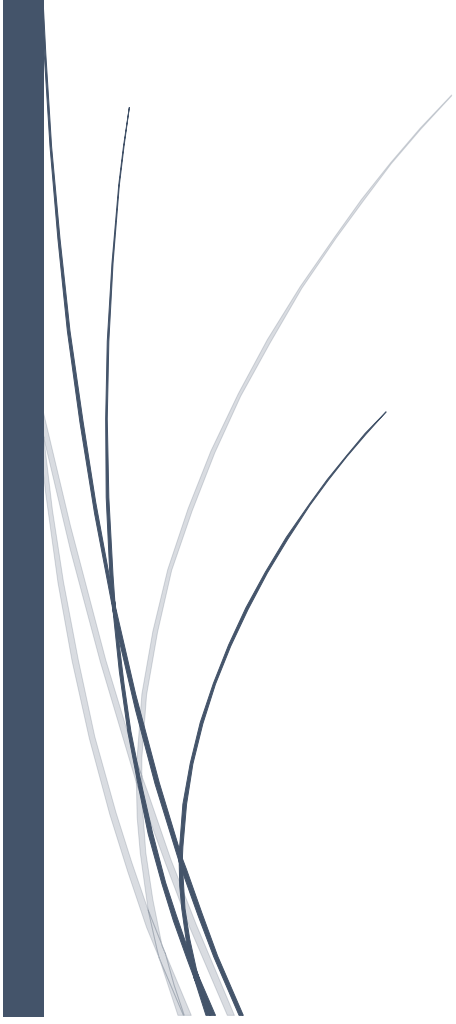
12/12/2024

Rapport des designs patterns utilisés

Projet programmation concurrente



Groupe 6 – X2027
UCAC-ICAM - 2024

1 Contents

1. C’est quoi un design pattern ?

D’après Wikipédia, on entend par « design pattern » ou patron de conception, est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.

En gros, les design pattern permettent de standardiser l’architecture d’un projet et le rendre plus maintenable.

Certains design pattern ont été utilisés pour la réalisation de ce projet, ils sont de différents types notamment : des designs pattern **créatif, architecturaux** et **de comportement**.

2. Designs patterns créatif

Ils se concentrent sur le processus de création d'objets ou sur les problèmes liés à la création d'objets. Ils permettent de rendre un système indépendant de la manière dont ses objets sont créés, composés et représentés.

2.1. Design pattern factory

Pour permettre de séparer la création d’un objet de son implémentation. On peut ainsi définir des fonctions pour créer des instances avec différents paramètres. On peut retrouver ce design pattern pour la création de nos table (pour le restaurant) avec les classes *Table* et *TableFactory* ou encore pour la création de nos groupes de clients.

3. Designs pattern architecturaux

Qui se chargent de définir la structure de l’application en couche en fonction du flux des données.

3.1. Design pattern MVC

Pour Modèle – Vue – Contrôleur, il divise l'application en trois composants principaux :

- **Modèle :** Représente les données et les règles d'affaires. C'est la couche qui interagit avec la base de données ou les API.
- **Vue :** Gère la présentation des données à l'utilisateur. Elle se charge de l'interface graphique.
- **Contrôleur :** Gère les interactions utilisateur et fait le lien entre le modèle et la vue.

Ce pattern améliore la modularité et facilite les tests en isolant les différentes responsabilités.

4. Designs patterns structurels

Les patterns structurels se concentrent sur la composition des classes ou des objets. Ils facilitent la définition de relations flexibles entre les entités pour répondre à des exigences complexes.

4.1 Design pattern Decorator

Le pattern Decorator permet d'ajouter dynamiquement des responsabilités à un objet sans modifier sa structure d'origine. Il favorise l'utilisation de classes pour enrichir le comportement d'un objet à la volée.

Exemple d'utilisation : Dans une application, on peut utiliser ce pattern pour ajouter des fonctionnalités à un composant graphique, comme des bordures ou des couleurs supplémentaires, sans affecter le composant de base.

5. Designs patterns comportementaux

Les patterns comportementaux s'attachent aux interactions entre les objets. Ils facilitent la communication et la gestion des responsabilités entre différents composants.

5.1 Design pattern Observer

Le pattern Observer permet de définir une relation un-à-plusieurs entre des objets, de sorte qu'un changement d'état dans un objet (le sujet) notifie automatiquement tous les objets dépendants (les observateurs).

Exemple d'utilisation : Dans une application de gestion de notifications, lorsqu'un utilisateur publie une mise à jour, tous les abonnés reçoivent une notification. Notre vue « CommonPointView » par exemple observe notre model (pour chaque instance de nos classes héritant de MobileElementModel)