Pattern Command

(repris de L.Debrauwer)

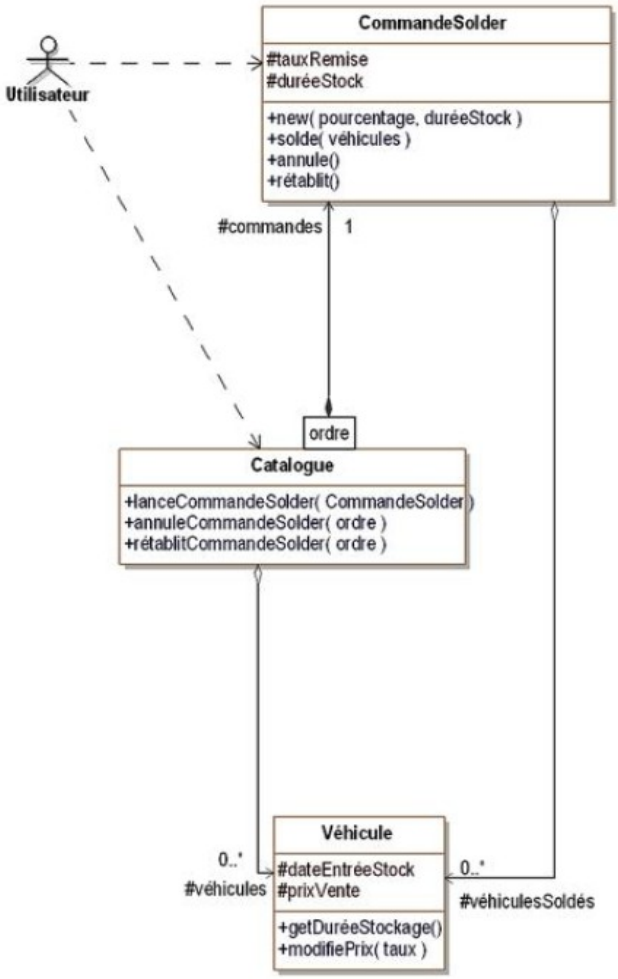
Le pattern Command a pour objectif de transformer une requête en un objet, facilitant des opérations comme l'annulation, la mise en file des requêtes et leur suivi.

Exemple

Dans certains cas, la gestion d'une commande peut être assez complexe : elle peut être annulable, mise dans une file d'attente ou être tracée. Dans le cadre du système de vente de véhicules, le gestionnaire peut demander au catalogue de solder les véhicules d'occasion présents dans le stock depuis une certaine durée. Pour des raisons de facilité, cette demande doit pouvoir être annulée puis, éventuellement, rétablie.

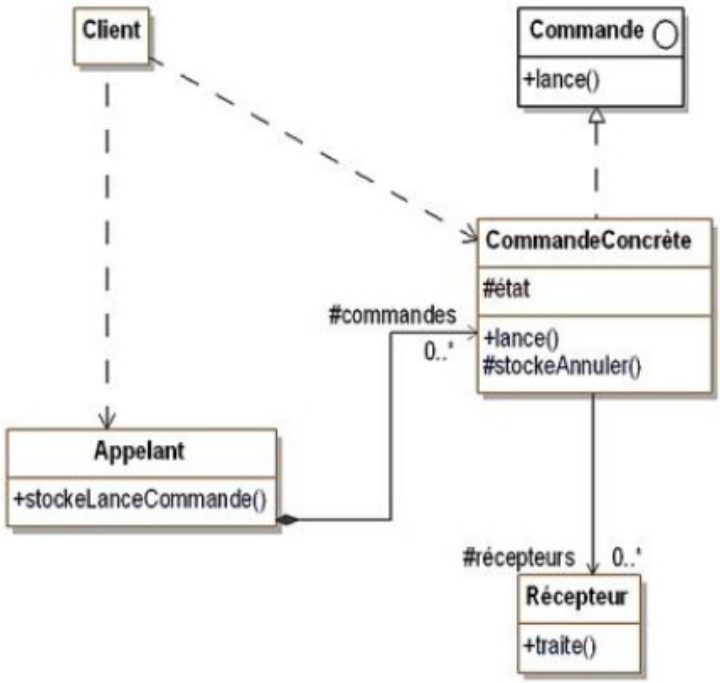
Pour gérer cette annulation, une première solution consiste à indiquer au niveau de chaque véhicule s'il est ou non soldé. Cette solution n'est pas suffisante car un même véhicule peut être soldé plusieurs fois avec des taux différents. Une autre solution serait alors de conserver son prix avant la dernière remise, mais cette solution n'est pas non plus satisfaisante car l'annulation peut porter sur une autre requête de remise que la dernière.

Le pattern `Command` résout ce problème en transformant la requête en un objet dont les attributs vont contenir les paramètres ainsi que l'ensemble des objets sur lesquels la requête a été appliquée. Dans notre exemple, il devient ainsi possible d'annuler ou de rétablir une requête de remise.



La figure ci-dessus illustre cette application du pattern Command à notre exemple. La classe `CommandeSolder` stocke ses deux paramètres (pourcentage et duréeStock) ainsi que la liste des véhicules pour lesquels la remise a été appliquée (association véhiculesSoldés).
Il faut noter que l'ensemble des véhicules référencés par `CommandeSolder` est un sous ensemble de l'ensemble des véhicules référencés par Catalogue.
Lors de l'appel de la méthode lance `CommandeSolder`, la commande passée en paramètre est exécutée puis elle est stockée dans un ordre tel que la dernière commande stockée se retrouve en première position.

Structure du pattern Command



Participants

Les participants au pattern sont les suivants :

- `Commande` est l'interface qui introduit la signature de la méthode lance qui exécute la commande ;
- `CommandeConcrète` (CommandeSolder) implante la méthode lance, gère l'association avec le ou les récepteurs et implante la méthode stockeAnnuler qui stocke l'état (ou les valeurs nécessaires) pour pouvoir annuler par la suite ;
- `Client` (Utilisateur)crée et initialise la commande et la transmet à l'appelant ;
- `Appelant` (Catalogue)stocke et lance la commande (méthode stockeLanceCommande) ainsi qu'éventuellement les requêtes d'annulation ;
- `Récepteur` (Véhicule)traite les actions nécessaires pour effectuer la commande ou pour l'annuler.

Collaboration

Le diagramme de séquence ci-dessous illustre les collaborations du pattern.

- Le client crée une commande concrète en spécifiant le ou les récepteurs ;

- le client transmet cette commande à la méthode `stockeLanceCommande` de l'appelant qui commence par stocker la commande ;
- l'appelant lance ensuite la commande en invoquant la méthode `lance` ;
- l'état ou les données nécessaires à l'annulation sont stockés (méthode `stockeAnnuler`) ;
- la commande demande au ou aux récepteurs de réaliser les traitements.