LOG2410 – TP5 Automne 2016

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, la théière PolyInfusion est constituée d'un ou de plusieurs circuits hydrauliques et mécaniques. Les circuits hydrauliques permettent de stocker, chauffer et acheminer les liquides jusqu'au filtre lors de l'infusion d'un thé. Les circuits mécaniques, quant à eux, permettent de stocker et d'acheminer les solides sous forme de poudre jusqu'au filtre, où les liquides et solides sont mis en contact pour préparer le thé. La version de base de la théière PolyInfusion inclut un circuit hydraulique pour l'eau chaude et un circuit mécanique pour le thé. La version de luxe de la théière ajoute un second circuit hydraulique pour le lait, et un second circuit mécanique pour le sucre.

Dans le développement de l'application PolyInfusion, on veut éventuellement faire une panoplie d'opérations sur les circuits liquide et solide. Ces opérations seront implémentées à l'aide du patron Visiteur. Par exemple, on veut calculer le volume du circuit liquide et la puissance totale utilisée par l'ensemble des circuits de la machine. Par rapport au TP4, remarquez que les éléments de circuit liquide `ElmCircuitLiquide` et les éléments de circuit solide `ElmCircuitSolide` dérivent maintenant de la classe abstraite `ElmVisitable` qui contient la méthode pure virtuelle `accepter` recevant en paramètre un pointeur à un visiteur abstrait.

Implémentation

On vous demande de compléter les fichiers suivants :

- `VisiteurCalculVolumeLiquide.cpp`
- `VisiteurCalculerPuissance.cpp`

afin que les tests programmés dans la méthode `Test_TP5::executeVisiteurTest()` s'exécutent avec succès.

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 volume du liquide et de la puissance), et ajouter des notes en UML pour indiquer les rôles, et exportez le tout en pdf.
3) Si en cours de conception vous constater qu'il manque un type d'élément dans les circuits liquide (une sous-classe concrète de la classe `ElmCircuitLiquide`, par exemple la classe `Valve)`, établissez la liste de toutes les classes qui doivent être modifiées.
4) Selon vous, le nettoyage de la machine pourrait-il 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 (60 points)

Plusieurs cas d'utilisations de la théière PolyInfusion, tels infuser un thé, infuser un thé sélectionné, diagnostiquer la théière, nettoyer la théière, etc, nécessitent l'exécution dans un ordre spécifique d'une séquence d'opérations qui s'appliquent au circuit liquide et au circuit solide. Les programmeurs veulent évidemment programmer ces cas d'utilisation, mais dans un souci de flexibilité, veulent aussi pouvoir programmer de futurs cas d'utilisation à être livrés dans les mises à jour du firmware. Et même éventuellement, on aimerait permettre à l'usager de la théière de définir sa propre séquence d'utilisation pour obtenir un résultat spécifique non

prévu par les développeurs. Par exemple, chauffer de l'eau, ajouter beaucoup de lait et de sucre pour se faire un lait chaud et sucré pour apaiser une quinte de toux (c'est encore mieux en ajoutant un peu de rhum…) Le patron de conception Commande permet cette flexibilité de programmation.

Étudiez attentivement la classe `CommandeFixerTemperatureLiquide` ainsi que son fonctionnement. Cette partie du code est complète. Notez que cette commande inclut un visiteur qui parcourt tous les éléments du circuit liquide. On vous demande de compléter le cas d'utilisation `InfuserThe` qui exécute des commandes qui transfèrent de l'eau du réservoir vers la bouilloire et qui chauffe la bouilloire.

Implémentation

On vous demande de compléter les fichiers suivants :

- `ExecuteurCommandes.cpp`
- `CommandeTransfertLiqReservBouil.cpp`

afin que les tests programmés dans la méthode `Test_TP5::executeCommandeTest()` s'exécutent avec succès.

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 `ExecuteurCommandes` qui permet de gérer la relation entre les commandes et les différents éléments de la théière. En plus de participer au patron Commande, elle 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 ce patron de conception ?

3) Pour compléter la fonctionnalité de la théière, il faudrait ajouter de nouvelles sous-classes de la classe `CommandeAbs`. 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 6 décembre 2016 à 17h00**. 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_VisiteurCalculVolumeLiquide.pdf` pour le diagramme de classes du patrons Visiteur de la question 2.1b).
c) Le fichier `DiagrammeDeClasses_VisiteurCalculerPuissance.pdf` pour le diagramme de classes du patrons Visiteur de la question 2.1c).
d) Le fichier `DiagrammeDeClasses_Commande.pdf` pour le diagramme de classe de la question 3.1b).
e) Les 4 fichiers C++ que vous avez modifiés, c'est-à-dire, `VisiteurCalculVolumeLiquide.cpp`, `VisiteurCalculerPuissance.cpp`, `ExecuteurCommandes.cpp` et `CommandeTransfertLiqReservBouil.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.