Devoir Maison de Modélisation et Programmation Objet

Cazorla Axel,
Laviron Pablo
https://github.com/Ethazio/dmMPO
L2 informatique, Groupe C
Faculté des Sciences
Université de Montpellier

22 avril 2022

1 Introduction

Le but de ce devoir est de modéliser puis implémenter en Java le suivi de la gamme de sandwichs d'une boutique de vente de plats à emporter, notamment leur pain, leur sauce et les autres ingrédients les composant.

Pour cela, nous avons utilisé le logiciel «StarUML» pour modéliser le diagramme de classes, «Eclipse» pour la programmation en Java, et nous avons créé une zone de dépôt git sur la plateforme GitHub pour synchroniser notre travail en salle de TP et chez nous.

Au niveau du rythme de travail, nous étions présents lors des TP du vendredi matin pour avancer sur le travail et poser des questions aux professeurs pour éclaircir le sujet. Nous travaillions parallèlement individuellement chez nous, mais aussi ensemble en dehors des cours, à la BU ou l'un chez l'autre.

2 Installation du projet

Afin d'installer le projet et ainsi pouvoir le tester, vous aurez besoin de tous les fichiers *.java* téléchargés sur Moodle ou sur notre dépôt GitHub et d'un IDE (par exemple Eclipse).

Procédure d'installation pour Eclipse :

1. Ouvrez Eclipse dans n'importe quelle *workspace*
2. Cliquez sur «File» → «Open Projects from File System...»
3. Dans «Import source», cliquez sur «Directory...» et sélectionnez le répertoire du projet téléchargé
4. Enfin, cliquez sur «Finish»

Vous pouvez ainsi démarrer le programme par défaut *Main.java* qui illustre les fonctions demandées dans le sujet du devoir en simulant une commande chez une chaîne de magasin de sandwichs connue.

3 Choix de modélisation

Nous avons dû faire certains choix de modélisation que nous allons détailler. Premièrement, nous sommes parti du principe qu'un *Sandwich* comporte **obligatoirement** un *Pain* et une *Sauce*. C'est pour cela que nous

les avons mis dans le constructeur. Par conséquent, nous avons choisi que nos classes Pain et Sauce soient indépendantes des autres aliments, ce qui facilite la compréhension de la généricité du sandwich.

Deuxièmement, nous avons fait le choix de distinguer les aliments des ingrédients : un *Aliment* représente un aliment de la vie de tous les jours, comme une tomate, un oeuf... là où un *Ingredient* représente une quantité d'un certain aliment utilisé pour composer un sandwhich, par exemple une tranche de tomate de 34 grammes. Un même Aliment est utilisé à chaque fois que l'on veut en utiliser comme Ingrédient. C'est pourquoi sur notre modèle UML, nous avons décidé de modéliser cela comme une classe d'association.

La méthode *composer()* permet d'ajouter un aliment au sandwich, en vérifiant si le régime du sandwich le permet, et, le cas contraire, retourne une exception. Ce choix est appuyé par notre expérience client dans la chaine de restauration de sandwichs «Subway» : en effet, lorsque nous composons nos sandwichs, nous le faisons ingrédient par ingrédient, et non tout d'un coup.

De plus, nous n'avons pas inventé les données concernant les ingrédients : nous avons cherché leur valeur calorique sur le site infocalories.fr ainsi que leur quantité dans la documentation de l'enseigne «Subway».

Afin de reproduire une expérience client dans notre restaurant factice numérique, nous avons développé un autre programme contenu dans la méthode *main2()* que vous pouvez essayer et profiter d'une interface pour créer, voir, manger et tester les différentes méthodes sur des sandwichs.

4 Conclusion

TODO