**Patron visiteur**

1) Identifiez l’intention du patron Visiteur.

L’intention du patron visiteur est de représenter une opération qui doit être appliquée sur les éléments d’une structure d’objets afin de permettre de définir une nouvelle opération sans modification aux classes des objets sur lesquels l’opération va agir.

2) Tracez un diagramme de classes avec Enterprise Architect illustrant les deux instances du patron Visiteur (impression en format XML et sélection des primitives) et les classes sur lesquelles agissent les visiteurs. Ajoutez des notes en UML pour indiquer les rôles de chaque classe, et incorporez le diagramme à votre document de réponses.

3) Deux approches différentes pour implémenter une fonction d’impression sur un stream sont démontrées dans le TP. La première, utilisant l’opérateur de sortie std::ostream& operator<<(std::ostream& o, const Object3DAbs& obj3d); est basée sur le patron Template Method. La seconde, utilisant la classe VisitorXMLWriter est basée sur le patron visiteur. Identifiez l’intention du patron Template Method.

4) Établissez la liste des classes et méthodes impliquées dans l’implémentation de chaque approche et discutez, en 250 mots ou moins, des avantages et inconvénients de chaque approche.

5) Si en cours de conception vous constatiez que vous voudriez ajouter une nouvelle sous-classe dérivée de PrimitiveAbs, établissez la liste de toutes les classes qui doivent être modifiées à cause de l’utilisation du patron Visitor.

6) Selon vous, l’application des transformations aux primitives pourrait-elle être implémentée comme un visiteur ? Si oui, discutez en 250 mots ou moins, des avantages et inconvénients d’utiliser le patron visiteur pour cette fonction et sinon expliquez pourquoi le patron n’est pas applicable.

**Patron Commande**

1) Identifiez les points suivants :

a) L’intention du patron Command.

L’intention du patron Commande est d’encapsuler une requête dans un objet de façon à permettre de supporter facilement plusieurs types de requêtes, de définir des queues de requêtes et de permettre des opérations d’annulation.

b) La structure des classes réelles qui participent au patron Command ainsi que leurs rôles. Faite un diagramme de classes avec Enterprise Architect, ajoutez des notes en UML pour indiquer les rôles, et incorporez le tout à votre document de réponses.

2) Observez attentivement la classe Invoker qui permet de gérer la relation entre les commandes et les icônes 3D. En plus de participer au patron Commande, cette classe agit comme Mediator entre les commandes et les icônes. Puisque la classe Invoker agit comme Mediator, il pourrait sembler logique d’utiliser le patron Singleton lors de la définition de cette classe.

a) Quel sont les intentions des patrons de conception Mediator et Singleton?

L’intention du patron Mediator est de définir un objet qui encapsule la manière dont un ensemble d’objets interagissent afin de promouvoir un couplage faible et de laisser varier l’interaction entre les objets de façon indépendante et l’intention du patron Singleton est de s’assurer qu’il ne soit possible de créer qu’une seule instance d’une classe et de fournir un point d’accès global à cette instance.

b) Quels sont les avantages de définir la classe Invoker comme Mediator ?

c) Discuter en 250 mots ou moins, des avantages et des inconvénients de définir la classe Invoker comme Singleton ?

3) Pour compléter la fonctionnalité de PolyIcone3D, il faudrait ajouter de nouvelles sous-classes de la classe CommandAbs. Selon vous, est-ce que d’autres classes doivent être modifiées pour ajouter les nouvelles commandes? Justifiez votre réponse.