Groupe 4

Détails et explications des DP utilisés dans notre application

Membres du groupe:

- MBECK MBÔH Lula (Chef de groupe).
- SOUMBOU Patrick Divin.
- NWANTOU Joy.
- KENMEGNE Yoann.

Design Patterns

Dans le contexte du développement logiciel, les Design Patterns (DP) sont des solutions récurrentes à des problèmes communs. Chaque pattern a un objectif spécifique et fournit une solution testée et optimale à un type particulier de problème. Voici des détails et explications pour les Design Patterns que nous utiliserons dans notre application y compris MVC (Model-View-Controller) :

1. **Bridge :**
    - **Explication :** Le Bridge Pattern peut être utilisé pour séparer l'abstraction des différentes parties du système. Dans le contexte du restaurant, cela pourrait impliquer de séparer la gestion des clients de la gestion des commandes.
    - **Application dans le restaurant :** La gestion des clients pourrait être l'abstraction, et les implémentations pourraient inclure des fonctionnalités spécifiques telles que la gestion des réservations et la gestion des clients sans réservation.
2. **Observer :**
    - **Explication :** L'Observer Pattern peut être utilisé pour informer différentes parties du système des changements d'état. Dans le restaurant, cela pourrait être utile pour notifier la cuisine du statut des commandes dès qu'elles sont prises par les chefs de rang.
    - **Application dans le restaurant :** Lorsqu'une commande est prise, les cuisiniers et les serveurs sont notifiés automatiquement pour qu'ils puissent commencer à travailler sur la préparation et le service.
3. **Factory :**
    - **Explication :** Le Factory Pattern peut être utilisé pour créer des instances spécifiques d'objets en fonction des besoins du système. Dans le restaurant, cela pourrait signifier la création d'instances de plats en fonction des commandes des clients.

- **Application dans le restaurant :** Une Factory de plats pourrait créer des instances de différents plats en fonction des choix des clients.

4. **Singleton :**

   - **Explication :** Le Singleton Pattern garantit qu'il n'y a qu'une seule instance d'une classe. Dans le restaurant, cela pourrait être utile pour des classes critiques où une seule instance est nécessaire.

   - **Application dans le restaurant :** La gestion des réservations pourrait être un Singleton pour garantir qu'il n'y a qu'une seule source d'autorité pour la disponibilité des tables.

5. **Strategy :**

   - **Explication :** Le Strategy Pattern permet de définir une famille d'algorithmes, les encapsuler et les rendre interchangeables. Dans le restaurant, cela pourrait être utilisé pour gérer différentes stratégies de traitement des commandes en fonction de la charge de travail ou de la période d'ouverture.

   - **Application dans le restaurant :** Différentes stratégies d'ordonnancement des commandes pourraient être utilisées pendant le déjeuner par rapport au dîner.

6. **MVC (Modèle-Vue-Contrôleur) :**

   - **Explication :** MVC sépare la logique métier, l'interface utilisateur et la gestion des interactions. Dans le restaurant, cela pourrait impliquer de séparer la représentation des données des commandes et des clients (Modèle), l'interface utilisateur qui affiche ces informations (Vue), et le contrôleur qui gère les interactions telles que la prise de commandes.

   - **Application dans le restaurant :** Le Modèle représenterait les données des commandes et des clients, la Vue serait l'interface utilisateur affichant les informations, et le Contrôleur gérerait les interactions telles que la prise de commandes.

En appliquant ces Design Patterns de manière judicieuse, l'application de gestion et de supervision du restaurant peut bénéficier d'une meilleure organisation, d'une flexibilité accrue et d'une facilité de maintenance. L'application précise dépendra des détails spécifiques de la mise en œuvre et des exigences du système.