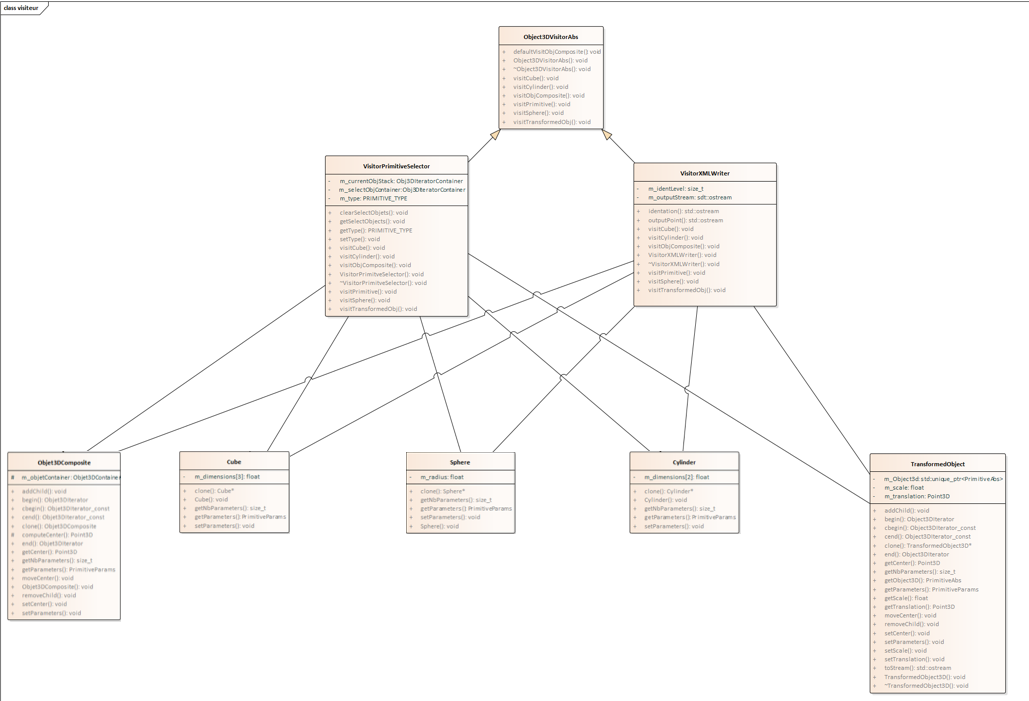
**Réponses aux questions**

**Patron Visiteur**

#1- L’intention de ce patron est de représenter une opération qui doit être appliquée sur les éléments d’une structure d’objets. Cette opération peut donc être définie sans qu’il ne soit nécessaire de modifier les classes des objets sur lesquels l’opération va agir.

#2- 

#3- L’intention du patron Template method est de définir le squelette de base d’un algorithme dans une opération, puis de laisser les sous-classes définir le reste des étapes et des paramètres.

#4- Les classes impliquées dans le patron Visiteur sont visibles dans le diagramme de classe présenté plus tôt. Les méthodes impliquées sont les méthodes visitObjComposite(const class Object3DComposite& comp), visitCube(class Cube& cub), visitCylinder(class Cylinder& cyl), visitSphere(class Sphere& sph), visitTransformedObj(class TransformedObj& tobj) et visitPrimitive(class PrimitiveAbs& prim) pour les deux classes (VisitorPrimitiveSelector et VisitorXMLWriter).

Les classes impliquées dans le patron Template Method sont la classe Obj3Dabs et la classe PrimitiveAbs. Les méthodes impliquées sont std ::ostream operator<<(std ::ostream o, const Object3DAbs& obj3d).

L’avantage de l’approche Visitor est qu’on ajoute la méthode qui affiche toutes les composantes d’un type de primitive donné, sans avoir à implémenter de nouvelles méthodes dans les classes des primitives. Le désavantage est qu’il faut implémenter une nouvelle classe complète, avec ses méthodes et attributs, afin de simplement implémenter l’opération qui sélectionne des primitives d’un type donné.

L’avantage de l’approche Template Method est qu’elle implémente la méthode d’affichage de différentes primitives ou ObjetComposite en ajoutant simplement la méthode Object3DAbs ::toStream, cependant cette approche nécessite d’implémenter une méthode différente pour toutes les primitives et ObjetComposite.

#5- Aucune classe ne serait modifiée. C’est là l’intérêt du patron Visitor; l’opération de sélectionner des primitives est indépendante des classes et donc, ajouter une nouvelle classe de primitive n’affecte en rien les autres classes.

#6- Oui, le patron Visiteur pourrait être utilisé afin d’appliquer des transformations aux primitives. Cela permettrait d’appliquer les transformations aux primitives via l’objet Visiteur, sans devoir ajouter des méthodes de transformations aux objets Primitives. Il faudrait donc créer une classe Visiteur, au même titre qu’une classe Decorator, avec des attributs et des méthodes de transformations.

**Patron Command**

#1- a) Encapsuler une requête dans un objet afin que des requêtes de types différents puissent être supportées par la méthode, que des queues de requêtes puissent être définies et que l’opération « annuler » soit supportée.

b) INSÉRER ENTERPRISE ARCHITECT ICI.

#2-