Clean Code Cours / TP 4 Design Pattern par Grégoire de Montety 2022

Essayer de chercher seul les concepts d'un design pattern, puis essayer de l'implémenter en suivant l'énoncé.

Exercice adapter

Un modèle d'adaptateur est un modèle de conception structurelle, qui permet aux classes existantes de fonctionner avec d'autres classes sans modifier leur code source

Un iphone nouvelle génération peut à travers le port Lightning se connecter à un chargeur ou à des écouteurs.

Nous devons créer un adaptateur qui va permettre de connecter des écouteurs en port jack au port lightning.

Créer une interface qui permet aux appareils d'être connectés au téléphone à l'aide de la méthode de connexion.

Coder une classe de chargeur, puis une classe d'écouteurs, connecter vous à l'un puis à l'autre.

Exercice : Pattern Builder (Command)

Blabla utile :
« *Le pattern Builder est utilisé pour la création d'objets complexes dont les différentes parties doivent être créées suivant un certain ordre ou algorithme spécifique. Une classe externe contrôle l'algorithme de construction* »

« dissocier la construction d'un objet de sa représentation, afin que le même processus de construction ait la possibilité de créer des représentations différentes »

ce design pattern sera préconisé dans les situations où au moins :

- l'objet final est imposant, et sa création complexe ;
- beaucoup d'arguments doivent être passés à la construction de l'objet, afin d'avoir un design lisible ;
- certains de ces arguments sont optionnels (par ex. : un bateau n'a pas forcément de capitaine, mais toujours une taille), ou ont plusieurs variations (la taille peut par exemple être passée en mètres ou en pieds).

Créé un package builder

Créer une classe personne avec comme attributs : name, size (double)(cm), title (string) (M, Mr....), birthday (Date)
Créer les setter associés ainsi que le toString()
Créer un builder de personne francaise avec des tailles en cm et les titres francais (Mme, M, Mlle) et les dates au format dd/MM/yyyy
Créer un builder de personne américaine avec des tailles en feet et les titres anglais (Mr, Ms, Miss) et les dates au format MM/dd/yyyy
Penser à mutualiser le code des deux builder dans une classe abstraite
Penser à créer deux enum dans vos builder pour les titre anglais et français

Dans une classe Main, créer deux personnes une américain, l'autre française et afficher ces deux objets.

Ajouter, adapter et utiliser cette méthode pour créer votre américain

```
public AmericanPersonBuilder with(Consumer<AmericanPersonBuilder> builder){
    builder.accept(this);
    return this;
}
```

Expliquer en commentaire pourquoi c'est brillant d'utiliser cette méthode ? Puis expliquer le fonctionnement pas à pas

Exercice : Factory

Le Factory Design Pattern est un modèle de création utilisé pour résoudre le problème de la création d'objets sans avoir à spécifier la classe exacte de l'objet qui serait créé.

Créer une classe VehiculeFactory qui va créer des Bus,Car,Truck de couleurs différentes
Créer une enum de type de vehicule et une autre pour les couleurs

Exercice : Strategy

Implémenter avec le pattern stratégie un calcul de coût de livraison de Pizza.
Si la pizza est a emportée, le cout et de -10%
Si la pizza est a livrer dans Paris, le cout et de 5%
Si la pizza est a livrer à l'extérieur de Paris, le cout et de 10%

Créer une enum avec les types de livraison et une interface avec une méthode de calcul de prix de livraison

Exercice : Chaine de responsabilité

Implémenter avec un pattern de chaine de responsabilité le nettoyage d'un costume.
-Créer une classe abstraite SuitWashStep avec un attribut avec l'étape du nettoyage
-une méthode concrète andThen avec en param un SuitWashStep pour chainer la prochaine opération
-une méthode abstract applyTo avec en paramètre un Suit

Invertion of Control (IoC) (facultatif)

https://gfx.developpez.com/tutoriel/java/ioc/

Un conteneur d'IoC peut être identifié par trois caractéristiques majeures : il contient des objets, il contrôle la création de ces objets et il résout les dépendances entre les objets.
Par sa nature le conteneur gère le cycle de vie de ces objets.
Vous n'avez pas à créer les instances ni à libérer les ressources.