SAMUEL GOULET

Analyse de systèmes

420-3D6-LL, Groupe 00001

TP3

Chaîne de responsabilités (et autres)

Travail présenté à

M. Olivier Lafleur

Département d’informatique

Cégep de Lévis-Lauzon

18 avril 2017

Table des matières

**Chaîne de responsabilité3**

**Interpreteur4**

**Façade5**

**Commande6**

Note : Les diagrammes sont disponibles en format XML (ouvrable sur Draw.io) et en format PNG sur le GitLab, pour rendre la visibilité meilleure.

Chaîne de responsabilité

C:\Users\Administrateur\Desktop\TP3\TP3 - Chaine de responsabilite.png

En gros, la classe GestionnaireDeRecherches (le formulaire dans l’application) appelle la débute la chaîne de responsabilités en appelant la méthode ayant un nom très long dans la classe Recette, qui elle-même appelle cette même méthode dans l’objet suivant, et ainsi de suite jusqu’à la fin de la chaîne. Cette file est construite en appelant la méthode « SetProchain » dans la classe Recette, pour déterminer le prochain chaînon de chaque objet. Le dernier objet pointe vers « null », signifiant la fin de la chaîne.

C:\Users\Administrateur\Desktop\TP3\TP3 - Interpreteur.pngInterpreteur

Bienvenue en Enfer. La classe statique « Interpreteur » commence l’interprétation en appelant la méthode « Décoder » de chaque Expression qui ont été « Registered » préalablement (via la méthode RegisterExpression), à la manière d’une chaîne de responsabilités. La méthode « Décoder » indique que la chaîne de caractères fournie est lisible par l’objet « Expression » donné, et cette méthode appelle les méthodes « Décoder » des « Nodes » (parties d’expressions) qu’elle contient. Lorsque l’expression est déterminée comme étant lisible, leur méthode « Interpreter » est appelée. Celle-ci ne fait que populer un objet « CommandeSquelette », pour ensuite le transformer, via un délégué, en Commande (voir Commande) qui sera retourné.

Façade

C:\Users\Administrateur\Desktop\TP3\TP3 - Facade.png

La façade est un pattern assez simple. La classe « BaseDeDonnees » agit comme façade pour la classe « Inventaire », qui est mal documentée et a une interface peut utilisable (avec des noms de méthode peu explicites).

Commande

C:\Users\Administrateur\Desktop\TP3\TP3 - Commande.png

(Voir l’interpréteur)

Lorsqu’on interprète une chaîne de caractères, l’interpreteur retourne un objet implémentant l’interface « ICommande ». Cet interface force les objets l’implémentant à avoir une méthode « Executer() : void », ce qui permet de standardiser l’appel des méthodes, peu importe lesquelles, que l’interpréteur retourne. Par exemple, la chaîne de caractères « ingredient ajouter <nom> » retourne un objet « CommandeAjouterIngredient » qui possède déjà les informations nécessaires pour créer l’ingrédient en question. Il ne reste qu’à l’exécuter. Cette manière de faire les choses permet d’ailleurs, éventuellement, d’ajouter une méthode « Annuler », ou « AppelerAsync » aux commandes, pour mieux contrôler leur fonctionnement, tout en restant standardisé. Un objet « CommandeVide » est retourné en cas d’erreur, et sa méthode « Executer() » ne fait rien. Ainsi, il n’est pas nécessaire de valider que l’objet retourné n’est pas null.