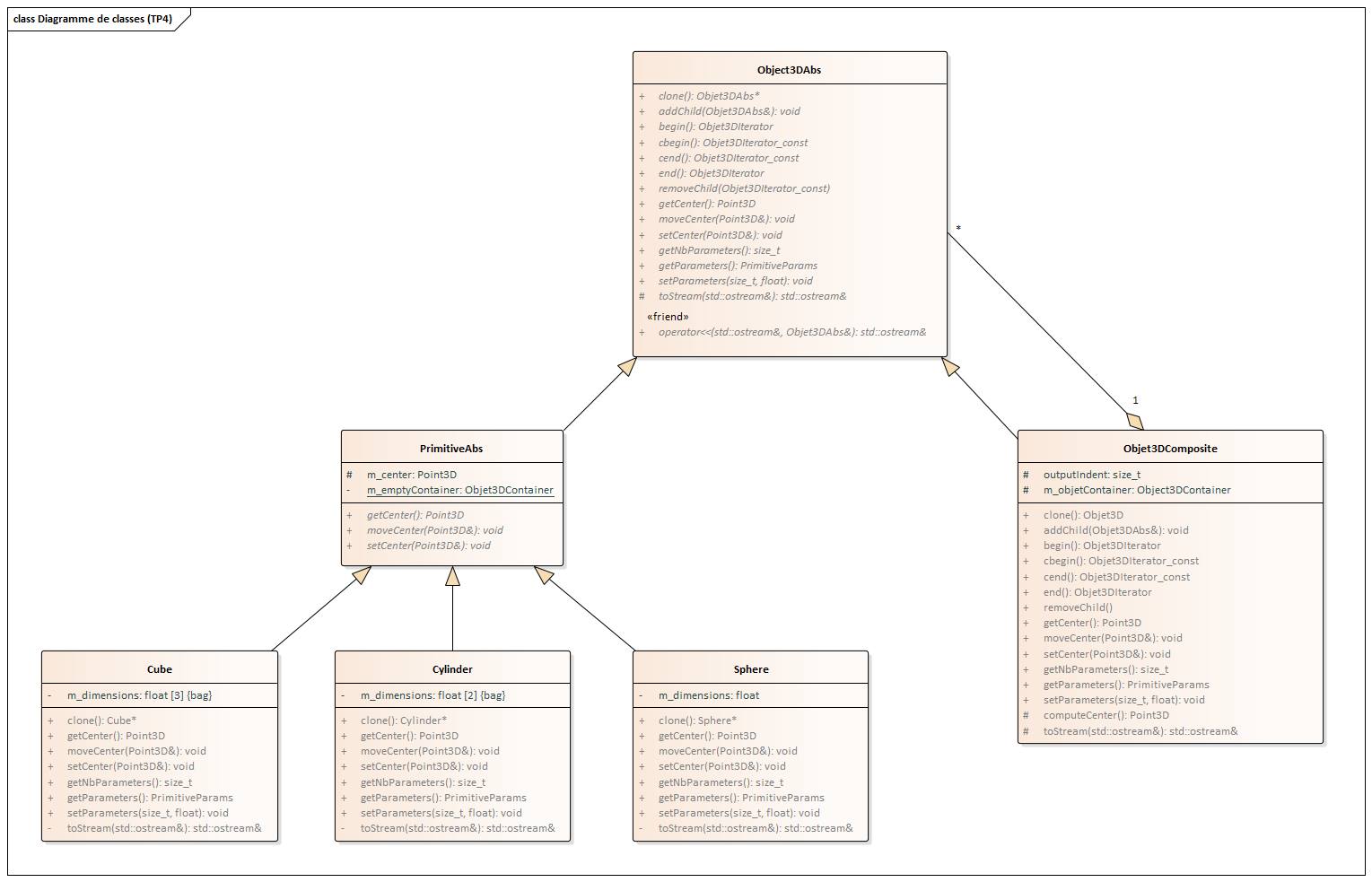
# **TP 4 – Questions théoriques**

## 1 – Patron Composite

1. Identifiez les points suivants :
   1. L’intention du patron Composite.

Le Patron Composite vise la création d’une structure de classes en arbre, permettant ainsi des structures composites qui seront traités de la même manière que les structures non-composites (feuilles). Ceci permet donc d’uniformiser l’utilisation de ces structures pour simplifier l’expérience de l’utilisateur.

* 1. La structure des classes réelles qui participent au patron ainsi que leurs rôles.

Composite

Leaves

Abstract Component

1. Identifiez la ou les abstractions présentent dans la conception du TP4, et pour chacune, identifiez les responsabilités spécifiques qui lui ont été assignées.
2. *Objet3DAbs* : représente une interface qui permet l’utilisation uniforme de tous les objets, composites ou non. Elle définit donc toutes les méthodes utilisées par les objets et c’est à travers elle que l’utilisateur les manipulera.
3. *PrimitiveAbs*: représente l’interface d’utilisation d’objets singuliers qui permet d’obtenir et manipuler les paramètres et le point central des primitives. Cette abstraction permet d’implémenter les méthodes de gestion du point central, tandis que celles servant aux manipulations de paramètres seront spécifiques à chacune des primitives concrètes.

## 2 – Patron Décorateur

1. Identifiez les points suivants :
   1. L’intention du patron Décorateur.

Le patron décorateur permet d’ajouter des attributs ou des méthodes (et donc d’étendre les fonctionnalités) à des classes déjà existantes, et ce sans modifier la classe de base. Cet ajout dynamique peut tout aussi bien être retiré, ce qui augmente grandement la flexibilité des classes et évite la création d’un réseau d’héritage complexe.

* 1. La structure des classes réelles qui participent au patron ainsi que leurs rôles.  
     (paste screenshot here)

1. Identifiez les responsabilités des classes primitives qui sont réinterprétées lorsque le Décorateur est utilisé.

L’utilisation du décorateur dans cet exemple implémente l’ajout de deux paramètres de transformation dans les classes primitives. Ces dernières auront maintenant la possibilité d’être déplacées (translation) et agrandies (mises à l’échelle). Les classes primitives pourront donc effectuer les mêmes actions que précédemment, en plus de retourner le nouveau centre après translation et les nouveaux paramètres après mise à l’échelle.

1. Selon vous, pourquoi dans la conception actuelle, un Décorateur s’applique aux primitives (classe *PrimitiveAbs*) et non à tous les objets 3D (*Objet3Dabs*) ? Serait-il possible d’appliquer le Décorateur à tous les objets et quelle en serait les conséquences ?

Not sure about the answer here :\