LOG2410 – Conception Logicielle

Automne 2016

TP No. 5

Groupe 2

1792473 – Richer Archambault

1794745 – Kevin Pantelakis

Soumis à : Soumaya Medini

le 6 décembre 2016

Réponses aux questions

Partie 2

1. L'intention du patron visiteur, pour ce projet, est de pouvoir implémenter une fonctionnalité sans avoir à modifier toutes les classes du système pour lesquelles l'opération va se faire. Le principal avantage est d'éviter la dispersion du code partout dans le projet. De plus, l'ajout d'une nouvelle fonctionnalité ne demande pas de faire de changement dans la hiérarchie des classes, seulement une nouvelle classe héritant de VisiteurSansEffet.

2. Les classes suivantes devront être modifiés
   - VisiteurAbs
   - VisiteurSansEffet
   - VisiteurCalculVolumeLiquide
   - VisiteurCalculPuissance(Potentiellement)
   - Les classes des machines qui utiliseront la valve

   Lorsqu'il est question du patron Visiteur, il faut effectivement ajouter une méthode pour tous les objets visitables. En plus, il faudra également modifier les classes des machines qui vont implémenter le nouveau composant.

3. Oui effectivement il pourrait être implémenté en utilisant le patron visiteur. L'avantage principal de le faire ainsi est que tout le code se retrouverais au même endroit. Par contre, cela nécessiterait la création d'une nouvelle classe visiteur, ainsi que la création de plusieurs nouvelles méthodes. Au final, il est plus simple de garder l'implémentation a l'aide du patron composite qui s'occupe d'itérer à travers ses composantes en appelant la bonne méthode à chaque fois.

Partie 3

1.
   a) L'intention du patron commande est de pouvoir prendre en charge plusieurs suites d'opérations différentes et ainsi permettre une flexibilité au niveau des utilisation du système. Il facilite l'ajout de nouvelles commandes.

2.
   a) Les deux patrons impliqués sont le Singleton et le Médiateur
      i. L'intention du patron Singleton est de s'assurer qu'il n'existe qu'une seule instance de la classe ExecuteurCommande et de permettre son accès global.

      ii. L'intention du patron Médiateur est de simplifier l'interaction entre deux classes afin de promouvoir le faible couplage.

b) L'élément qui confirme la présence du patron Singleton est la méthode getInstance() qui crée et/ou retourne l'instance unique de la classe. L'élément qui confirme la présence du patron Médiateur est la méthode executer (class ProgrammeMachine*, class ElmVisitable*) qui relie ensemble deux classes dans le but de simplifier leur interaction.
c) Ici, l'utilisation du patron Singleton est utilisé se justifie par le fait qu'il n'est pas nécessaire d'avoir plus d'une instance de la classe ExecuteurCommandes. De plus, l'utilisation du patron médiateur est utile pour réduire le couplage entre les classes ProgrammeMachinne et ElmVisitable.

3. Une des utilités du patron Command est qu'ajouter une nouvelle commande est relativement simple, on ajoute simplement la nouvelle sous-classe et ensuite on devra naturellement modifier le code des machines qui utiliseront ce programme en question.