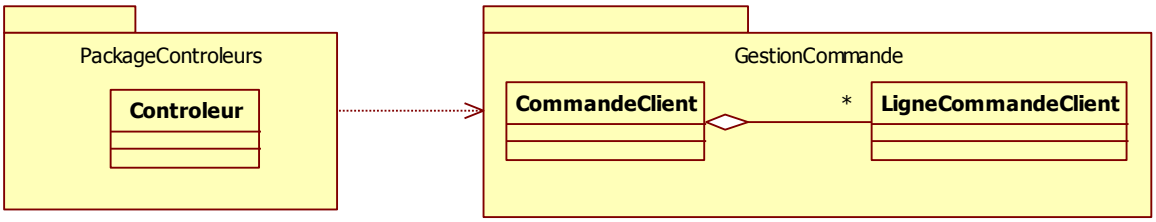
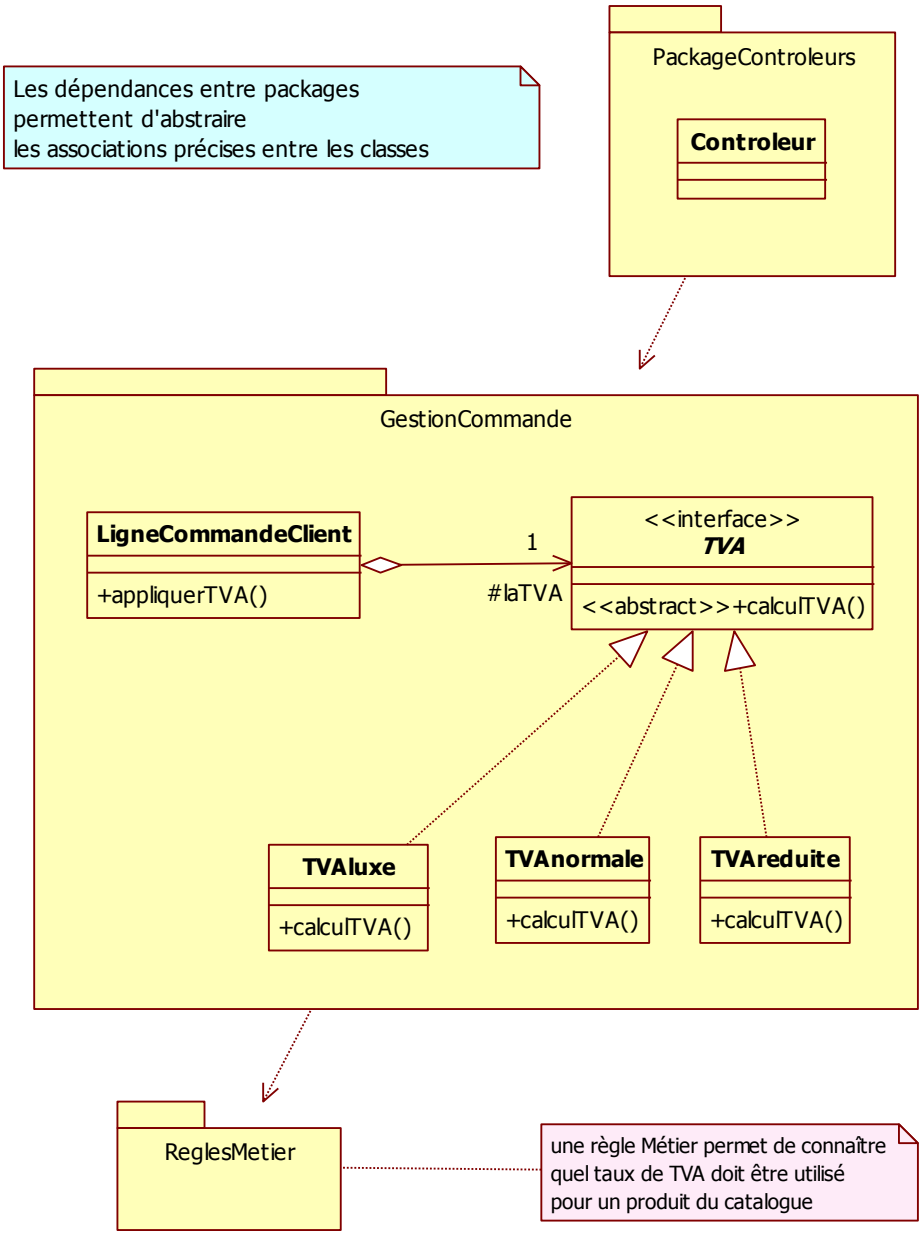
UE INF222
*Corrigé*** *extrait CC* 10 février 2012

**** Remarque : le maximum de points était obtenu lorsque les réponses brèves et simples utilisaient quelques mots clés ! dans le temps imparti, il s'agissait de tester vos « réflexes » de modélisation. Ce document n'est pas un corrigé ! c'est un commentaire élargi.**

Supposons que vous ayez développé un système de commerce électronique. L'architecture globale comporte un objet de contrôle qui gère des commandes. Cet objet reconnaît l'arrivée d'une commande et la transmet à un objet CommandeClient, regroupement d'objets LigneCommande, qui en effectue le traitement. Une vue partielle très abstraite de l'architecture est illustrée par le diagramme de classes **DC1**suivant :



En écoutant les informations à la radio, vous entendez qu'un nouveau taux de TVA pourrait être créé pour les produits importés … pas de souci pour votre modèle ! La classe LigneCommandeClient possède entre autres une opération qui permet de calculer la TVA à appliquer. Une vue partielle de l'architecture de votre conception est illustrée par ce diagramme de classes **DC2** :



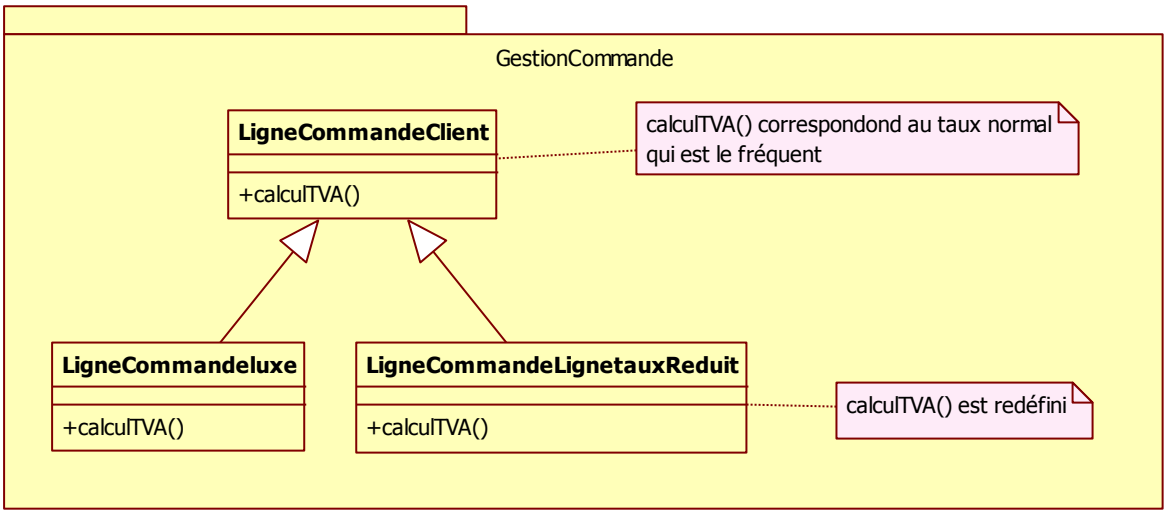
Question 1 : quel Design Pattern aviez-vous appliqué ?

Il s'agit du Design Pattern **Stratégie** dont l'objectif est : Définir une famille d'algorithmes, encapsuler chacun d'eux et les rendre interchangeables. Ce DP permet à l'algorithme de varier indépendamment des clients qui l'utilisent. Le GOF classe ce DP dans la catégorie comportementale.

Commentaire à propos de la question 1 :
Le diagramme de classes du DP Stratégie ressemble à celui du DP Etat. Pourtant l'objectif de ce dernier est bien différent : Un objet s'adapte en fonction de son état interne. Le DP Etat s'appliquerait à une classe active gérée par le cycle de vie de ces états, ce qui n'est pas le cas de Commandeclient.
Il ne s'agit pas non plus d'un DP de Fabrication d'objets : pas d'instanciation dans les classes dérivées !

Question 2 : quels sont les avantages de votre modélisation illustrée par DC2 ?

Une conception « naïve » utilisant l'héritage aurait été celle décrite en **DC5**. L'extension serait difficilement maintenable (si on admet que le changement des taux de TVA va devenir fréquent et comportera de plus en plus de diversités !).

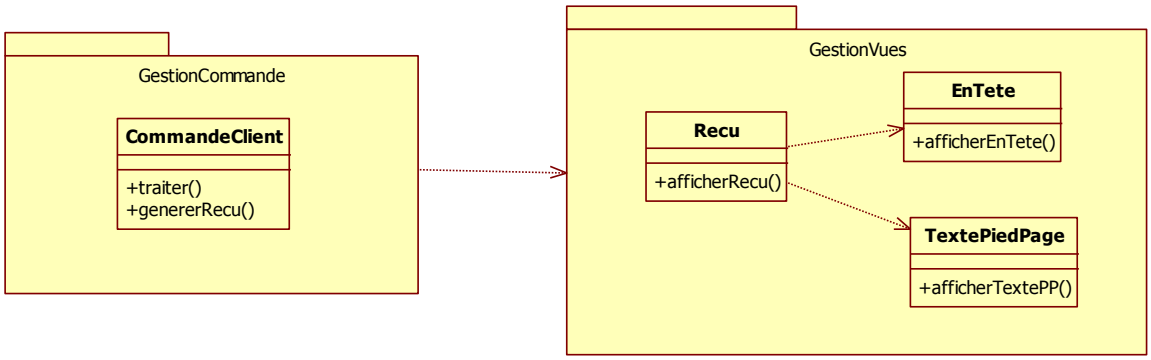


Une règle importante des DP est de préférer l'agrégation d'objets à l'héritage de classes.

La modélisation **DC2** appliquant le DP Stratégie évite le piège de l'héritage en préservant la « variabilité » du modèle. Le « découplage » des règles métier y contribue aussi.

Une autre règle de bonne conception est de « responsabiliser » les objets. Cette règle est mise en œuvre dans le DP Stratégie : les différentes implémentations de ces responsabilités sont mises en œuvre par le polymorphisme (TVA est une Classe Interface). De plus, dans le modèle **DC5**, le choix de la taxe à appliquer vient du Contrôleur qui décide du type de CommandeClient/LigneCommandeClient. Dans le modèle **DC2**, ce choix revient soit au contrôleur, soit à la LigneCommandeClient qui interroge les règles métiers. Cette modélisation permet aux règles métier de varier indépendamment d'une instance de LigneCommandeclient. Ce découplage qui permet de modifier la classe responsable d'un comportement sans en affecter une autre est une bonne pratique de modélisation.

Une autre fonctionnalité attendue du système est de générer un reçu de la commande pour l'afficher. Dans le cahier des charges, l'entête et le texte en pied de page à faire figurer sur ce reçu sont décrits. Une vue partielle de votre modélisation est illustrée par le diagramme de classes **DC3** :



Question 3 : Cette solution est correcte mais quelle limitation a-t-elle?

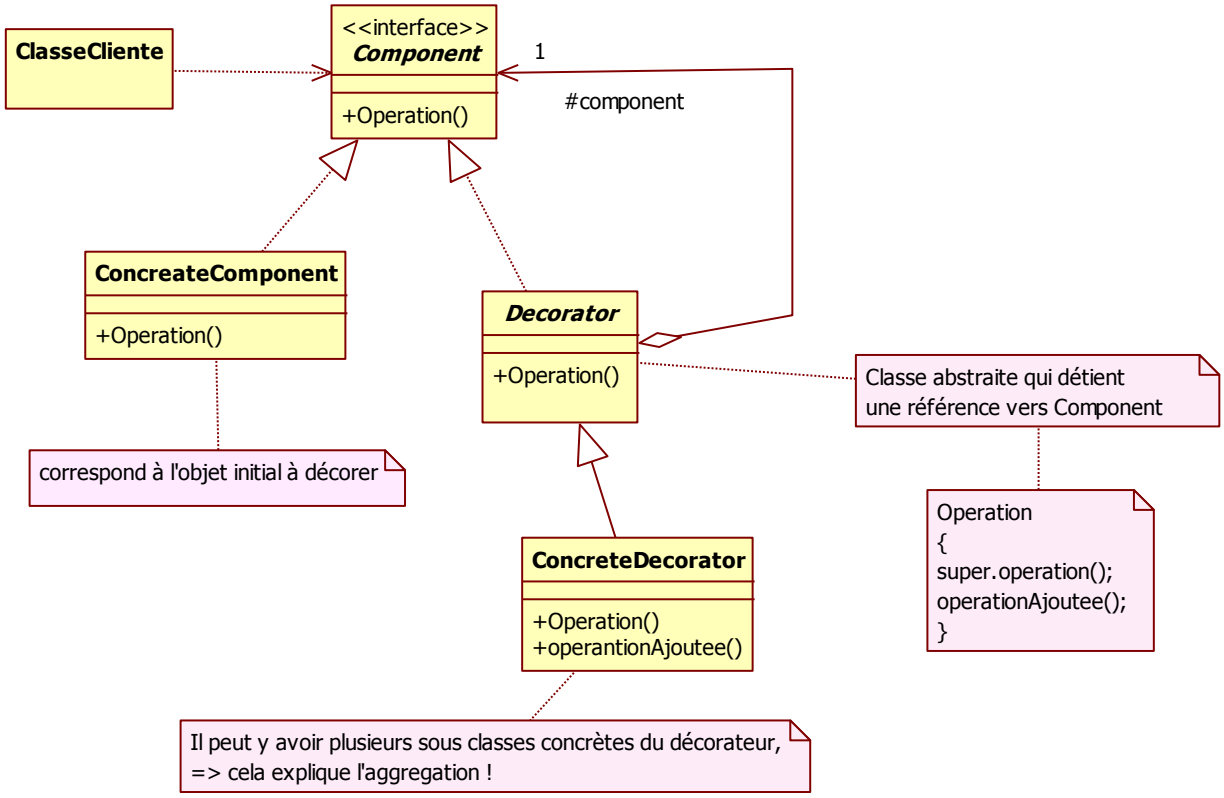
Cette solution suppose qu'il y a toujours la même entête et le même pied de page ce qui est par ex le cas si le site ne concerne qu'une seule société. Le contrôle de ce qui doit être affiché est dans une instance de Recu. Cette solution est extensible si le nombre d'options reste en faible nombre.

S'il faut gérer plusieurs types de textes à insérer, mais en imprimer un seul à la fois, pourquoi ne pas appliquer le DP Stratégie… deux fois ! un DP pour l'entête, un autre pour le pied de page.

De plus, la spécification évolue ! Il faut gérer plusieurs types d'entêtes et plusieurs textes de pied de page à insérer simultanément.

Donc le double DP Stratégie n'est pas applicable pour afficher plusieurs textes en haut et en bas et en plus modifier l'ordre d'impression…

Vous n'aviez pas anticipé cette demande. En parcourant le GOF, vous vous arrêtez sur le Design Pattern Décorateur, que vous aviez vu en TD ! Voici son modèle UML :



Question 4 : Quel est l'objectif essentiel d'application ce DP Décorateur dans ce cas?

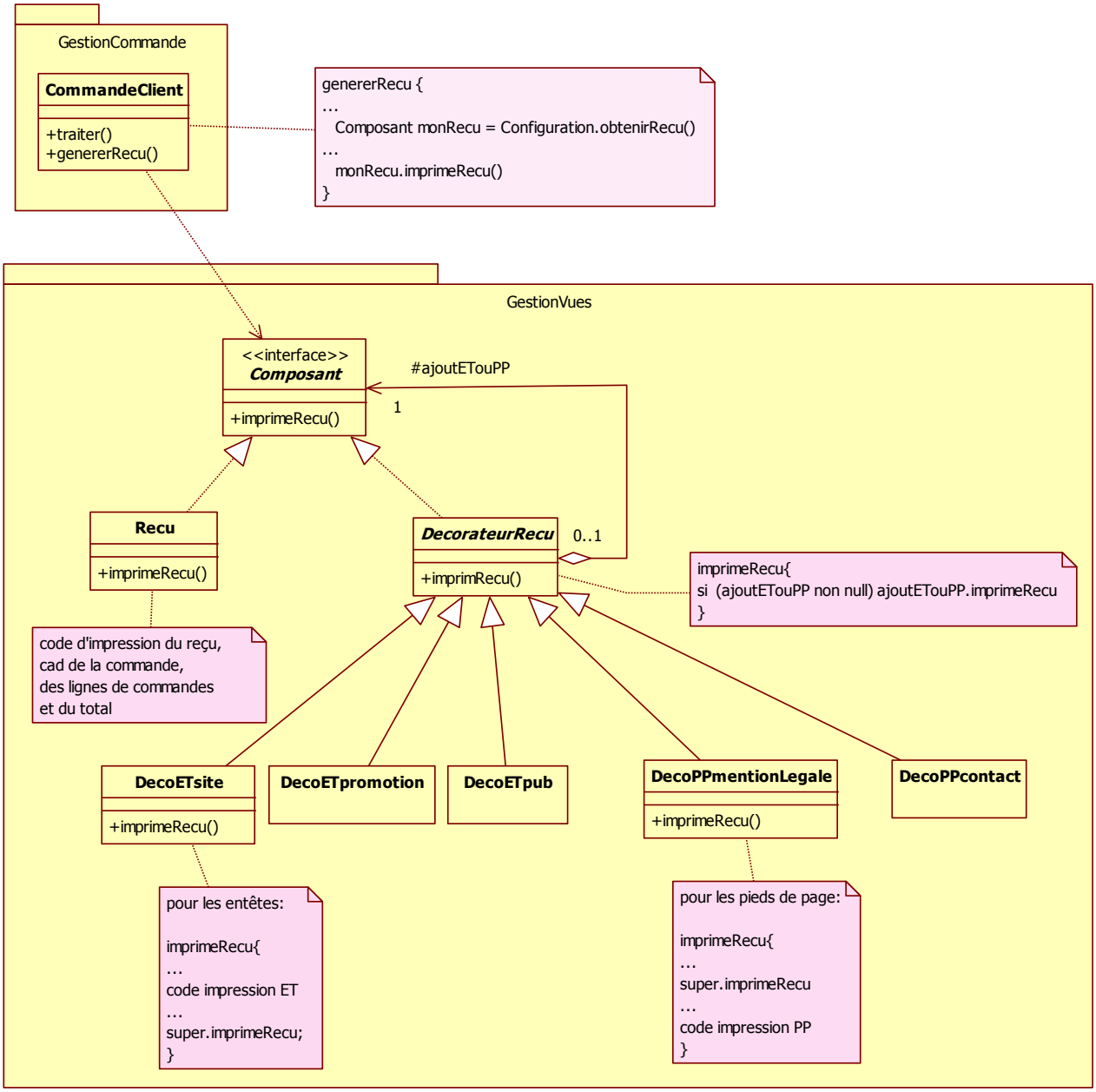
Vous ne contrôlerez plus la fonctionnalité ajoutée à l'aide d'une méthode de contrôle (dans Recu) mais en enchaînant les fonctions voulues selon un ordre donné. Le pattern décorateur sépare ensuite la construction dynamique de cette chaîne de fonctions et le client qui l'utilise (ici CommandeClient)

Question 5 : Appliquez ce DP à votre modèle.

a) Dessinez le diagramme de classes DC4.

b) pour vérifier, dessinez le diagramme de séquences pour le cas où il y a 2 entêtes (ETsite, ETpromotion) et un pied de page (PPmentionLegale)

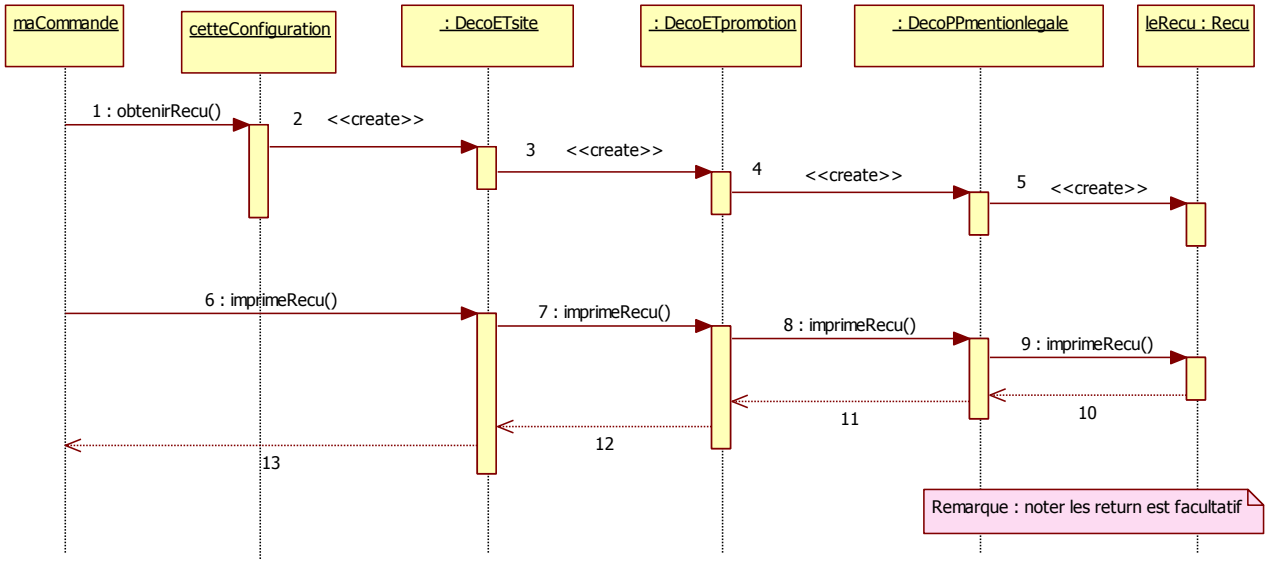
Diagramme DC4 :



DecoETsite par exemple effectue la méthode imprimeRecu qui permet d'imprimer l'entête et appelle la méthode imprimeRecu de DecorateurRecu qui appelle imprimeRecu du Composant suivant s'il y en a, etc. Ce pattern permet de créer une chaîne d'objets qui commence par les objets décorateurs, c'est-à-dire ceux responsables de la nouvelle fonctionnalité d'impression et se termine par l'objet initial. Cet ordre de chaîne d'objets correspond à l'application du DP. Pour placer en entête ou en pied de page un texte, c'est le code de imprimeRecu des décorateurs concrets DecorETsite ou DecorPPmentionLegale qui changent l'ordre de présentation.

Dans cet exemple, on suppose que la fonctionnalité est décrite dans un objet de configuration. La méthode obtenirRecu, pour répondre à la question 5 b) doit retourner
Return (new DecorETsite (new DecorETpromotion (newDecorPPmentionLegale (new Recu()))))

On peut vérifier le scénario d'enchaînement des appels pour l'impression sur un diagramme de séquence. Sur ce diagramme on a aussi représenté la construction des objets.



On a découplé la construction des objets nécessaires de l'impression de elle-même. Cela pourrait se faire simultanément mais la pratique du découplage est recommandée.

Une application différente du DP serait de prendre ET l'entête et PP le pied de page pour Composant Concret et de les gérer séparément par 2 Decorateurs.