**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- INSÉRER ENTERPRISE ARCHITECT ICI

#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

#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-