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

TP 07 : Patron Composite

Exemple 1 : du système de vente en ligne de véhicules

Description

Au sein de notre système de vente de véhicules, nous voulons représenter les sociétés clientes, notamment pour connaître le nombre de véhicules dont elles disposent et leur proposer des offres de maintenance de leur parc.

Les sociétés qui possèdent des filiales demandent des offres de maintenance qui prennent en compte le parc de véhicules de leurs filiales. Une société peut posséder des filiales qui possèdent elles-mêmes d'autres filiales.

Définition des classes

a) La classe Societe

La classe **abstraite** *Societe* définit une méthode concrète (ajouteVehicule) et deux autres abstraites (*calculeCoutEntretien* et *ajouteFiliale).*

- `Void ajouteVehicule() :` compte le nombre de véhicule d'une société.
- `boolean ajouteFiliale(Societe filiale) :` renvoie un résultat booléen qui indique si l'ajout a pu ou non être réalisé.

b) Le classe SocieteSansFiliale

Cette classe étend la classe *Societe*. Elle calcule le coût d'entretien en multipliant le nombre de véhicules par le coût unitaire par véhicule. Les instances de cette classe ne peuvent pas ajouter de filiales.

c) La classe SocieteMere

Elle est composée de filiales. Le résultat de la méthode *calculeCoutEntretien* est la somme du coût des filiales et du coût de la société mère.

d) La classe utilisateur

La méthode main de créer une société mère qui possède un véhicule et deux filiales.

La première filiale possède un véhicule tandis que la seconde filiale en possède deux. La société mère possède donc quatre véhicules, pour un coût total d'entretien de 20 (le coût pour un véhicule est de 5) clients.

Diagramme de classes correspondant