## **RÉPONSES AUX QUESTIONS**

### **Question 2.1 : Identifiez l’intention et les avantages du patron Visiteur.**

Nous avons vu en classe que l’intention du patron Visiteur s’agit de « permettre de définir une nouvelle opération sans modifier aux classes des objets sur lesquels l’opération va agir ». En effet, cette façon de faire est permise puisque le patron Visiteur est capable de séparer l’algorithme des objets sur lesquels ils opèrent dessus. Ce patron offre alors une meilleure flexibilité puisqu’il y a indépendance entre les Visitors et la structure d’objets. Puis, étant donné que les méthodes du Visitor se retrouve tous dans une classe, c’est-à-dire un endroit commun, il y a fonctionnalité localisée.

### **Question 2.2 : Tracer un diagramme de classes avec EA pour chacune des deux instances du patron Visiteur.**

### **Question 2.4 : Selon vous, la fonction d’ajout d’un rôle pour un membre d’une équipe pourrait-elle être implémentée comme un visiteur? Si oui, discuter des avantages et des inconvénients d’utiliser le patron visiteur pour cette fonction et sinon expliquez pourquoi le patron n’est pas applicable**

### **Question 3.1 : Identifiez les points suivants :**

### **L’intention et les avantages du patron Commande;**

### **La structure des classes réelles qui participent au patron Commande ainsi que leurs rôles (faire un diagramme de classes avec Enterprise Architect, ajouter des notes en UML pour indiquer les rôles, et exportez le tout en pdf);**

### **Question 3.2 : Observez attentivement la classes *CommandInvoker* qui permet de gérer la relation entre les commandes et les différents membres des équipes. En plus de participer u patron Commande, cette classe participe à deux autres patrons de conception vus en cours.**

### **Quels sont les noms et les intentions de ces patrons de conception?**

### **Quels sont les éléments de la classe *CommandInvoker* qui sont caractéristiques de es patrons de conception?**

### **Pourquoi avoir utilisé ici ces patrons de conception?**

### **Question 3.3 : Pour compléter la fonctionnalité de *TeamViewer*, il faudrait ajouter de nouvelles sous-classes de la classe *AbsCommand*. Selon vous, est-ce que d’autres classes doivent être modifiées pour ajouter les nouvelles commandes? Justifiez votre réponse.**

### **Question 3.4 : Dans la version proposée de la commande de traduction par la classe *CommandTranslate*, on suppose que la traduction d’une langue vers l’autre peut être effectuée de façon « inversible », c’est-à-dire sans perte d’information durant la traduction. Selon vous, cette supposition est-elle réaliste? Sinon, que faudrait-il faire pour rendre l’opération « undo » robuste si on ne peut pas supposer que l’opération est inversible? On ne vous demande pas d’implémenter une solution au problème, mais simplement de décrire le problème et la façon dont vous vous y prendriez pour le régler.**