UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie informatique

RAPPORT APP1

Modèles de conception
GIF350

Présenté à
Équipe de formateurs de la session S3

Présenté par
Raphael Bouchard – bour0703
Alexis Guérard – guea0902

Sherbrooke – 10 mai 2023

1. DIAGRAMME DE CLASSES AVEC LES MODÈLES DE CONCEPTION IMPLÉMENTÉS

(Le fichier .png est inclus dans le .zip pour une meilleure clarté)

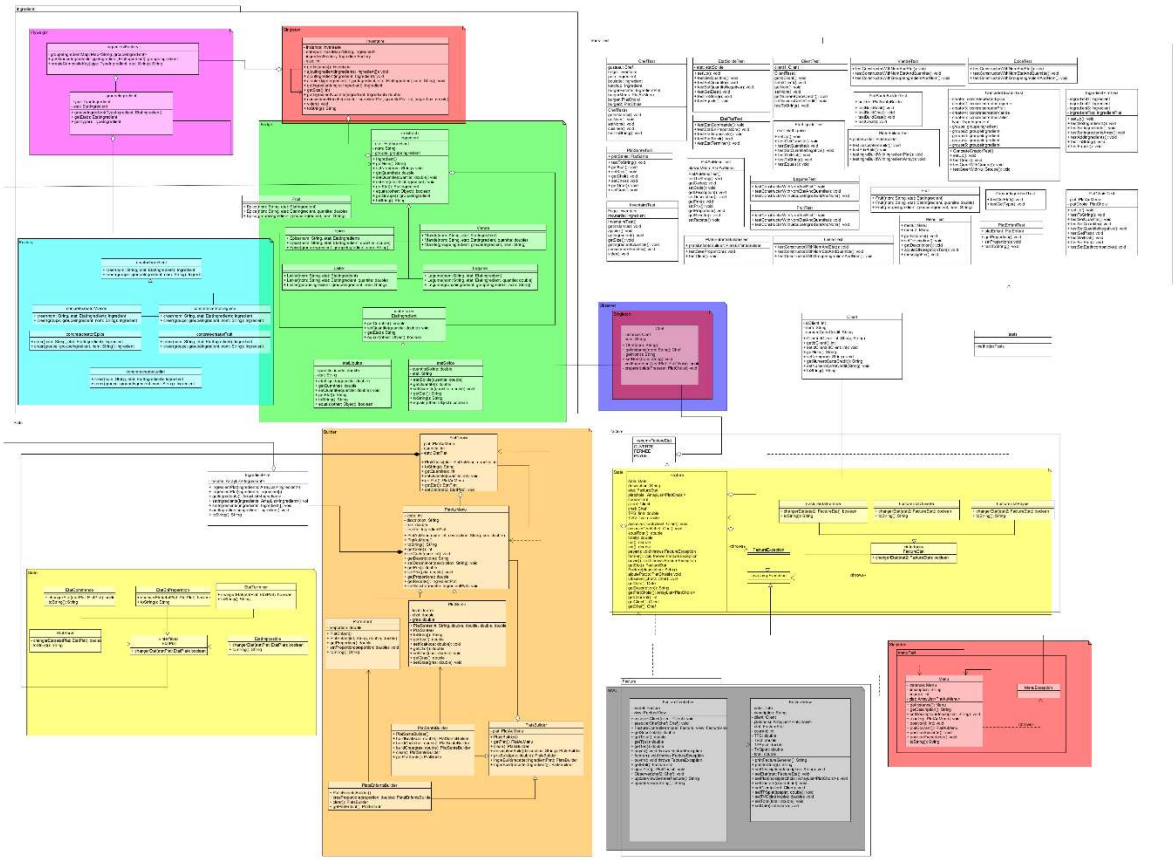

Figure 1 : Diagramme de classe avec les modèles de conception

2. JUSTIFICATION DES MODÈLES DE CONCEPTION

Tableau 1 : Tableau des explications de chaque modèle de conception utilisé

Modèle de conception	Explication de l'utilisation et justification de son utilisation
Flyweight	Le Flyweight est utilisé lorsque nous avons besoin de créer plusieurs objets similaires et que nous voulons économiser de la mémoire en partageant certaines parties communes entre ces objets. Dans notre cas, lors de la création de plusieurs ingrédients cela pouvait prendre beaucoup de mémoire c'est pourquoi nous avons utilisé le Flyweight.
Factory	Le Factory est utilisé lorsque nous voulons déléguez la création des objets à une méthode de fabrication, permettant ainsi une flexibilité et une centralisation de la création d'objet. Pour MenuFact, le Factory a été utilisé pour la création d'ingrédients. Puisqu'il y a plusieurs types d'ingrédients il est plus favorable de les regroupés dans un Factory.
MVC	Le MVC est utilisé pour organiser une application en divisant les responsabilités entre le Modèle, la vue (interface utilisateur) et le Contrôleur (gère les interactions entre le modèle et la vue). Dans l'application MenuFact nous avons utilisé ce modèle de conception pour la facture puisque c'est celle-ci qui affiche.
Builder	Le Builder est utilisé lorsque nous voulons créer des objets complexes étape par étape en masquant les détails complexe de la construction. Dans notre cas, il a été utilisé pour la création des plats puisqu'il faut plusieurs ingrédients ainsi qu'un inventaire ce qui rend le tout beaucoup plus complexe lors de sa création.
Singleton	Le Singleton est utilisé lorsqu'il faut assurer qu'une seule instance d'une classe est créée. Dans notre cas, le Singleton est utilisé pour le chef. Il peut seulement y avoir un chef pour effectuer les fonctions. Ce modèle de conception est aussi utilisé dans le Menu et l'Inventaire.
Observer	L'Observer est utilisé lorsque nous devons implémenter un mécanisme de notification automatique entre objets de sorte que la modification d'un objet entraîne la mise à jour automatique des autres objets dépendants. Dans notre cas, il a été utilisé pour notifier le chef qu'un plat a été ajouté à la facture et qu'il devait commencer à cuisiner.
State	Le State est utilisé lorsqu'il faut gérer plusieurs états d'un objet et que son comportement varie en fonction de son état actuel. Dans le programme MenuFact, ce modèle de conception est utilisé pour s'assurer que la progression des états se font dans le bon ordre par exemple lors de la préparation d'un plat. Ce modèle de conception est aussi utilisé avec la facture.
Bridge	Le Bridge est utilisé lorsque nous voulons séparer une abstraction (interface) d'une ou plusieurs implémentations possibles, permettant ainsi de les modifier indépendamment les unes des autres. Pour MenuFact, nous avons utilisés le Bridge pour les états solide et liquide des ingrédients.

3. AVANTAGE DES MODÈLES DE CONCEPTION

Les modèles de conceptions ont plusieurs avantages dans la programmation. Premièrement, les modèles de conception sont des guides de structure de classe pour résoudre des problèmes connus en programmation. En utilisant ceux-ci, cela permet de standardiser les différents codes pour facilité la maintenance de celui-ci en étant rapidement compris par les programmeurs souhaitant apporter des modifications au code. En effet, on favorise donc la réutilisation du code. Cela permet également d'éviter des erreurs lors de l'implémentation d'un logiciel, puisque les modèles de conception sont bien documentés pour bien les comprendre. Nous recommandons donc d'utiliser cette manière de créer une application pour toutes ces raisons, mais il faut savoir quand et comment bien utiliser les modèles de conception, car ces derniers peuvent augmenter la complexité et nuire à la productivité s'ils ne sont pas nécessaires dans certain cas plus simple. Il faut également s'assurer d'utiliser le bon patron de conception au bon endroit.

4. AVANTAGE DES TESTS UNITAIRES

Les Tests unitaires sont des tests qui vérifie chaque fonction et méthode d'un code individuellement. Le principal avantage d'utiliser les Tests Unitaires est entre autres de s'assurer que toutes les fonctions du code fonctionnent correctement pour avoir un code de la meilleure qualité possible. En effet, ces tests permettent de détecter rapidement les erreurs dans le code. Cela évite donc de perdre du temps dans la correction d'erreurs propagées plus tard dans le projet. Ces tests favorisent également la maintenance du logiciel, puisque lorsqu'on rajoute une nouvelle fonctionnalité, on peut facilement vérifier si cette dernière a engendré des erreurs et on peut savoir dans quelle partie du code l'erreur se fait. Les Tests Unitaires sont également une documentation évolutive, puisqu'elle permet de voir ce que l'application est supposé faire. Les Tests Unitaires permettre donc de s'assurer que tout le code fonctionne correctement et de regarder chaque cas limites. À l'avenir, nous allons utiliser des tests unitaires dans le développement d'application, car ceux-ci améliorent grandement la qualité du code et facilite le débogage. Avec ces tests, on pourra vérifier chaque partie de notre application et regarder plus efficacement les cas limites.