Examen Java

Simulation de commande d'un restaurant

Partie 1: POO

Un restaurant propose plusieurs plats, composé de 1 à plusieurs ingrédients, avec un à plusieurs allergènes et un prix unitaire.

Un client, qui a 0 à plusieurs allergies, avec une somme d'argent disponible va essayer de passer une commande d'un à plusieurs plats.

Une vérification sera faite sur cette commande, si elle ne contient aucun plat avec des allergènes dont le client est allergique, et que le client possède assez d'argent pour faire une commande.

Lorsque la commande est validé ou non via une chaîne de vérification, alors, le statut de la commande va passer de vide à un message, et le client sera mis à jour de cette valeur une propriété sur ce client comme "maCommande" qui est une chaîne de texte contenant, soit le motif de refus, soit le texte: "Validé"

Chacune de ces commandes passées, valide ou non, sera notée dans une "archive" accessible globalement, idéalement, le contenu de la commande sera mise au format "String" pour être le plus lisible possible.

Vous devrez utiliser les principes de la **programmation orientée objet** ainsi que les **design patterns** suivants :

- **Singleton**
- **Builder**
- **Observer**
- **Chain of Responsibility**

Pour vous aider, voici une liste des classes possibles et non exhaustive.

- Plat contenant un builder
- Client contenant un builder
- Restaurant
- Commande
- ValidationAllergene, ValidationSoldeClient (qui sera mise en Chaîne de Responsabilité comme un middleware) -> Attention, je vous demande une variante de celle vu en cours.
- Archive & SingletonArchive

Voici un repository avec une ébauche de structure: Lien

Partie 2: Algo pur

Générateur de Hashtags

Le service marketing passe beaucoup trop de temps à taper des hashtags. Aidons-les avec notre propre **Générateur de Hashtags** ! Ecrivez la fonction.

Voici les règles :

- Le hashtag doit commencer par un **#**.
- Chaque mot doit avoir sa **première lettre en majuscule**, et le reste en minuscules.
- Si le résultat final dépasse **140 caractères**, il faut retourner **false**.
- Si l'entrée ou le résultat est une **chaîne vide**, il faut retourner **false**.

Exemples :

- `" Hello there thanks for trying my Kata"` => `"#HelloThereThanksForTryingMyKata"`
- `" Hello World "` => `"#HelloWorld"`
- `""` => `false`

Barème

- Poo: 14 points
  - Ce qui est attendu, fonctionne comme attendu: 6 points
  - Qualité et propreté du code: 4 points
  - Usage pertinent des design pattern et des interfaces: 4 points
- Algo: 6 points
  - Le résultat est bon: 4 points
  - Qualité de code: 2 points

ALGO