## **RÉPONSES AUX QUESTIONS**

**Question 2.1 Identifiez les points suivants :**

1. **L’intention du patron *Composite*:**

Comme le décrit les notes cours, le patron *Composite* s’agit d’un patron structurel; plus précisément, elle permet de créer une structure arborescente. En effet, ce dernier permet de traiter de ses composantes, c’est-à-dire des feuilles et des structures composites, pareillement. Par ailleurs, Nous avons vu durant le cadre du cours que l’intention du patron *Composite* est de « [t]raiter les objets individuels et les objets multiples, composés récursivement, de façon uniforme »; cela simplifie le tout puisque les différences entre les structures composite et non composite sont omises, et donc ces primitives sont utilisées de manière uniforme.

**Question 2.2**

1. **La structure des classes réelles qui participent au patron ainsi que leurs rôles. Faites un diagramme de classes avec Enterprise Architect pour l’instance du patron *Composite*. Ajouter des notes en UML pour indiquer les rôles, et exportez le tout en PDF.**

* *AbsTeamComponent*:
* Classe abstraite dont toutes les méthodes sont virtuelles pures;
* Bref, s’agit d’une interface de base.
* *TeamMember*:
* Classe concrète dérivée de la première, permettant de représenter un membre individuel d’une équipe;
* Modélise les objets de la classe *TeamMember* qui pourrait se retrouver dans *Team*.
* *Team*:
* Seconde classe concrète dérivée de la première classe qui permet de regrouper des membres ou des sous-équipes dans une équipe;
* Contient des méthodes supplémentaires à celles du *AbsTeamComponent*, ce qui lui permet de manipuler ses enfants.
* *TeamManager*:
* Peut accéder et traiter des feuilles et des structure composites uniformément.

Figure 1. Diagramme de classes du patron *Composite*.

**Question 3.1 Identifiez les points suivants :**

1. **L’intention du patron *Decorator* :**

Le patron *Decorator* est similaire à celui de type *Composite,* dans le sens où les objets sont composés récursivement. L’avantage d’utiliser ce patron est de permettre la manipulation des attributs et des méthodes à des objets, de façon à ne pas modifier sa classe de base; cela a pour effet d’augmenter la flexibilité et d’éviter les classes racines complexes.

**Question 3.2**

1. **La structure des classes réelles qui participent au patron ainsi que leurs rôles. Faites un diagramme de classes avec Enterprise Architect pour l’instance du patron *Decorator*. Ajouter des notes en UML pour indiquer les rôles, et exportez le tout en PDF.**

* *TeamMemberRole* :
* Dérive de classe de base abstraite *AbsTeamComponent*;
* Permet d’ajouter du texte sur la photo d’un membre, d’une équipe et d’afficher chaque membre avec son rôle.

Figure 2. Diagramme de classes du patron *Decorator*.

**Question 4**