LOG2410 – TP5 Hiver 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 (50 points)

Comme vu lors du TP5, le système PolyVersion permet de manipuler deux classes de fichiers audio, soit les fichiers audio stockés uniquement sur le disque dur, et les fichiers audio dont les données sont chargées en mémoire durant l'exécution du programme, selon le patron Proxy. Afin de poursuivre le développement de l'application PolyVersion, on veut éventuellement permettre de définir un grand nombre d'opérations sur les fichiers, qu'ils soient sur disque ou en mémoire.

Ces opérations seront implémentées à l'aide du patron Visiteur. Par exemple, on veut calculer une somme de contrôle (checksum) sur un fichier ou permettre de remplacer une séquence de données par une autre de même longueur (Find/Replace). Une méthode pure virtuelle `accepte` recevant en paramètre une référence à un visiteur abstrait a été ajoutée à l'interface de la classe `AbsAudioFile` afin de permettre aux visiteurs d'agir sur les fichiers audio.

Implémentation

On vous demande de compléter les fichiers suivants :

- `FileChecksumCalculator.cpp`
- `FileStringFindReplace.cpp`

afin que les tests programmés dans la méthode `Test_TP5::executeVisiteurTest()` s'exécutent avec succès. 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 du checksum 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 vous constatiez que vous voudriez ajouter une nouvelle sous-classe dérivée de `AbsAudioFile`, établissez la liste de toutes les classes qui doivent être modifiées.
4) Selon vous, l'application des transformations aux fichiers audio pourrait-elle être implémenté 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 (50 points)

Afin de modifier et transformer plus efficacement les fichiers audios, il pourrait être très intéressant pour un usager de pouvoir définir une séquence d'opérations qui s'appliquent au fichier, et de pouvoir sauvegarder et exécuter ces opérations sur plusieurs fichiers, à la façon de macros. Le patron de conception Commande permet cette flexibilité en permettant d'exécuter les différents types de modifications sur les fichiers en invoquant la méthode `CommandExecutor::executeAllCommands()`.

Implémentation

On vous demande de compléter les fichiers suivants :

- `CommandExecutor.cpp`
- `ChecksumCommand.cpp`
- `StringFindReplaceCommand.cpp`
- `TransformCommand.cpp`

afin que les tests programmés dans la méthode `Test_TP5::executeCommandeTest()` s'exécutent avec succès. 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 `CommandExecutor` qui permet de gérer la relation entre les commandes et les différents fichiers audio. En plus de participer au patron Commande, cette classe participe à deux autres patrons de conception vu en cours.
   a) Quel sont les noms et les intentions de ces patrons de conception ?
   b) Quels sont les éléments de la classe `ExecuteurCommandes` qui sont caractéristiques de ces patrons de conception ?
   c) Pourquoi avoir utilisé ici ces patrons de conception ?
3) Pour compléter la fonctionnalité de PolyVersion, 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 - À remettre

Le TP5 est à remettre sur le site Moodle du cours au plus tard le **mardi 16 avril 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_FileChecksumCalculator.pdf` pour le diagramme de classes du patrons Visiteur de la question 2.1b).

c) Le fichier `DiagrammeDeClasses_FileStringFindReplace.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 6 fichiers C++ que vous avez modifiés, c'est-à-dire, `FileChecksumCalculator.cpp`, `FileStringFindReplace.cpp`, `CommandExecutor.cpp`, `ChecksumCommand.cpp`, `StringFindReplaceCommand.cpp` et `TransformCommand.cpp`. Vous ne pouvez pas modifier les autres fichiers `.h` et `.cpp`. Le correcteur va insérer vos 4 fichiers dans le code, et ça doit compiler et s'exécuter.