LOG2410 – TP4 Automne 2019

Conception à base de patrons I

1 - Objectifs

Ce laboratoire permettra aux étudiants de se familiariser avec l'implémentation des patrons de conception « Composite » et « Decorator ». Cette implémentation est effectuée à l'aide du logiciel Visual Studio et de la bibliothèque Qt. Le langage C++ sera utilisé tout au long du processus de développement. Un cadriciel est fourni, qui doit être complété afin d'obtenir le résultat final souhaité.

2 - Patron Composite (40 points)

L'application *TeamViewer* permet d'ajouter des membres dans des équipes et d'afficher leurs photos à l'écran. Une équipe peut être composée de membres individuels ou de sous-équipes, composées elles-mêmes, de façon récursive, de membres ou de sous-équipes, selon le patron Composite. Trois classes participent au patron :

- *AbsTeamComponent* est une classe abstraite dont toutes les méthodes sont virtuelles pures.
- *TeamMember* est une classe concrète dérivée de la première, qui permet de représenter un membre individuel d'une équipe.
- *Team* est une seconde classe concrète dérivée de la première classe qui permet de regrouper des membres ou des sous-équipes dans une équipe.

Les fichiers d'entête de chacune des classes vous sont fournis et sont complets. Les seuls fichiers à compléter sont les fichiers .cpp correspondant aux deux classes concrètes. Chaque fonction à compléter est clairement identifiée et sa fonctionnalité est décrite directement dans le fichier.

Une application de test est fournie (*TeamViewer*), qui permet de vérifier interactivement le bon fonctionnement de l'application. Pour le patron Composite,

la fonction interactive « Open Team » devrait charger les photos de deux membres de 2 équipes bien connues. La fonction « Open… » permet quant à elle de charger un membre à la fois et de l'ajouter dans une équipe racine stockée comme attribut membre de la classe principale de l'application.

Implémentation

On vous demande de compléter les fichiers suivants :

- `TeamMember.cpp, et`
- `Team.cpp.`

afin que l'application interactive permette de charger les photos individuellement ou en équipe.

Questions à répondre

1) Identifiez les points suivants :
   a) L'intention du patron Composite.
   b) La structure des classes réelles qui participent au patron ainsi que leurs rôles. Faite 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.

3 – Patron Decorator (30 points)

Pour permettre d'assigner des rôles aux différents membre de chaque équipe, la classe *TeamMemberRole* permet d'ajouter du texte sur la photo d'un membre d'une équipe et ainsi d'afficher chaque membre avec son rôle. Cette e transformation simple est fournie, qui permet d'appliquer des translations dans l'espace aux sommets des triangles. La classe *TeamMemberRole* dérive, elle aussi, de la classe de base abstraite *AbsTeamComponent*, selon le patron Decorator.

Implémentation

On vous demande de compléter le fichier suivant :

- `TeamMemberRole.cpp`

afin que l'application interactive permette de charger les photos en équipe grâce au bouton « Open Role » du menu « File ».

Questions à répondre

1) Identifiez les points suivants :
   a) L'intention du patron Decorator.
   b) La structure des classes réelles qui participent au patron ainsi que leurs rôles. Faite 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).

4 – Architecture 3 niveaux et modèle MVC (30 points)

L'application *TeamViewer* est une application interactive comprenant plusieurs classes et reposant sur l'utilisation de la bibliothèque Qt. Ce type d'application devrait être développée selon les principes de séparation des responsabilités entre différents niveaux, et en appliquant le patron architectural Modèle-Vue-Contrôleur. À votre avis, telles que les classes ont été conçues, l'application TeamViewer respecte-t-elle ces principes architecturaux ? On vous demande de justifier votre réponse en identifiant quelles classes appartiennent à quel niveau architectural, et en précisant pour chacune des classes si elle assume correctement ou non des responsabilités liées au Modèle, à la Vue ou au Contrôleur. Fournissez votre réponse sous la forme d'un diagramme UML annoté de toutes les classes fournies. Le diagramme doit être organisé de façon à faire apparaitre clairement les niveaux et indiquer dans des annotations les responsabilités M, V ou C pour chaque classe. Vous pouvez également fournir un texte explicatif joint au diagramme.

5 – À remettre

1) Une archive `LOG2410_TP4_matricule1_matricule2.zip` qui contient les éléments suivants :
   a) Le fichier `ReponsesAuxQuestions.pdf` avec la réponse aux questions 2.1a), 2.2), 3.1a), 3.2) et 4)
   b) Le fichier `DiagrammeDeClasses_Composite.pdf` pour le diagramme de classes du patron composite de la question 2.1b)

c) Le fichier `DiagrammeDeClasses_Decorator.pdf` pour le diagramme de classes du patrons Decorator de la question 3.1b)
d) Le fichier `DiagrammeArchitectural.pdf` pour la partie 4.
e) Les trois fichiers C++ que vous avez modifiés , c'est-à-dire,
   i) `TeamMember.cpp`
   ii) `Team.cpp, et`
   iii) `TeamMemberRole.cpp`