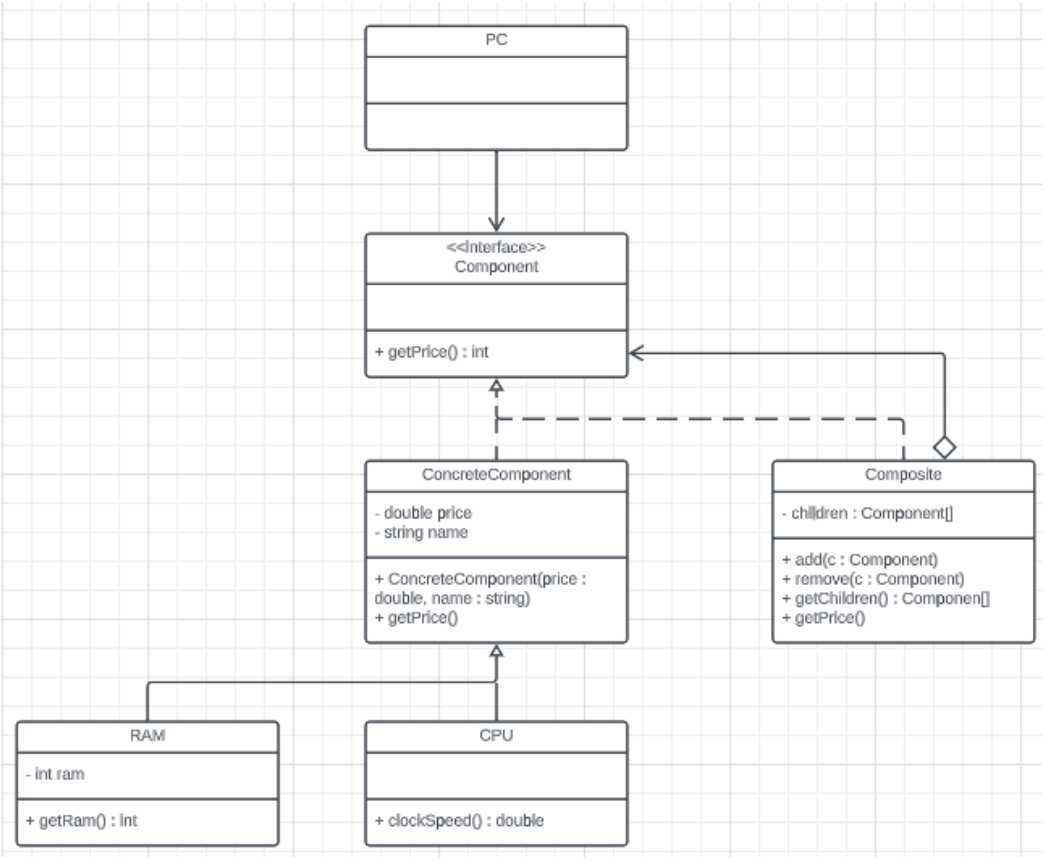
Quel patron de conception pourriez-vous mettre en œuvre pour représenter ces abstractions?
Écrire le code de la solution dont celui permettant d'effectuer le calcul du prix.

## Réponse :

On a décidé d'utiliser le patron "Composite" car on retrouve sur l'énoncé une information nous disant que le PC peut être configuré en RAM, en CPU et en cartes additionnelles qui sont eux mêmes composés de RAM et de CPU. On retrouve donc un arbre de composants similaires ce qui justifie l'utilisation du patron de conception "Composite".

De plus, cette solution nous permet aussi de facilement calculer le prix du PC en additionnant simplement les éléments feuilles de l'arbre.

On pourrait aussi le combiner avec un Builder pour pouvoir gérer les différents paramètres des composants du PC si on essaye de complexifier notre arbre.



Janssen
Ould Slimane
Troadec
Tedongap Tsapi

HEPL
Haute Ecole de la Province de Liège

## Cas n°2

Vous disposez d'une interface Component qui représente les composants d'un ordinateur. Les classes spécifiques qui implémentent cette interface pour représenter les différents composants sont connues. Cependant, les fonctionnalités ou opérations que vous souhaitez effectuer sur ces classes ne sont pas connues à l'avance et peuvent évoluer au fil du temps.

Plutôt que d'ajouter ces fonctionnalités dans l'interface Component, que pourriez-vous faire? Si les fonctionnalités sont déclarées dans Component, quels principes ne sont pas respectés ?

## Réponse :

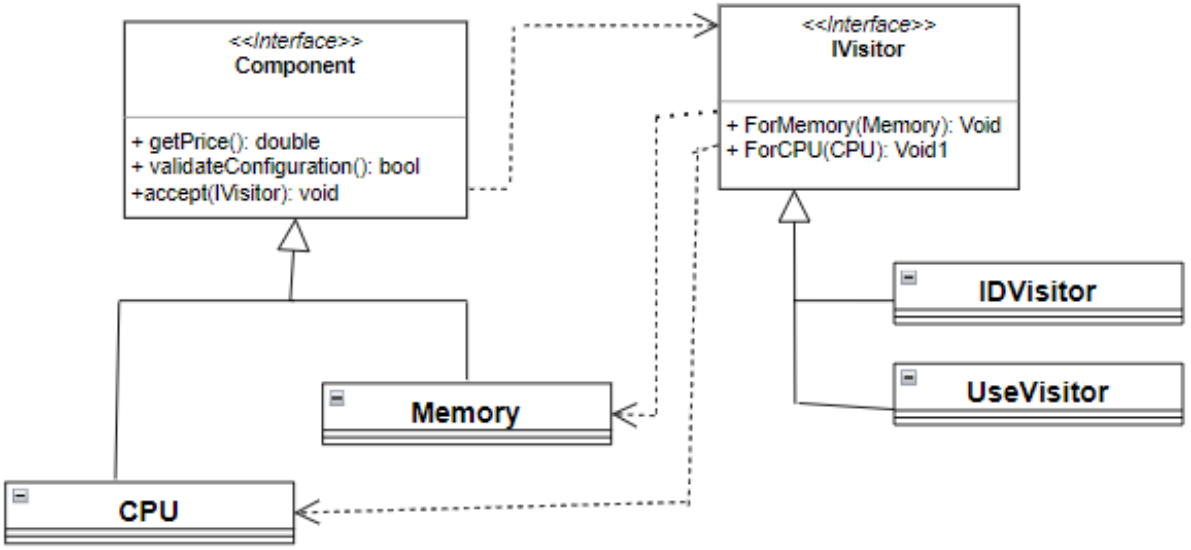
Initialement, le pattern "Command" semblait être une option pertinente. Cependant, en l'utilisant, nous aurions risqué de surcharger "component" avec des fonctionnalités supplémentaires, compliquant le code et introduisant des comportements potentiellement inattendus. En revanche, le pattern "Visitor" offre une flexibilité distincte, nous permettant d'introduire des opérations aux classes (dans ce contexte, des fonctionnalités) sans nécessiter de modifications larges de celles-ci.
Cependant, l'ajout d'un nombre croissant de visiteur soulèverai quelques préoccupations :

- Complexité accrue : chaque fois qu'une nouvelle classe de visiteur est introduite, on devra nous assurer qu'elle interagit correctement avec chaque composant. Si le nombre de composants est élevé, cela signifie un grand nombre de combinaisons possibles, ce qui augmente la complexité du système.
- Couplage accru : même si l'un des objectifs du pattern "Visiteur" est de réduire le couplage, l'ajout d'un grand nombre de visiteurs peut en réalité augmenter le couplage entre les composants et les visiteurs. Cela pourrait rendre le système plus rigide à long terme.

Design pattern : Visiteur – Jonathan SAEZ (kandran.fr)
Si les fonctionnalités sont déclarées dans Component, quels principes ne sont pas respectés ?
Violation du Principe de Responsabilité Unique (SRP) : Chaque classe doit avoir une seule raison de changer. Si nous ajoutons différentes fonctionnalités dans la même classe ou interface, nous introduisons plusieurs raisons pour que cette classe ou interface change, violant ainsi le SRP. SOLID : les 5 premiers principes de conception orientée objet | DigitalOcean
Violation du Principe Ouvert/Fermé (OCP) : Les entités logicielles (classes, modules, fonctions, etc.) doivent être ouvertes à l'extension, mais fermées à la modification. Si nous modifions continuellement l'interface Component pour y ajouter de nouvelles fonctionnalités, nous ne respectons pas ce principe. Qu'est-ce que le principe ouvert-fermé (OCP) ? - Cybermédiane (cybermedian.com)

Janssen
Ould Slimane
Troadec
Tedongap Tsapi

HEPL
Haute Ecole de la Province de Liège



Janssen
Ould Slimane
Troadec
Tedongap Tsapi

HEPL
Haute Ecole de la Province de Liège

## Cas n°3

Dans le cadre du développement d'une application de gestion, vous êtes amené à programmer l'évaluation d'expressions arithmétiques. Vous disposez déjà des classes suivantes comme point de départ. Par la suite, d'autres traitements pourront être effectués sur des objets de ces classes. A titre d'exemple, une expression pourrait être représentée sous forme textuelle et être sérialisée dans une chaîne de caractère.

Écrire le code permettant d'évaluer une expression arithmétique quelconque (celle-ci peut être composée de plusieurs opérations). Afin d'illustrer les appels de méthodes, représentez la solution à l'aide d'un diagramme de séquences

## Réponse :

Ici, on a pour but de résoudre une opération arithmétique donnée par une chaîne de caractères (on a utilisé ici l'exemple "11+6-5"). On a donc séparé nos différentes informations en Expression notamment "Plus", "Moins" et "Nombre".

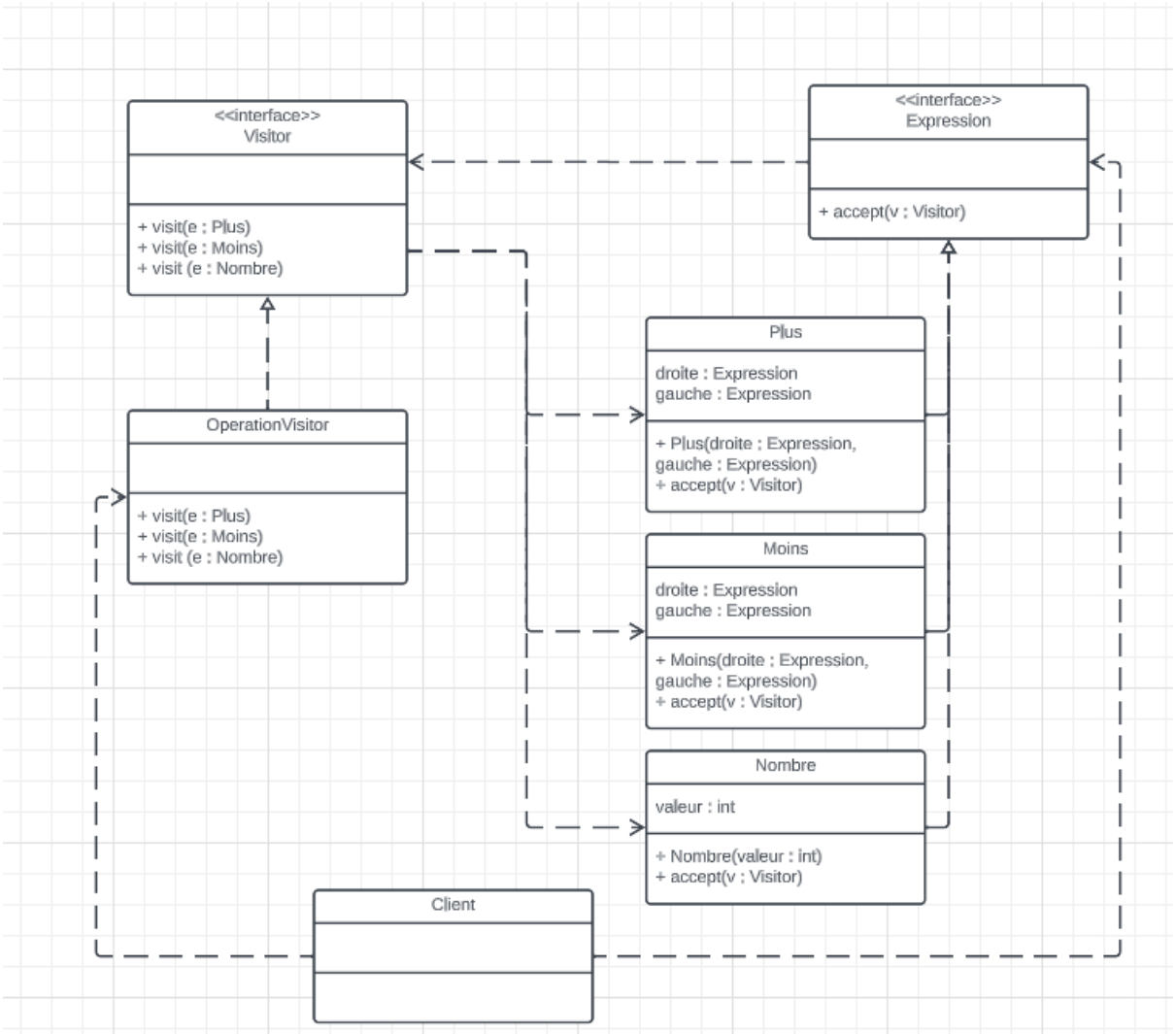
L'objectif de ce cas est de trouver un paterne nous permettant d'implémenter différentes expressions sans pour autant avoir à modifier toutes les expressions existantes. Dans le cas où pour l'expression Moins, si on ajoute une expression Mul, la classe Moins doit savoir gérer cette nouvelle expression, il faudra donc ajouter à la classe une méthode pour le gérer et cela pour chaque expression existante.

Cette manœuvre est facilement redondante et peut être évitée grâce au paterne visiteur ! En effet, "Visiteur" nous permet d'avoir une classe regroupant toutes les gestions des expressions par la méthode visit() existant pour chaque type d'expression. Chaque expression possède alors une méthode accept() qui permet de gérer les différentes expressions du visiteur.

Grâce à cette méthode nous avons facilement fait un code pour réaliser notre exemple. Vous pouvez voir ci dessous le fonctionnement grâce au diagramme de séquence.

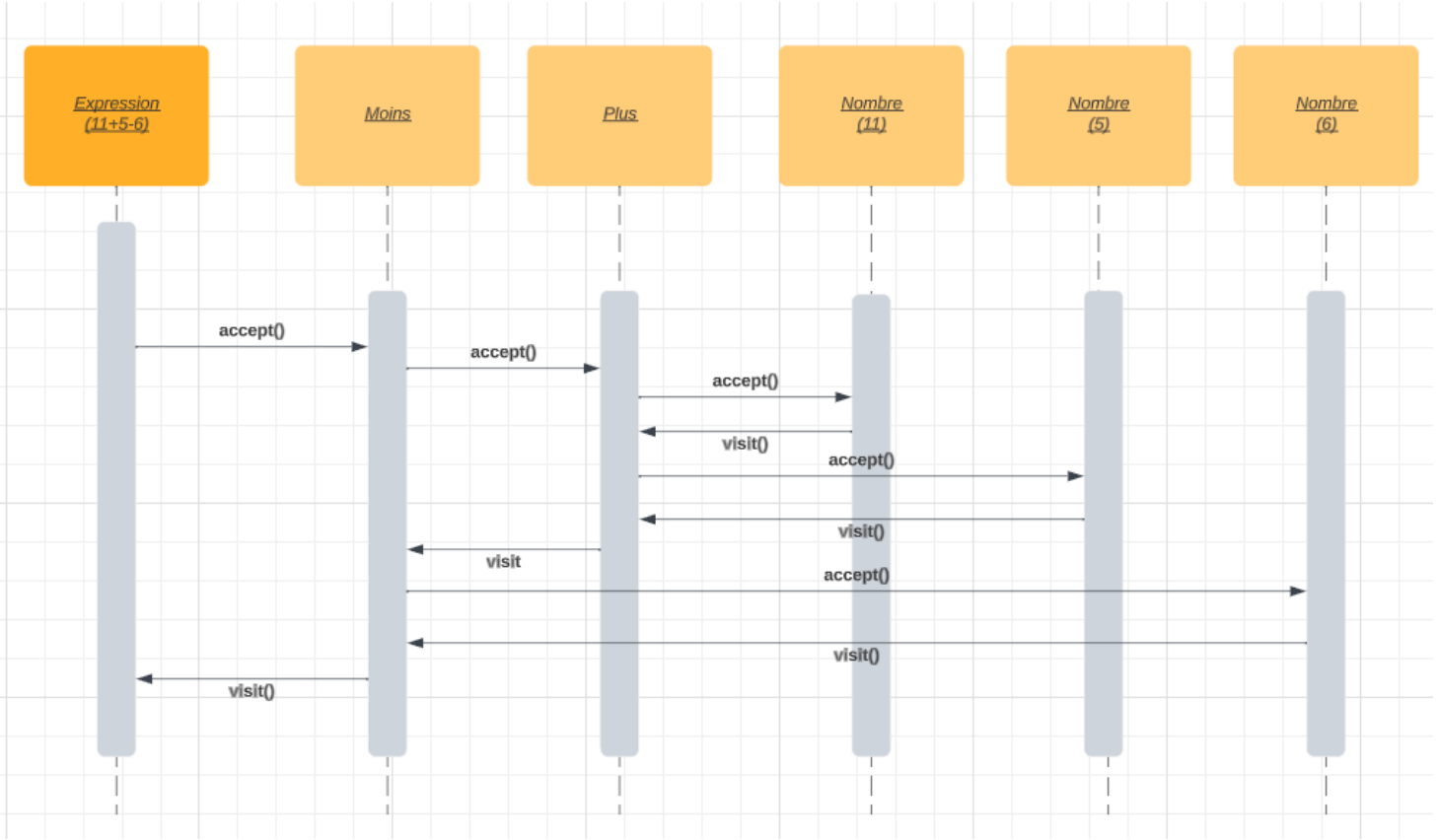
Janssen
Ould Slimane
Troadec
Tedongap Tsapi

HEPL
Haute Ecole de la Province de Liège



Janssen
Ould Slimane
Troadec
Tedongap Tsapi

HEPL
Haute Ecole de la Province de Liège



Janssen
Ould Slimane
Troadec
Tedongap Tsapi

HEPL
Haute Ecole de la Province de Liège

## Cas n°4

Lorsque vous surfez sur internet, vous consultez des pages qui contiennent des images. Charger l'entièreté des images d'une page au début de sa consultation peut nécessiter un certain temps.

Vous désirez trouver une solution à ce problème et améliorer l'expérience utilisateur, d'autant plus que toutes les images n'ont pas besoin d'être affichée dès le début. Quel patron de conception GoF pourrait vous aider à solutionner le problème ? Ecrivez un code HTML permettant d'illustrer la problématique

## Réponse :

Le problème décrit est celui du chargement inutile d'images qui peut entraîner des temps de chargement longs, même si toutes les images n'ont pas besoin d'être affichées dès le début. Donc il faudrait charger les images uniquement lorsqu'elles sont nécessaires ou demandées, ce qui peut améliorer les temps de chargement et l'expérience utilisateur. Autrement dit privilégier le fonctionnement de certains objets afin que leur exécution se termine en premier, et exécuter les autres objets lorsqu'ils sont demandés.
Analogie avec la vie courante : grande bibliothèque de livres
Dès mon entrée dans la bibliothèque, un employé court dans toutes les allées et ramasse chaque livre de la bibliothèque, pensant que je pourrais vouloir le lire. Il m'apporte ensuite une énorme pile de livres. C'est évidemment inefficace et épuisant pour l'employé. De plus, je n'ai besoin que d'un ou deux livres spécifiques, et non de la pile entière.

Cette situation est similaire à un site web qui charge immédiatement toutes les images : c'est lent et inutilement gourmand en ressources.
Solution de l'analogie :
Lorsque j'entre dans la bibliothèque, au lieu de me donner tous les livres immédiatement, l'employé me donne un catalogue (ou une tablette électronique moderne). Le parcoure le catalogue et demande seulement les livres que je souhaite consulter. L'employé va alors chercher ces livres spécifiques pour moi. Si je veux un autre livre plus tard, je peux le demander à ce moment-là. Cette méthode est beaucoup plus efficace et moins gourmande en ressources.
Dans cette analogie, le catalogue agit comme un "proxy". Il nous donne un aperçu de ce qui est disponible (comme une image de chargement ou un espace réservé sur un site web) et nous permet de demander le véritable objet (le livre ou l'image réelle) uniquement lorsque nous en avons besoin.
En fin de compte, dans les deux situations (site web et bibliothèque), le but est d'améliorer l'efficacité et l'expérience utilisateur en fournissant des ressources uniquement lorsqu'elles sont demandées ou nécessaires.