LOG2410 – TP5 Automne 2019

Conception à base de patrons II

1 - Objectifs

Ce laboratoire permettra aux étudiants de se familiariser avec l'implémentation des patrons de conception « Visiteur » et « Commande ». Cette implémentation est effectuée à l'aide du logiciel Visual Studio et 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 Visiteur (40 points)

Comme vu lors du TP4, l'application *TeamViewer* permet de créer des équipes et d'afficher les photos, les noms et les rôles des membres de chaque équipe. Afin de poursuivre le développement de l'application *TeamViewer*, on veut éventuellement permettre de définir un grand nombre d'opérations sur les membres des équipes, quelle que soit la taille et la composition d'une équipe.

Ces opérations seront implémentées à l'aide du patron Visiteur. Par exemple, on veut calculer l'espace mémoire occupé par les images (photos) d'un ou des membres d'une équipe, et chercher des membres par leur nom ou leur rôle (ou une partie de ceux-ci) et remplacer une chaîne ou une partie de chaîne par une autre. Une méthode pure virtuelle `accept` recevant en paramètre une référence à un visiteur abstrait a été ajoutée à l'interface de la classe `AbsTeamComponent` afin de permettre aux visiteurs d'agir sur les équipes.

Implémentation

On vous demande de compléter les fichiers suivants :

- `TeamImageSizeCalculator.cpp`
- `MemberTextFindReplace.cpp`

afin qu'une fois complétée, les commandes pilotées à partir de l'interface graphique s'exécutent correctement. Les parties à compléter ont été clairement

identifiées par un commentaire fournissant une description des algorithmes à implémenter.

Questions à répondre

1) Identifiez L'intention et les avantages du patron Visiteur.
2) Tracer un diagramme de classes avec Enterprise Architect pour chacune des deux instances du patron Visiteur (calcul de taille et find & replace), et ajouter des notes en UML pour indiquer les rôles, et exportez le tout en pdf.
3) Si en cours de conception, si vous constatiez que vous voudriez ajouter une nouvelle sous-classe dérivée de `AbsTeamComponent`, établissez la liste de toutes les classes qui doivent être modifiées.
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 inconvénients d'utiliser le patron visiteur pour cette fonction et sinon expliquez pourquoi le patron n'est pas applicable.

3 - Patron Commande (60 points)

Afin d'opérer plus efficacement sur les membres des équipes, il pourrait être très intéressant pour un usager de pouvoir définir une séquence d'opérations qui s'appliquent à une équipe, et de pouvoir sauvegarder et exécuter ces opérations sur plusieurs membres, à la façon de macros. Le patron de conception Commande permet cette flexibilité en permettant d'exécuter différents types d'opérations en invoquant la méthode `CommandInvoker::execute()`. Deux commandes ont été identifiées qui doivent être complétées. La première commande utilise le visiteur de calcul de taille des images pour calculer la somme des tailles de toutes les images et l'afficher dans la barre de statut lorsque les équipes sont chargées en mémoire. La seconde commande utilise un dictionnaire très simplifié pour traduire les rôles des membres des équipes du français à l'anglais et vice-versa. La première commande ne peut pas être annulée, mais la seconde commande peut l'être en inversant le sens de la traduction. Pour traduire les rôles, la commande utilise le visiteur de recherche et remplacement développé à la section précédente et l'applique plusieurs fois en utilisant toutes les paires de chaines de caractères contenues dans le dictionnaire.

Pour que les commandes s'exécutent, les méthodes de la classe `CommandInvoker` doivent être complétées, incluant l'exécution des commandes reçues en argument `CommandInvoker::execute()`, l'annulation d'une commande `CommandInvoker::undo()` et la réexécution d'une commande annulée `CommandInvoker::redo()`.

Implémentation

On vous demande de compléter les fichiers suivants :

- `CommandInvoker.cpp`
- `CommandCalculateSize.cpp`
- `CommandTranslate.cpp`

afin qu'une fois complétées, les commandes pilotées à partir de l'interface graphique s'exécutent correctement. Les parties à compléter ont été clairement identifiées par un commentaire.

Questions à répondre

1) Identifiez les points suivants :
   a) L'intention et les avantages du patron Commande.
   b) La structure des classes réelles qui participent au patron Commande ainsi que leurs rôles (faite un diagramme de classes avec Enterprise Architect, ajouter des notes en UML pour indiquer les rôles, et exportez le tout en pdf).
2) Observez attentivement la classe `CommandInvoker` qui permet de gérer la relation entre les commandes et les différents membres des équipes. En plus de participer au patron Commande, cette classe participe à deux autres patrons de conception vus en cours.
   a) Quel sont les noms et les intentions de ces patrons de conception ?
   b) Quels sont les éléments de la classe `CommandInvoker` qui sont caractéristiques de ces patrons de conception ?
   c) Pourquoi avoir utilisé ici ces patrons de conception ?
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.

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.

4 - À remettre

Le TP5 est à remettre sur le site Moodle du cours au plus tard le **dimanche 1er décembre 2019 avant 11h55**. Vous devez remettre une archive `LOG2410_TP5_matricule1_matricule2.zip` qui contient les éléments suivants :

a) Le fichier `ReponsesAuxQuestions.pdf` avec la réponse aux questions posées (sauf les diagrammes de classe).
b) Le fichier `DiagrammeDeClasses_TeamImageSizeCalculator.pdf` pour le diagramme de classes du patrons Visiteur de la question 2.1b).
c) Le fichier `DiagrammeDeClasses_MemberTextFindReplace.pdf` pour le diagramme de classes du patrons Visiteur de la question 2.1c).
d) Le fichier `DiagrammeDeClasses_Command.pdf` pour le diagramme de classe de la question 3.1b).
e) Les 5 fichiers C++ que vous avez complétés, c'est-à-dire, `TeamImageSizeCalculator.cpp`, `MemberTextFindReplace.cpp`, `CommandInvoker.cpp`, `CommandCalculateSize.cpp` et `CommandTranslate.cpp`. Vous ne pouvez pas modifier les autres fichiers `.h` et `.cpp`. Le correcteur va insérer vos fichiers dans sa solution, et le tout doit compiler et s'exécuter.