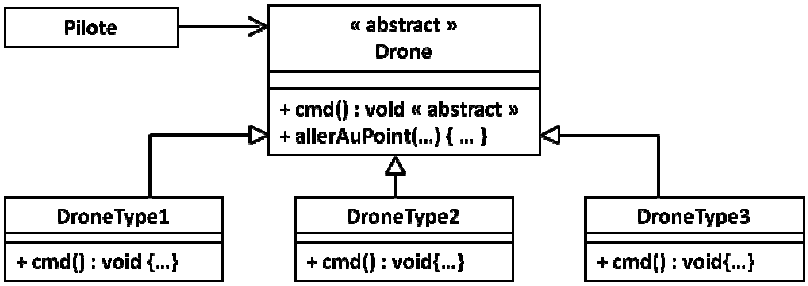
M1 - Design Patterns – Cours n°1 - Exercice

La société UPS-Software a obtenu le contrat de maintenance d'une application à base de drones aériens. Il existe de nombreux types de drones mais tous offrent des méthodes communes, en particulier des méthodes permettant à un « pilote » de les télécommander. Ces méthodes sont abstraites dans la classe mère Drone et implantées dans les différentes classes filles qui définissent les types de drones. Ici, par souci de simplification, il n'apparait qu'une seule méthode appelée `cmd()` dans le diagramme de classes

En réponse à une demande de modification, UPS-Software a défini et implanté dans la classe Drone la méthode `void allerAuPoint(Longitude longitude, Latitude latitude)` qui permet à un drone de se déplacer et de se positionner au dessus d'un point géographique dont les coordonnées sont passées en paramètre.



Dès les premiers tests, on constate quelques accidents… En effet, les caractéristiques mécaniques de quelques types de drones leur interdisent de se déplacer au-delà d'une certaine distance. A l'évidence, l'implantation partagée de la méthode `allerAuPoint()` ne convient pas à tous les types de drones.

a. *Quelle est l'origine du problème et quel concept objet en est la cause ? Expliquer.*

Une première solution simple est proposée : dans les classes qui posent problème, la méthode `allerAuPoint()` est …… de sorte que le déplacement soit contrôlé (en fonction des caractéristiques mécaniques). Les tests se passent alors sans problème. Ouf !

UPS-Software imagine cependant que d'autres évolutions de même nature vont se produire : par exemple, ajouter dans la classe Drone une méthode `void monterAuNiveau(Altitude altitude)` qui doit permettre à un drone de se placer à une altitude donnée, ce qui n'est pas possible pour certains types de drones limités techniquement.

b. *Critiquer la première solution proposée (ci-dessus, en a).*

Une nouvelle solution est alors proposée : identifier les aspects des drones qui peuvent varier, définir une interface pour chaque type de comportement variable, implanter chacune avec les différentes variantes (sans duplication du code) et associer ces implantations aux implantations des drones (de sorte que les drones les utilisent).

c. *Développer la solution : donner le diagramme de classes ainsi que les éléments de code Java utiles (la solution intègrera séparément le comportement de déplacement en longitude-latitude et le comportement de déplacement en altitude).*

d. *Le design pattern employé ici est nommé « …… ». Expliquer les avantages d'une telle solution en en cas de :*
   - *nouvelle implantation d'une variante de comportement,*
   - *changement dynamique de comportement,*

   Discuter aussi les avantages en matière de réutilisation des comportements.