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

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

TP 06 Patron Observateur

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

Description

Nous voulons mettre à jour l'affichage d'un catalogue de véhicules en temps réel. Chaque fois que les informations relatives à un véhicule sont modifiées, nous voulons mettre à jour l'affichage de celles-ci.

Il peut y avoir plusieurs affichages simultanés.

La solution préconisée par le pattern Observer consiste à établir un lien entre chaque véhicule et ses vues pour que le véhicule puisse leur indiquer de se mettre à jour quand son état interne a été modifié.

Définition des classes

a) La classe Sujet

La classe **abstraite** *Sujet* introduit tout objet qui notifie d'autres objets des modifications de son état interne ;

b) La classe Véhicule

La sous-classe *Véhicule* est une classe concrète de *Sujet* qui décrit les véhicules. Elle gère deux attributs : *description* et *prix* ;

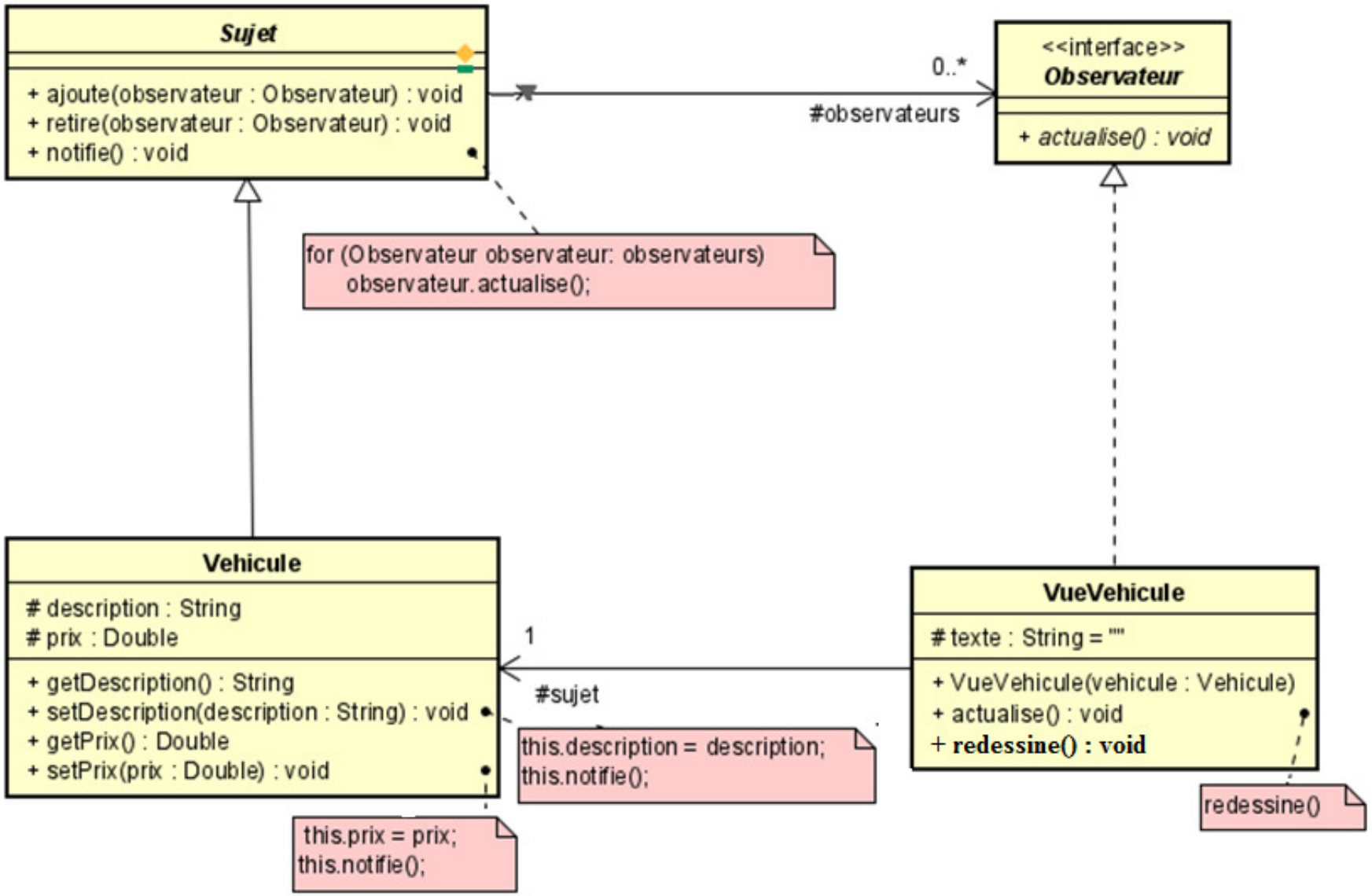
c) L'interface Observateur

L'interface *Observateur* est l'interface de tout objet qui a besoin de recevoir des notifications de changement d'état provenant des objets auprès desquels il s'est préalablement inscrit ;

d) La classe VueVéhicule

La sous-classe *VueVéhicule* est la classe concrète d'implantation d'Observateur dont les instances affichent les informations d'un véhicule.

Diagramme de classes correspondant



Le fonctionnement du patron de Observer

Le fonctionnement est le suivant : chaque nouvelle vue s'inscrit en tant qu'observateur auprès de son véhicule à l'aide de la méthode ajoute. Chaque fois que la description ou le prix sont mis à jour, la méthode notifie est appelée. Celle-ci demande à tous les observateurs de se mettre à jour en invoquant leur méthode actualise. Dans la classe *VueVéhicule*, cette dernière méthode appelle redessine. Ce fonctionnement est illustré par le diagramme de séquence suivant :