TP Supplémentaire 02 – Programmation de la BAKoffee

L'objectif de ce TP est d'implémenter la BAKoffee du TD 10 – Open Closed Principle

Exercice 0 Initialisation du TP

0.1 Récupération du code source initial

- Comme d'habitude, récupérez les sources depuis le dépôt git
  - git@gitlab-lepuy.iut.uca.fr:TPs-QualiteDeDev/tp-bakoffee.git
- Le code est censé être compilable. Il vous faut :
  - Visual Studio
  - Qt
    - https://www.qt.io/download-qt-installer
  - Qt Visual Studio Tools
    - https://marketplace.visualstudio.com/items?itemName=TheQtCompany.QtVisualStudioTools2022

0.2 Présentation du code source initial

Le code proposé s'articule autour du framework Qt pour l'interface graphique. Toute la partie IHM est déjà réalisée et ne nécessite pas de changement.

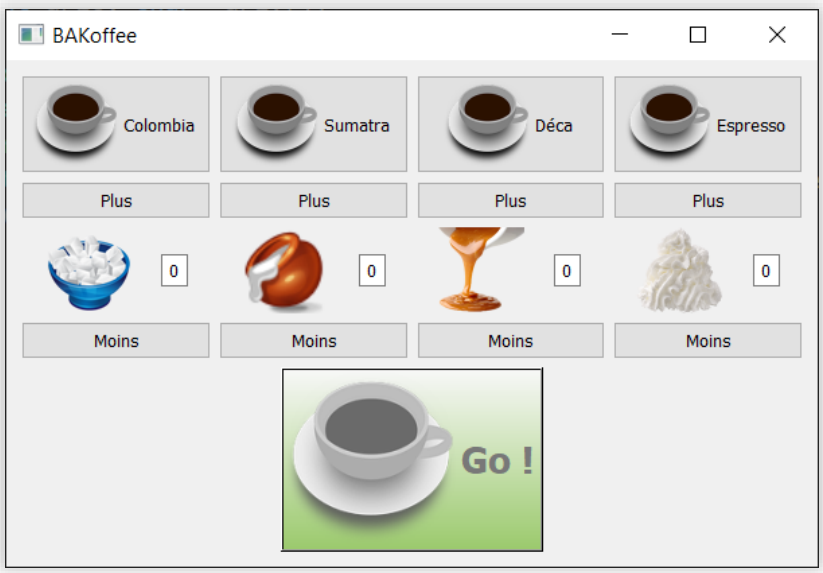

Figure 1 : GUI de BAKoffee

Cette interface est décrite par la classe **`BAKoffee`**. Le bouton principal « Go ! » est associé au slot **`BAKoffee::go()`**. L'objectif de ce slot est de construire un objet **`CBoisson`** pointé par **`BAKoffee::m_pBoisson`** correspondant aux choix de l'utilisateur. Une fois cet objet construit, le slot

demande à l'utilisateur d'insérer le montant nécessaire (via une simple boîte de dialogue de message) puis émet le signal **`BAKoffee::prepare`**.

Ce signal **`BAKoffee::prepare`** est connecté dans la fonction **`main()`** au slot **`QCooker::prepare`** de l'objet **`cook`**. La responsabilité de cet objet est de préparer la commande de l'utilisateur. Cela permet de dissocier très fortement la partie GUI de prise de commande de la partie de préparation concrète de la commande. La préparation de la commande sera simulée en affichant dans une boîte de message la composition exacte de la boisson.

La hiérarchie de classes des boissons concrètes est déjà réalisée et devra être complétée par celle du décorateur.

Exercice 1 A vous de « jouer »…

1.1 Pas café, pas travailler !

- Dans la méthode **`BAKoffee::go()`**, créez, en fonction des choix de l'utilisateur, la base de la boisson (Colombia, Sumatra, Déca ou Espresso)
  - La boisson doit être associée au pointeur **`BAKoffee::m_pBoisson`**.
  - Pour rappel, vous accédez aux objets de l'IHM via l'objet **`BAKoffee::ui`**.
- Vérifiez alors le fonctionnement de l'application :

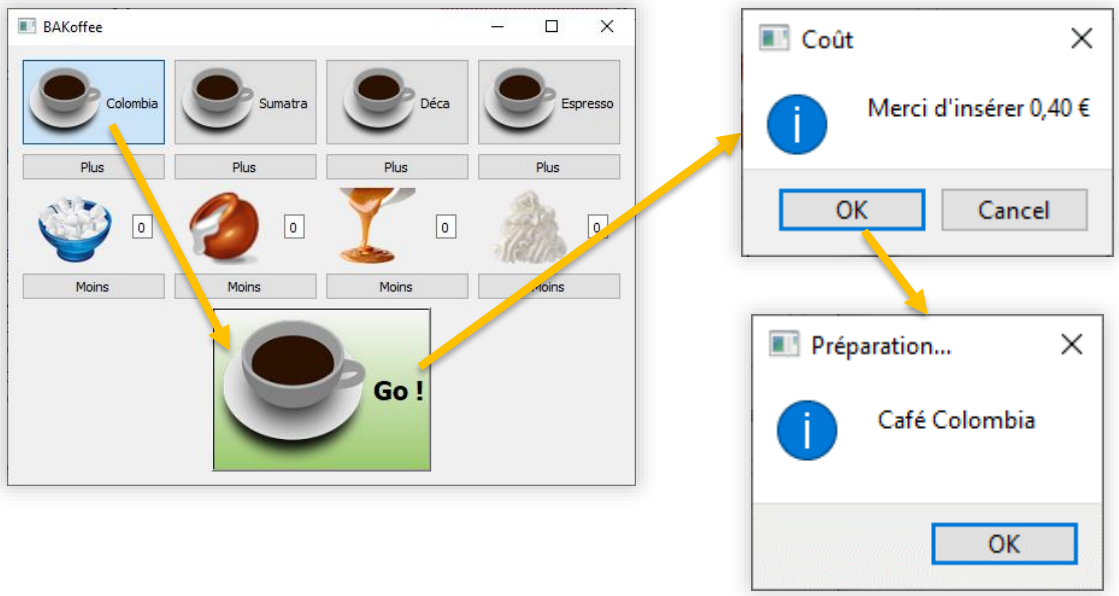

Figure 2 : Fonctionnement type de l'application en fin d'exercice 1.1

1.2 Pas de chantilly, pas d'énergie !

- Mettez en place le pattern décorateur comme vu en TD 08
  - Pour rappel, voici son diagramme de classe :

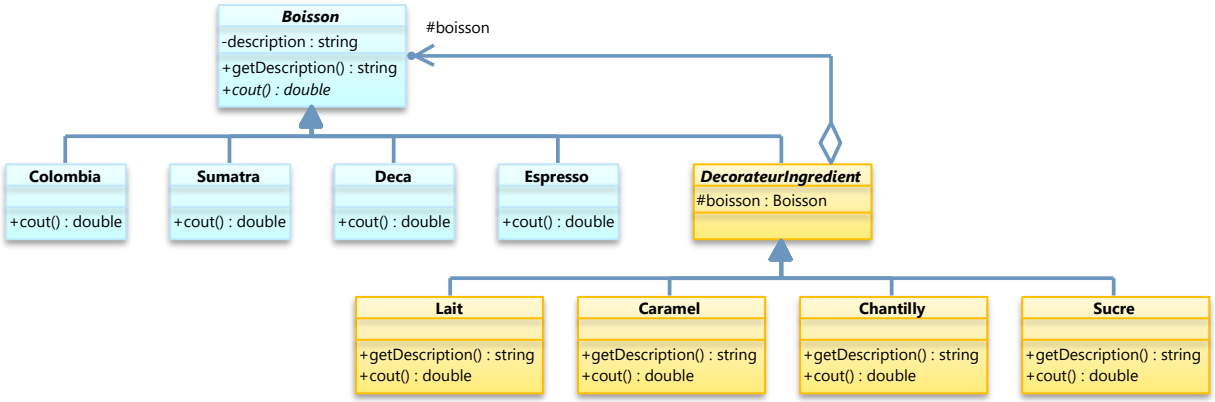

Figure 3 : Diagramme de classe avec pattern décorateur

- Faites en sorte que non seulement le coût de la boisson soit augmenté par les décorateurs ingrédients, mais également que la description « décoré » du nom de l'ingrédient. Ainsi, un café **`Colombia`** décoré par un ingrédient **`Sucre`** doit afficher « *Colombia avec du sucre* ».
- Modifiez la fonction **`BAKoffee::go()`** pour décorer les boissons de base avec la bonne quantité d'ingrédients.
- Vérifiez alors le fonctionnement de l'application :

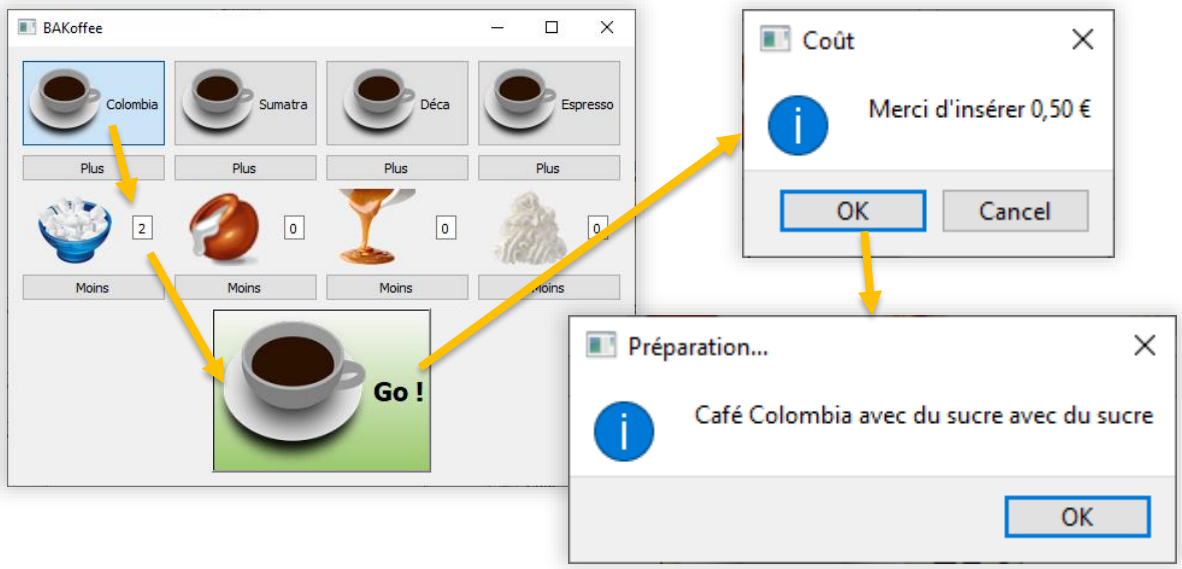

Figure 4 : Fonctionnement type de l'application en fin d'exercice 1.2

1.3 On ne peut pas tromper 1 fois 1000 personnes, euh, si, on peut tromper…

Comme vous pouvez le remarquer, la description du café « décoré » n'est pas terrible : « ***Café Colombia avec du sucre avec du sucre*** ». Nous allons remédier à ce problème. De même, lorsque l'on prépare un café, il est peu conseillé de déposer la crème chantilly en premier et de faire couler le café dessus. Il y a donc un ordre à respecter. Nous allons donc aussi remédier à ce problème.

- A vous de trouver une solution pour que l'objet **`QCooker`** qui doit préparer le café rétablisse l'ordre des ingrédients.
  - Rappel, la préparation de la boisson se fait dans la méthode **`QCooker::prepare`**.
  - Vous être libre de travailler sur le type **`CBoisson::description`** pour avoir un objet de description plus malin qu'une simple chaîne de caractères.
- Vérifiez le fonctionnement de votre application :

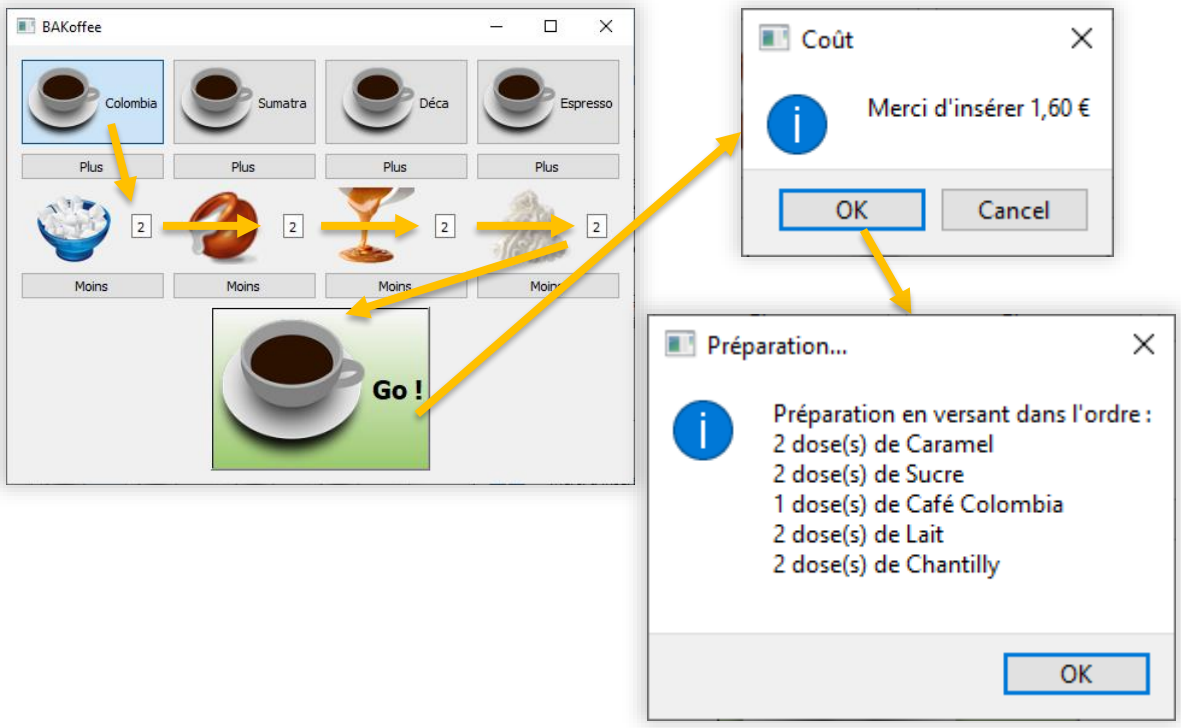

Figure 5 : Fonctionnement type de l'application en fin de programmation