Université de Yaoundé 1
Département d'informatique
Option Génie Logiciel
Année académique: 2023 - 2024

INF 4067 - Design Pattern

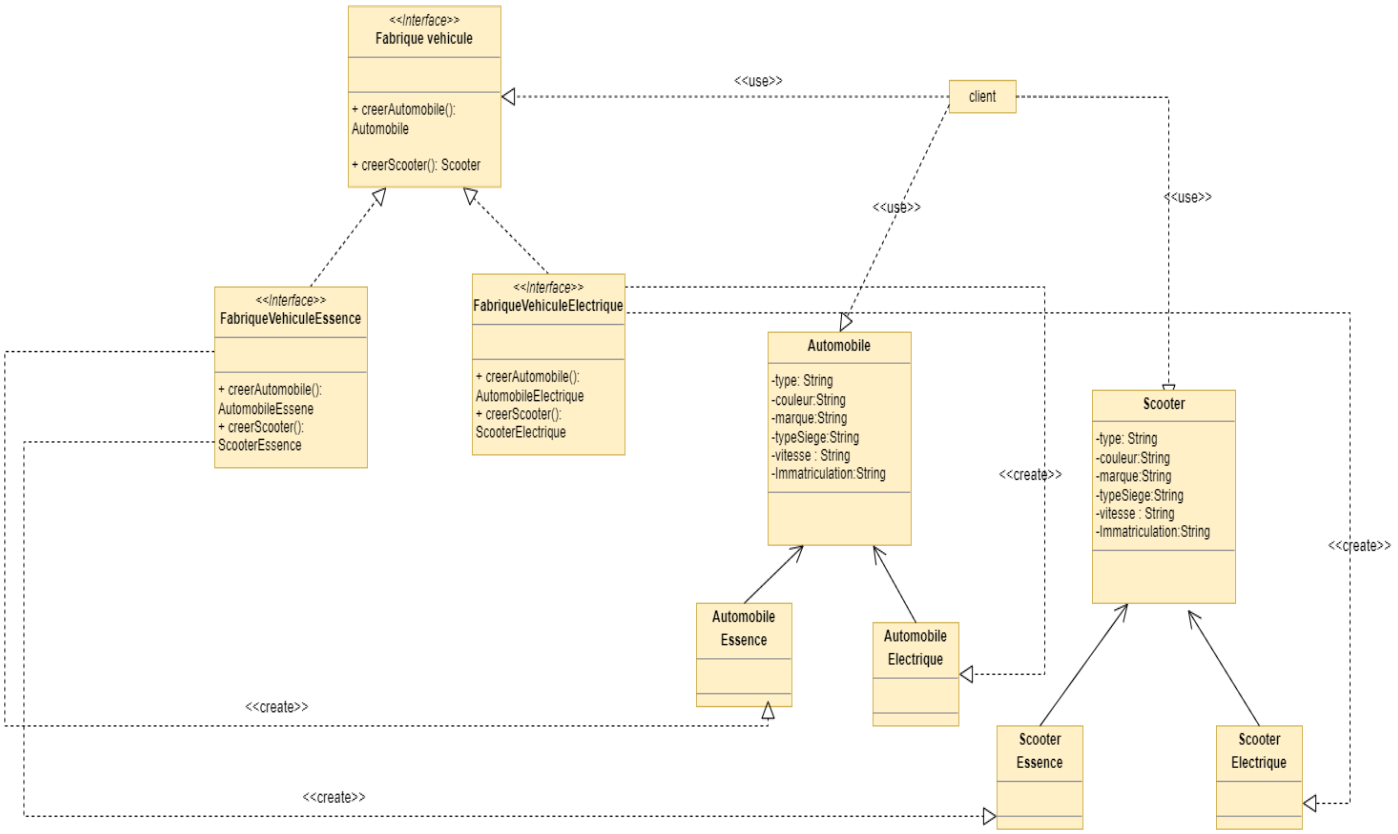
Modélisation et explication pour les différents patrons associés aux différents problèmes du cas d'étude

Groupe6

LEUNA FIENKAK NKEHEUP	20U2698
NEGOUE MAFO PATRICIA	20U2603
KAMDA MALVINA EVA	20U2843
TEGUIMENE YENDJI FUREL DE CONSOL	22V2453

Supervisé par **Dr VALERY MONTHE**

1. **<u>Abstract Factory</u> : Construire les objets du domaine (automobile à essence ou électrique , scooter à essence ou électrique)**

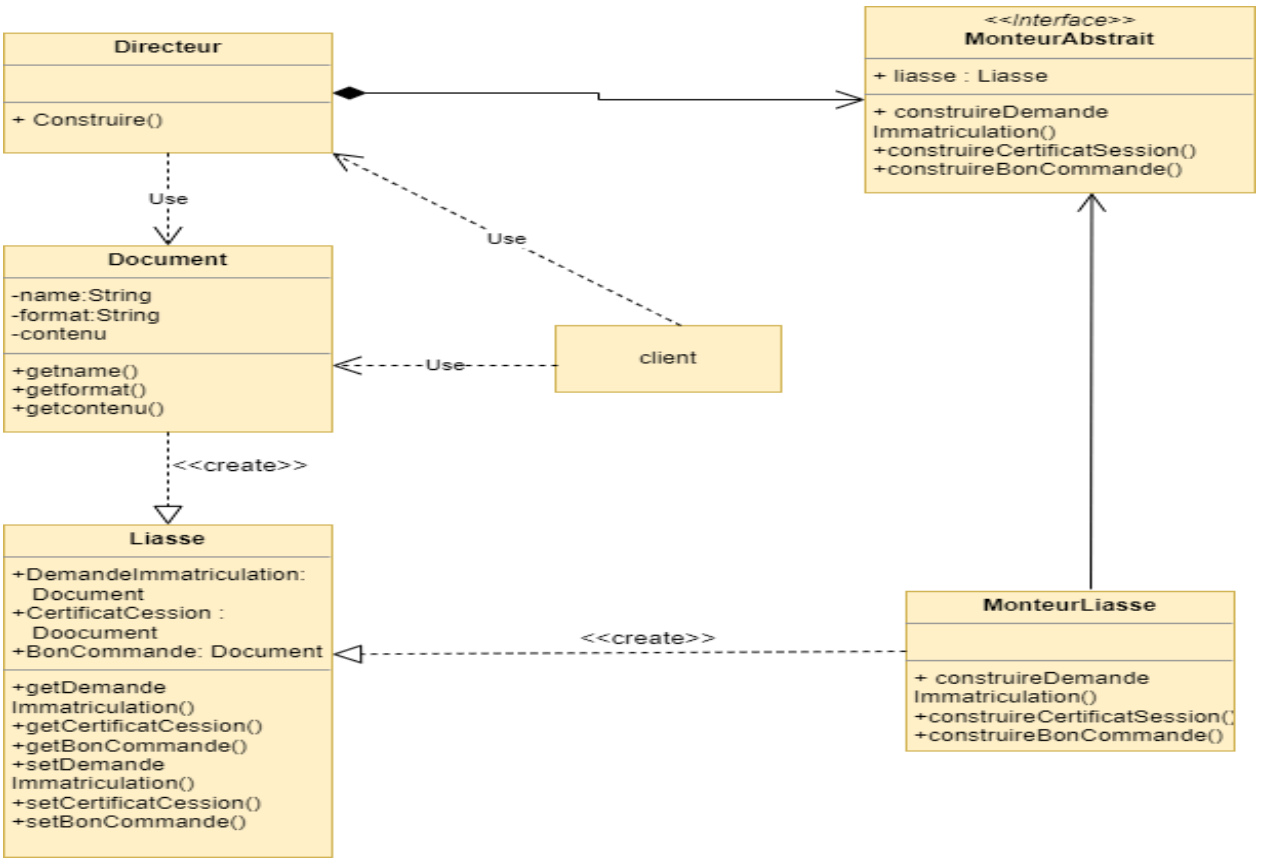


2. Builder : Construire les liasses de documents nécessaires en cas d' acquisition d'un véhicule

La liasse est constituée de 3 documents (demande d'immatriculation, certificat de cession, bon de commande). En appliquant le patron Builder, on construit la liasse en 3 étapes:

- construction de la demande d'immatriculation
- construction du certificat de cession
- construction du bon de commande

qui sont des méthodes que possède notre monteur concret de liasse. On considère ici que la demande, le certificat et le bon de commande sont représentés par des objets de type documents (nom, format) qui seront fournis par le client.



3. Factory Method : créer les commandes

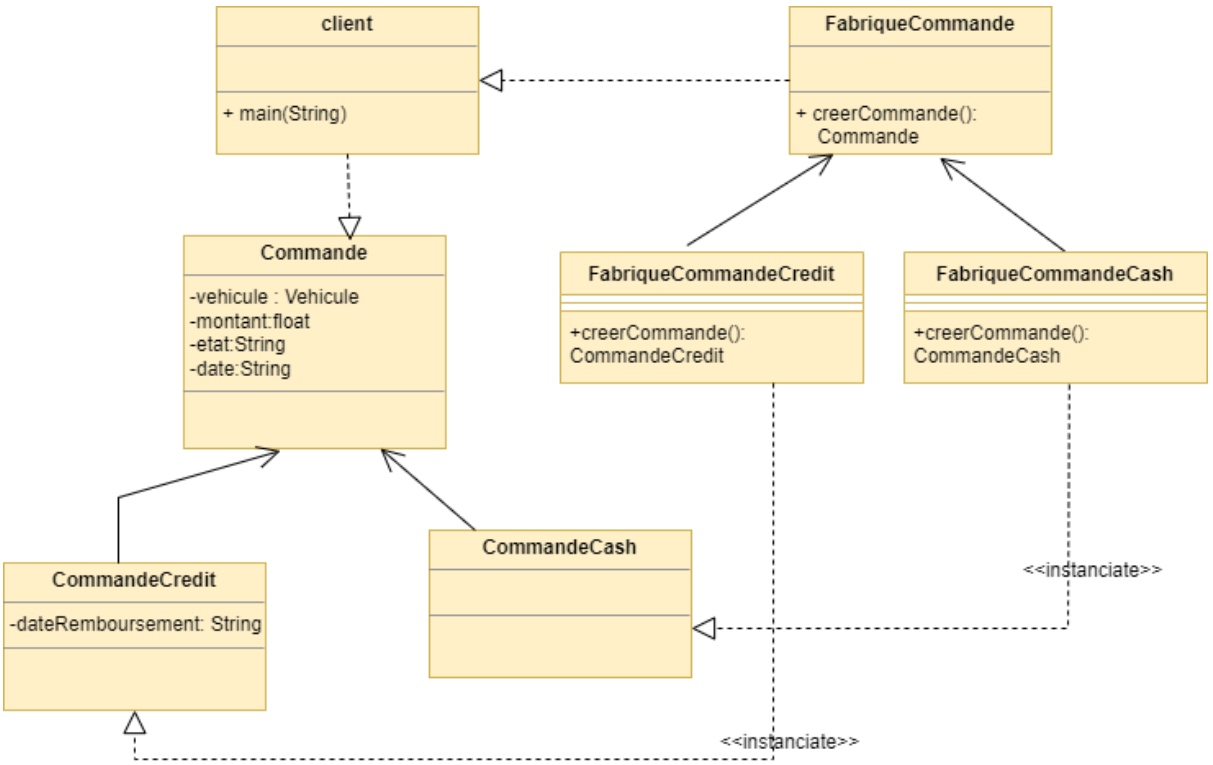
Le système gère les commandes en fonction du contexte:

- les commandes assorties de crédit (commandes Credit)
- les commandes payée au comptant (commandes Cash)

En appliquant la 2e méthode du patron Factory method, on a deux fabriques concrètes pour la création de ces types de commandes:

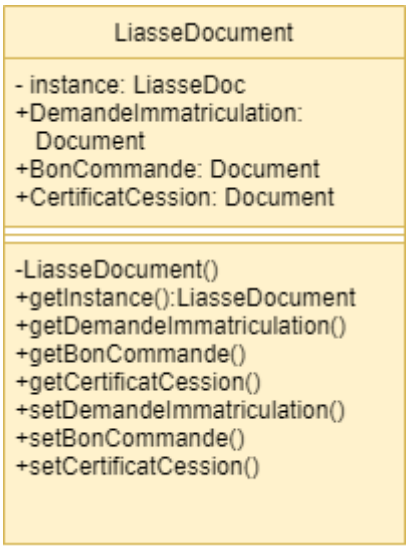
- une fabrique pour les commandes Crédit et
- une pour les commandes Cash.

on a la modélisation suivante:



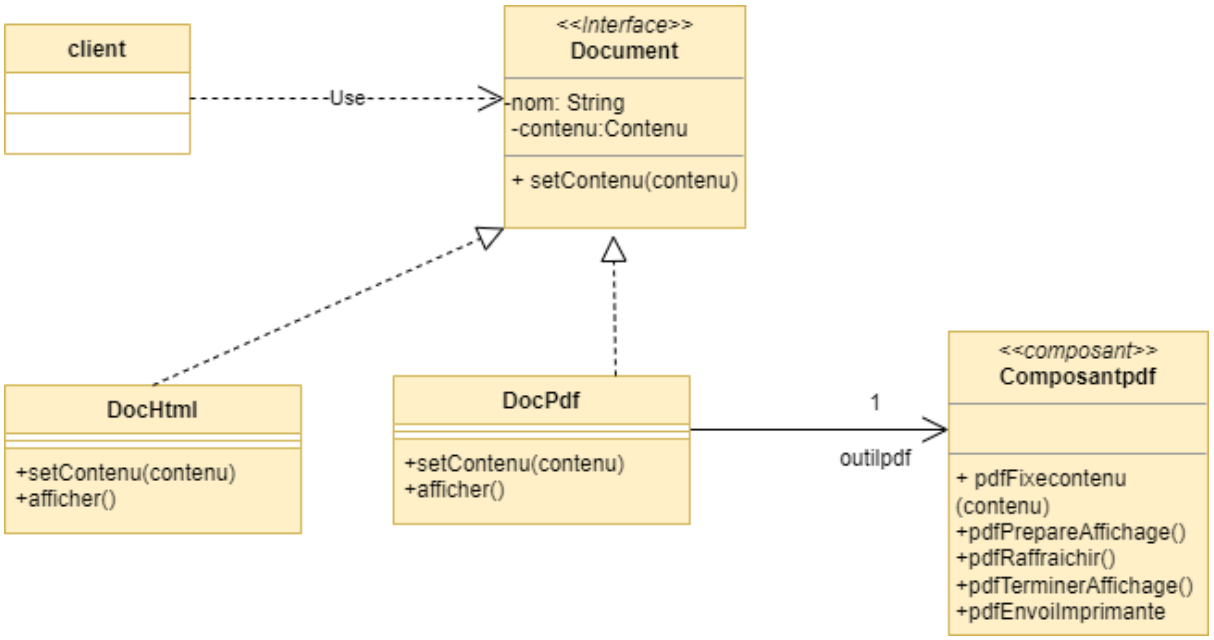
4. Singleton : créer la liasse vierge de documents

La liasse de document doit être unique lors de l'acquisition d'un véhicule. On modifie la classe "Liasse" de la question 2 pour la rendre finale, on ajoute une instance privée de liasse, on rend le constructeur par défaut privé et on définit une méthode de classe pour la récupération de l'instance unique en cours d'utilisation.



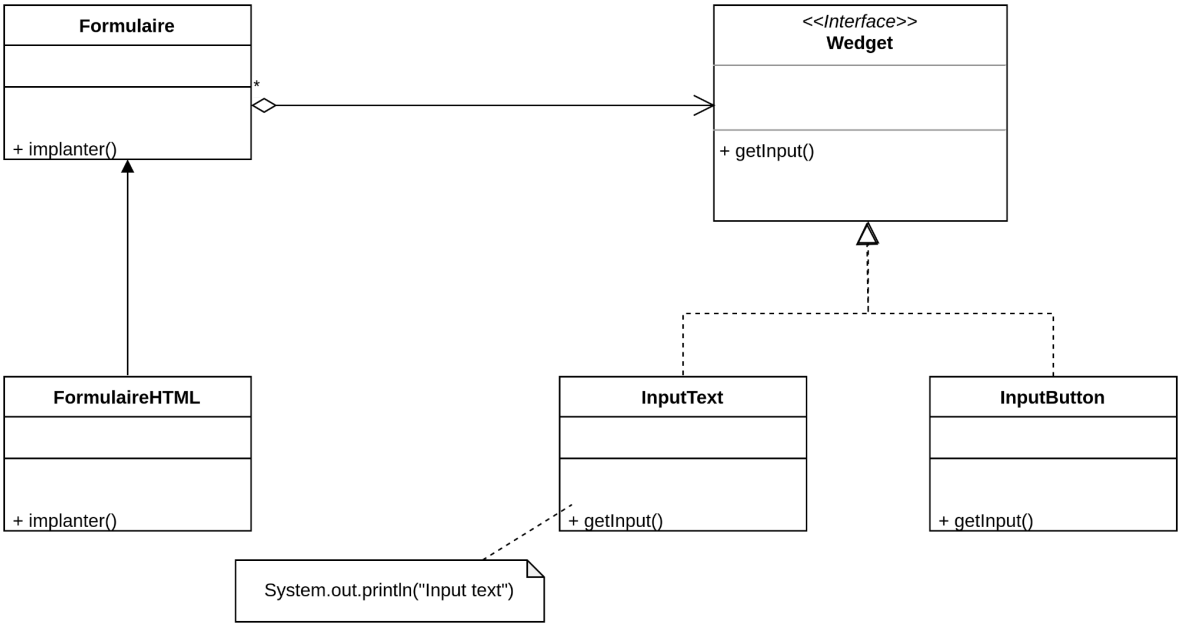
5. Adapter : Gérer des documents pdf

La modélisation de la question 2 nous montre qu'une liasse est constituée des documents qui sont des objets possédant un nom et un format. Et d'après l'énoncé, ils peuvent être soit au format pdf, soit au format Html. On décompose donc la conception de la classe "document" de la modélisation de la question 2 pour avoir une interface "Document" et des sous classes Document Html et Document Pdf. En supposant qu'on a un outil permettant de gérer les documents pdf de manière générale, on met en place un adaptateur à cet outil a l'aide du patron adapter pour gérer les documents pdf de notre système.



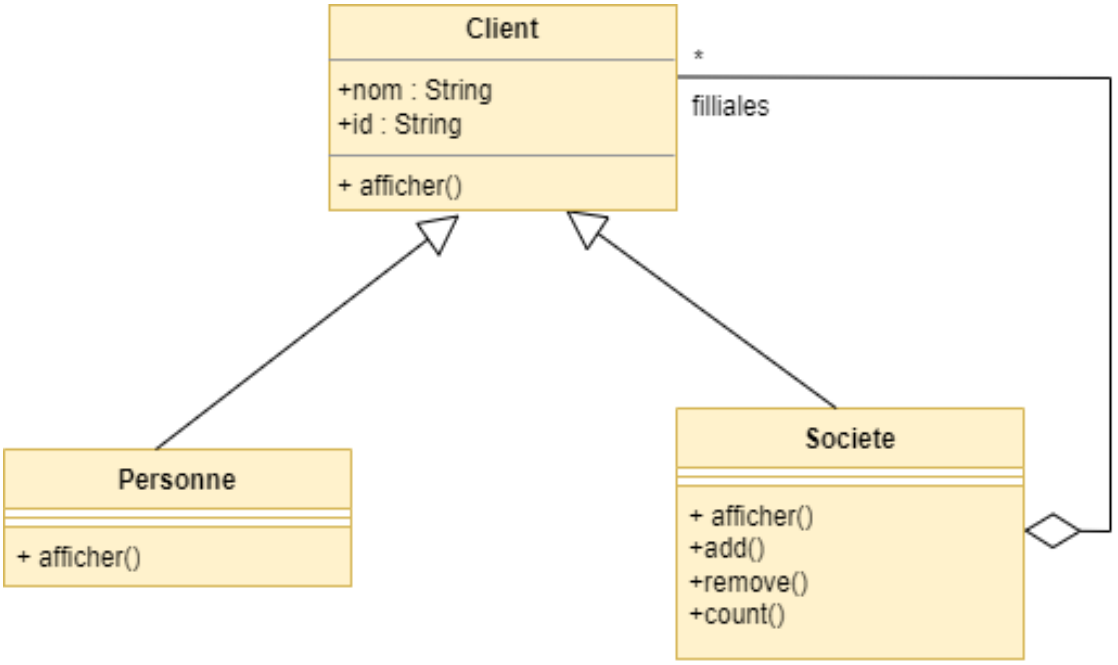
6. Bridge : Implanter des formulaires Html à l'aide des widgets

Le but est d'implanter les formulaires HTML a partir des widgets. On comprend donc qu'un formulaire est constitué par composition d'un ensemble de widgets, ceux pouvant varier/évoluer d'un formulaire à un autre. A l'aide du patron Bridge, on sépare donc le formulaire et ses widgets, de manière à ce que ceux-ci puissent évoluer séparément.



7. **composite: Représenter les sociétés clientes**

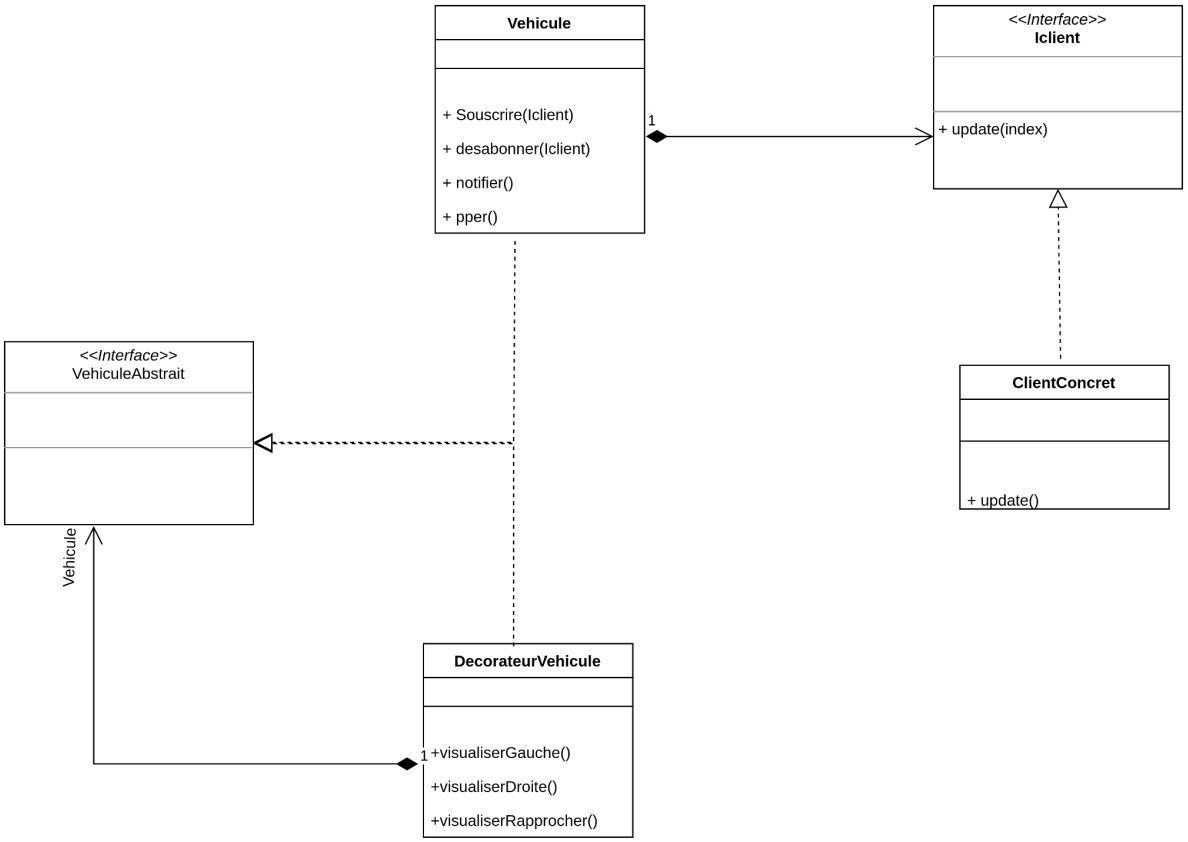
Le système doit gérer les clients. Un client peut être soit une personne, soit une société (constitué d'un ensemble de clients). On peut donc ajouter, retirer, compter ou récupérer un client d'une société.



8. Decorator , Observer : Afficher les véhicules du catalogue

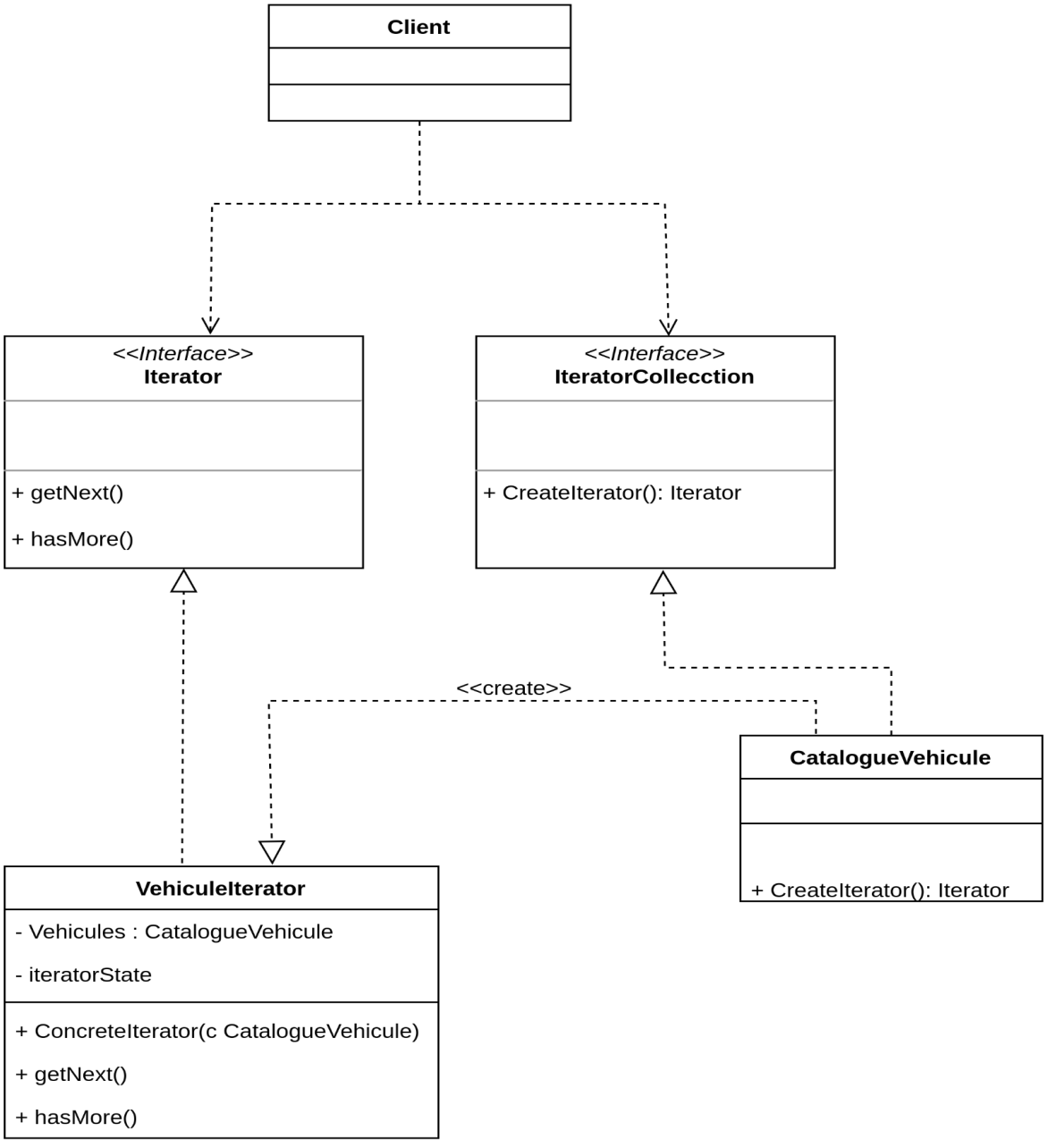
Avec le patron Observer, on définit le fait que un objet du domaine (véhicule) est considéré comme diffuseur et les clients comme les souscripteurs. Les clients peuvent donc souscrire à un véhicule et pour tous les clients qui auront souscrit à un véhicule, si jamais l'état du véhicule venait à changer (le prix peut changer, la couleur, le type de siège...), ceux-ci seront notifiés que l'état du véhicule a changé. On modifie donc la classe Client de la modélisation de la question 7 pour ajouter les méthode de souscription, désabonnement, notification et changement d'état. Et comme on peut avoir plusieurs types de client, on définit une interface Client que tous les clients concrets vont implémenter.

Le patron décorateur est utilisé pour donner dynamiquement la possibilité à un véhicule de pouvoir visualiser ses animations tels que tourner à gauche, tourner à droite, zoomer. on a donc un décorateur pour la visualisation des animations d'un véhicule.



9. iterator : Retrouver séquentiellement les véhicules du catalogue

On établit un algorithme de recherche en fonction de la marque, et de la couleur. A l'aide du patron iterator, on crée un itérateur pour le catalogue de véhicules et on utilise cet itérateur pour parcourir le catalogue et retourner les véhicules qui répondent aux critères de recherche.



10. Template Method : calculer le montant d'une commande

Pour permettre le calcul des montants des commandes en fonction des pays de livraison, on définit le squelette de la méthode de calcul d'une commande à l'aide du patron Template Method en se basant sur le calcul du montant TVA et en fonction du pays de livraison, on applique le Taux de TVA correspondant. en considérant le cameroun et la côte d'Ivoire, on a la modélisation suivante: