UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et génie informatique

# RAPPORT APP1

Modèles de Conception
GIF350

Présenté à
Eugène Morin

Présenté par
Équipe numéro 8
Étienne Beaulieu – beae0601
Emile Bureau – bure1301

Sherbrooke – 11 mai 2022

# Table des matières

# Diagramme de classes avec les Modèles de Conception

Nouveau diagramme de classes avec les fonctionnalités ajoutées. Pour mieux voir les informations de l'image, un fichier au format PNG a été ajouté au dossier du rapport.

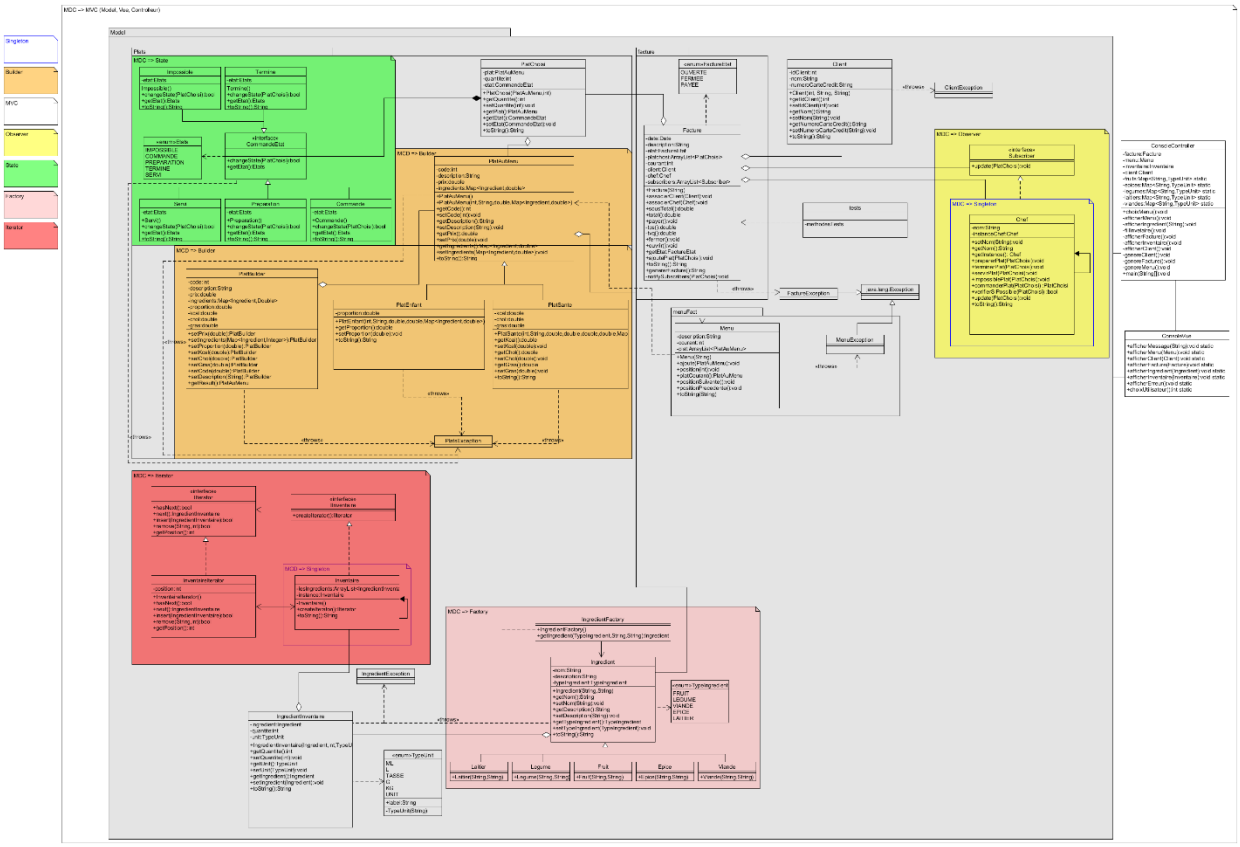

*Figure 1 : Diagramme de classe de menufact2*

# Tableau d'utilisation des Modèles de Conception

*Tableau 1 : Modèles de conception et leurs utilités*

| Modèle de conception | Utilisation |
|---|---|
| **Singleton** | Le singleton est utilisé afin de s'assurer qu'une seule instance d'une même classe est disponible en tout temps. Le singleton est utilisé dans la classe Inventaire pour s'assurer qu'un seul inventaire sera pris en compte tout au long de l'exécution. Aussi, il est utilisé dans la classe Chef afin de reproduire le même besoin qu'avec l'inventaire. |
| **Factory** | Le modèle Factory permet d'utiliser une interface afin de créer des objets, mais il permet aussi aux sous-classes de modifier le type d'objet qui va être créer. Dans ce projet, la Factory a été utilisé afin de créer les différents types d'ingrédients. La Factory utilise une seule interface qui est implémentée par chacune des sous-classes d'ingrédients. |
| **Observer** | L'Observer est très utile, il permet de créer un système d'abonnement afin de notifier des objets lorsqu'un évènement survient. Afin d'améliorer le code, l'Observer est utilisé pour notifier le chef lors de la commande d'un nouveau plat afin de le préparer. Le chef est inscrit à une liste d'abonnements, c'est pourquoi il reçoit une notification. |
| **MVC** | Le modèle de conception MVC sépare le code en trois différentes composantes principales afin de donner une tâche respective à chacune des trois. Dans notre code, les ensembles occupent ces fonctions : le modèle regroupe l'ensemble de la logique reliée à l'informations que l'usager du programme va utiliser. La vue s'occupe principalement de l'affichage, soit la console et les messages d'erreurs. Le contrôleur agit en tant qu'interface entre la vue et le modèle. Il permet de contrôler les interactions entre les données du modèle afin de les afficher aux bons endroits dans la vue. |
| **State** | Le state est un modèle de conception qui permet à un objet de changer son comportement selon son état. Il permet aussi de s'assurer que les transitions entre divers types d'états sont possibles. Dans ce code, il est utilisé lorsque l'on gère les états d'avancement d'un plat, soit l'interface CommandeEtat. |
| **Builder** | Le Builder permet de créer des éléments complexes en plusieurs étapes plus simples. Le Builder est utilisé, dans notre cas, afin de créer les différents plats qui composent le menu. Comme il y a deux sous-classes, le plat santé et enfant, on crée au départ un plat et on crée par la suite le type de plat désiré en fonction des attributs nécessaires. |
| **Iterator** | L'Iterator permet de se promener dans une structure de données sans révéler la structure de données à l'utilisateur. Dans notre cas, l'Iterator a été utilisé afin de parcourir l'inventaire. L'Iterator crée ces propres méthodes peu importe le type de structure de données. |

## Avantage d'avoir utilisé des Modèles de Conception

Les modèles de conception sont des solutions optimisées à des problèmes rencontrés principalement lors de la programmation d'un logiciel. Il existe une multitude de ces solutions, et celles-ci sont standardisées pour la plupart. La standardisation des modèles permet aux programmeurs de comprendre facilement le fonctionnement et les relations entre les diverses classes d'un logiciel. Afin d'éviter des problèmes d'intégration des modèles de conception, il existe de la documentation sur les différents types de modèles. Il faut tout de même utiliser les modèles de conception de façon judicieuse, il n'est pas nécessaire d'en mettre partout dans un code. Parfois, un code sans un modèle de conception est beaucoup plus simple et moins long à implémenter.  C'est pourquoi il est important d'apprendre où il serait avantageux de les utiliser, car ils sont fortement recommandés.

## Avantage d'avoir utilisé les Tests Unitaires

Les tests unitaires sont des tests qui permettent d'assurer le fonctionnement des différentes fonctionnalités d'un code. Ces tests couvrent l'ensemble des cas des méthodes afin d'assurer l'intégrité du projet. Les cas peuvent être, par exemple, des conditions à l'intérieur d'une fonction que l'on n'atteint que dans des cas extrêmes. Les tests unitaires vérifient chacune des possibilités en vérifiant que chaque ligne du code a été testée. Une bonne pratique de programmation est de créer ces tests en même temps que l'on crée une nouvelle fonctionnalité dans le code, ou même avant d'implémenter cette nouvelle fonctionnalité. De cette façon, on s'assure que l'intégration avec le reste du programme s'est bien déroulée, et que le reste du code fonctionne toujours comme prévu. Même si on vérifie chacune des lignes de code avec ces tests, il est important de noter qu'il est la responsabilité du développeur de répertorier chaque cas extrême, car certaines valeurs peuvent fonctionner alors que d'autres non. Les tests unitaires aident les programmeurs à subdiviser le code en plusieurs petits codes. De cette façon, il sera beaucoup plus simple de programmer les tests.