**Université des Sciences et de la Technologie Houari Boumediene**
**Faculté d'Informatique**
***Département SI***

Module : Génie Logiciel 02 / GL2
Spécialité : L3 ISIL B Année Universitaire : 2024 / 2025

# TP05 : Patron Stratégie

## 1. Exemple 1 : du système de vente en ligne de véhicules

### Description

Dans le système de vente en ligne de véhicules, la classe VueCatalogue dessine la liste des véhicules destinés à la vente. Un algorithme de dessin est utilisé pour calculer la mise en page en fonction du navigateur. Il existe deux versions de cet algorithme :

- une première version qui n'affiche qu'un seul véhicule par ligne (un véhicule prend toute la largeur disponible) et qui affiche le maximum d'informations ainsi que quatre photos ;

- une seconde version qui affiche trois véhicules par ligne mais qui affiche moins d'informations et une seule photo.

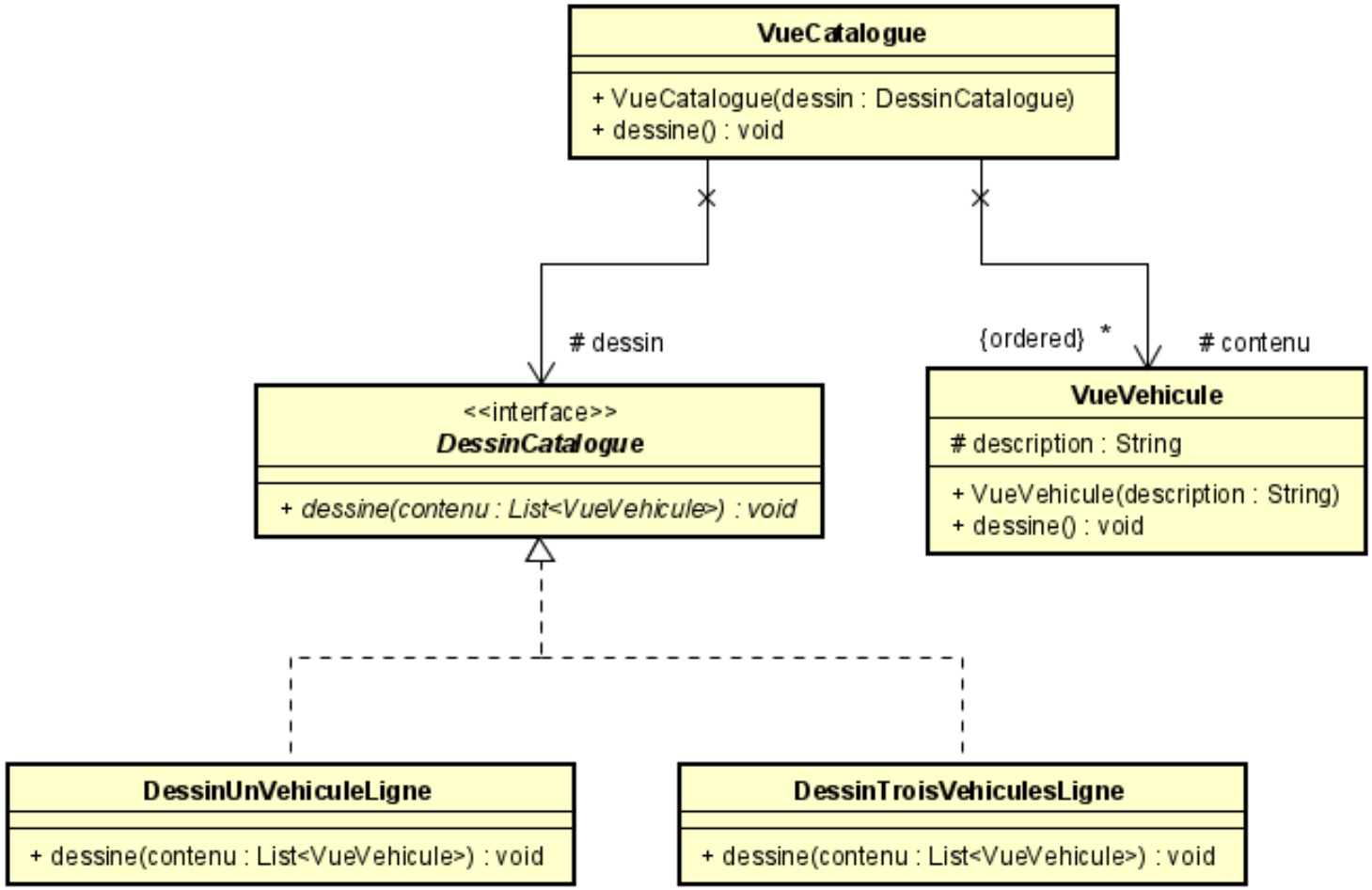
L'interface de la classe VueCatalogue ne dépend pas du choix de l'algorithme de mise en page. Ce choix n'a pas d'impact sur la relation d'une vue de catalogue avec ses clients. Il n'y a que la présentation qui est modifiée.

Une première solution consiste à transformer la classe VueCatalogue en une interface ou en une classe abstraite et à introduire deux sous-classes d'implantation différant par le choix de l'algorithme. Ceci présente l'inconvénient de complexifier inutilement la hiérarchie des vues de catalogue.

Une autre possibilité est d'implanter les deux algorithmes dans la classe VueCatalogue et à l'aide d'instructions conditionnelles d'effectuer les choix. Mais cela consiste à développer une classe relativement lourde et dont le code des méthodes est difficile à appréhender.

Le pattern Strategy propose une autre solution en introduisant une classe par algorithme. L'ensemble des classes ainsi créées possède une interface commune qui est utilisée pour dialoguer avec la classe VueCatalogue.

## Diagramme de classes correspondant



## Définition des classes

### a) L'interface DessinCatalogue

L'interface *DessinCatalogue* introduit la méthode dessine qui prend en paramètre une liste d'instances de VueVehicule.

### b) La classe DessinUnVehiculeLigne

La classe *DessinUnVehiculeLigne* implante la méthode dessine en affichant chaque véhicule sur une ligne (impression d'un saut de ligne après l'affichage d'un véhicule). ;

### c) La classe DessinTroisVehiculesLigne

La classe *DessinTroisVehiculesLigne* implante la méthode dessine en affichant trois véhicule par ligne (impression d'un saut de ligne après l'affichage de trois véhicules). ;

### d) La classe VueVéhicule

La classe *VueVehicule* qui dessine un véhicule. Il convient de noter que la méthode dessine de *VueVehicule*, dans le cas de cette simulation, est identique pour un affichage sur une ligne et sur trois lignes, ce qui n'est pas le cas dans la réalité d'une interface graphique.

### e) La classe VueCatalogue

La classe VueCatalogue possède un constructeur qui prend comme paramètre une instance de l'une des classes d'implantation de DessinCatalogue, instance qui est mémorisée dans l'attribut dessin. Ce constructeur initialise également le contenu qui devrait être normalement lu depuis une base de données.

La méthode dessine redirige l'appel vers l'instance mémorisée dans dessin. Elle transmet en paramètre une référence vers le contenu du catalogue.

### f) La classe Utilisateur

La classe Utilisateur contient la méthode main qui crée deux instances de VueCatalogue, la première est paramétrée par le dessin sur trois lignes. La seconde instance est paramétrée par le dessin sur une ligne.

Après les avoir créés, la méthode main invoque la méthode dessine de ses instances.

# 2. Exemple 2

## Description

Une entreprise qui possède plusieurs salles de cinémas dans plusieurs pays. Elle désire implémenter une solution pour la vente de tickets pour tous ses cinémas.

Le tarif du ticket dépend de la catégorie du client : enfant, adulte et Sénior.

De plus, chaque pays applique sa propre règlementation pour le calcul des taxes. Le prix final d'un ticket est la somme du tarif d'un ticket et la taxe appliquée par un pays sur ce tarif.

Modéliser cette solution.