Document de Conception - Application de Gestion de Glaces

Module : Génie Logiciel

Table des matières

1 Introduction

Ce document décrit la conception de l'application de gestion et personnalisation de glaces. L'objectif principal de ce projet est de fournir une architecture modulaire, extensible et robuste pour permettre la création, la modification et la visualisation de glaces personnalisées. Le document couvre les points suivants :

— Les patrons de conception utilisés.
— Les principes de bonne conception appliqués.

2 Objectifs du Projet

L'application vise à :

— Permettre la création de glaces à partir de bases (`GlacePot`, `GlaceCornet`).
— Ajouter dynamiquement des ingrédients (`Vanille`, `Chocolat`, `Fraise`, etc.) et des nappages (`NappageCaramel`, `NappageChocolat`, etc.).
— Gérer les contraintes métier, comme le nombre maximum de boules.
— Maintenir une conception ouverte à l'extension mais fermée à la modification (principe OCP).

3 Patrons de Conception Utilisés

3.1 Patron Décorateur

Le **Patron Décorateur** est au cœur de l'architecture. Il permet d'ajouter dynamiquement des fonctionnalités à une glace sans modifier la classe de base.

Structure du Patron Décorateur :

— **Composant abstrait** : La classe abstraite `Glace` définit l'interface commune pour toutes les glaces.
— **Composant concret** : Les classes `GlacePot` et `GlaceCornet` implémentent la glace de base.
— **Décorateur abstrait** : La classe `Ingredient` étend `Glace` et contient une instance de `Glace`.
— **Décorateurs concrets** : Les classes `Vanille`, `Fraise`, `NappageCaramel`, etc., ajoutent des fonctionnalités spécifiques.

Avantages :

— Extensibilité : Ajout facile de nouveaux ingrédients ou nappages.
— Modularité : Chaque décorateur est indépendant et peut être combiné avec d'autres.

3.2 Principe de Responsabilité Unique (SRP)

Chaque classe remplit une seule responsabilité :

— `GlacePot` et `GlaceCornet` : Gèrent les bases des glaces.
— `Vanille`, `Pistache`, etc. : Ajoutent des boules spécifiques.
— `NappageCaramel`, `NappageChocolat`, etc. : Ajoutent des nappages spécifiques.

3.3 Principe Ouvert/Fermé (OCP)

L'architecture est ouverte à l'extension mais fermée à la modification. Par exemple, pour ajouter un nouvel ingrédient, il suffit de créer une nouvelle classe qui hérite de `Ingredient`, sans modifier les classes existantes.

4 Règles de Bonne Conception

4.1 Respect des Principes SOLID

- **SRP (Single Responsibility Principle)** : Chaque classe a une responsabilité unique.
- **OCP (Open/Closed Principle)** : Les classes sont ouvertes à l'extension mais fermées à la modification.
- **LSP (Liskov Substitution Principle)** : Les instances des sous-classes de `Glace` et `Ingredient` peuvent remplacer leurs super-classes sans altérer le comportement du programme.

4.2 Utilisation d'Abstractions

- Les classes abstraites (`Glace` et `Ingredient`) garantissent une extensibilité.
- Les décorateurs héritent de `Ingredient`, ce qui uniformise leur comportement.

4.3 Encapsulation

- Les attributs sont privés ou protégés.
- Les méthodes publiques offrent un accès contrôlé aux données.

4.4 Testabilité et Validation

- Les règles métier (ex. : pas plus de 4 boules) sont implémentées et testées.
- Les exceptions comme une glace sans boule ou avec un trop grand nombre de boules sont gérées.

5 Outils de Collaboration

5.1 Organisation et Partage des Fichiers

Pour partager les fichiers du projet, nous avons utilisé le service de partage de fichiers proposé par l'université. Ce système nous a permis d'échanger facilement les différentes versions du projet. Chaque membre de l'équipe pouvait récupérer les fichiers nécessaires, apporter ses modifications, et les partager avec le reste du groupe. Cette méthode simple et efficace a assuré une collaboration fluide entre tous les membres.

5.2 Communication via Discord

Discord a été notre principal outil pour communiquer tout au long du projet. Nous avons créé un serveur dédié avec des salons spécifiques pour discuter, poser des questions, et partager

nos idées. Ces discussions en groupe ont permis à tous les membres de rester connectés et informés en temps réel sur les avancées du projet.

Pour les réunions à distance, nous avons utilisé les appels vocaux et vidéo de Discord. Pour les moments en présentiel, nous avons organisé nos sessions de travail à la BU Cortex. Cet environnement calme et adapté au travail collaboratif a été idéal pour se concentrer et avancer efficacement en groupe.

5.3 Répartition des Tâches avec un Diagramme de Gantt

Pour organiser et suivre la progression des tâches, nous avons initialement créé un diagramme de Gantt. Ce diagramme nous a permis de visualiser les différentes étapes du projet, d'attribuer les tâches à chaque membre de l'équipe et de définir des échéances claires. Cependant, au fur et à mesure que le projet avançait, nous avons ajusté ce diagramme en fonction des besoins et des imprévus.

Ces modifications ont permis d'adapter le planning aux évolutions du projet tout en respectant les délais. Cette méthode a renforcé la coordination entre les membres et nous a permis d'avoir une vision globale de l'état d'avancement à tout moment.

6 Conclusion

Le projet illustre une conception modulaire et extensible grâce à l'utilisation de patrons de conception et de bonnes pratiques. L'architecture permet d'ajouter facilement de nouveaux ingrédients ou nappages tout en maintenant une séparation claire des responsabilités.