HEI | PROG3
Évaluation K1 du 29 janvier 2026

Durée : 1h

Documents autorisés - Internet autorisé - IA générative autorisée. Seule la communication entre deux étudiants (physiquement ou par messagerie instantanée) est interdite, intentionnelle ou involontairement, sous peine d'être exclue définitivement du cours.

Rappel du contexte : Notre application est capable de gérer la prise de commande de plusieurs plats, où chaque plat est composé d'un ou plusieurs ingrédients définis par une quantité et d'une unité. Chaque ingrédient a un stock, calculé à travers le mouvement (entrée et sortie) du stock de l'ingrédient.

Attention ! Avant de commencer le travail, veillez à avoir un commit de votre projet car après l'évaluation, nous allons revenir à l'état initial et tout le travail à faire aujourd'hui DOIT disparaître.

Rappel des méthodes principales pré-requises dans la classe DataRetriever :

- `Order findOrderByReference(String reference);`
- `Order saveOrder(Order orderToSave);`
- `Ingredient saveIngredient(Ingredient ingredientToSave);`
- `Ingredient findIngredientById(Integer idIngredient);`
- `Dish findDishById(Integer idDish);`
- `Dish saveDish(Dish dishToSave);`

Ces méthodes doivent être implémentées correctement afin de pouvoir terminer le travail qui va suivre, **sauf pour le cas spécifique (*) évoqué à la fin.**

Travail à faire : Gérer les contraintes d'espace dans un restaurant

Allons plus loin dans notre contexte : jusqu'à maintenant, nous n'avons considéré que des ingrédients, des plats et des commandes. Tous ces éléments font partie du domaine de la restauration, et les restaurants ont une contrainte critique : l'espace.

Autrement dit, lorsqu'un client arrive, il s'installe sur une table, avant de pouvoir effectuer une commande. Si toutes les tables sont déjà prises pour une date et heure donnée, alors on ne peut plus prendre des commandes car dorénavant, toutes les commandes doivent être associées à une table.
Voici les éléments caractéristiques d'une table de restaurant au niveau de notre SI :

- Un identifiant unique, de type nombre entier, obligatoire;
- Un numéro de table, de type nombre entier, obligatoire;

Les modifications attendues au niveau d'une commande, toujours au niveau de notre SI, est donc l'ajout des attributs obligatoires suivants :

- L'identifiant d'une table de restaurant;
- La date d'installation d'un client à la table, de type date et heure sans fuseau horaire;
- La date de départ d'un client de la table, de type date et heure sans fuseau horaire;

Autrement dit, lors de la création d'une commande, il faut obligatoirement spécifier la table à laquelle est associée la commande. Si jamais la table spécifiée n'est pas disponible à l'instant de la création de la commande, alors l'application doit lever une exception informée que la table fournie n'est pas disponible.

1) Effectuez les modifications nécessaires pour implémenter cette nouvelle fonctionnalité. Si la table spécifiée lors d'une commande n'est pas disponible, alors l'application retourne juste une exception avec un message disant que la table n'est pas fournie.

2) Améliorez votre implémentation, de sorte à ce que lorsque l'application retourne une exception lorsque la table spécifiée lors d'une commande n'est pas disponible, le message d'erreur indique si d'autres tables sont disponibles et si c'est le cas, alors les proposer dans le message d'erreur. Par exemple, supposons que nous avons trois tables de restaurant représentées respectivement par le numéro 1, numéro 2 et numéro. Si nous essayons de spécifier la table numéro 1 lors de la création de la commande alors que celle-ci est déjà utilisée, et que la table numéro 2 et la table numéro 3 sont encore libres, alors l'exception doit contenir un message indiquant que les tables numéro 2 et numéro 3 sont actuellement libres mais pas la numéro 1.

   Autre exemple, si jamais les trois tables ne sont pas disponibles lors de la création de la commande, alors indiquez dans le message d'erreur qu'aucune table n'est disponible.

Notez que toute ces implémentations doivent être effectuées en rapport avec la méthode `Order saveOrder(Order orderToSave);`

Cas spécifique (*) : obtention de 66% de la note maximale uniquement

Si vous rencontrez encore des problèmes avec la méthode `Order saveOrder(Order orderToSave);` il vous reste une option pour réussir l'évaluation. Vous allez effectuer les

modifications nécessaires au niveau de la base de données (les commandes toujours obligatoirement reliées à une table de restaurant qui peut être indisponible pour une date et heure donnée), et y enregistrer directement les données de tests.

Le résultat attendu par l'évaluation dans ce cas précis est donc une méthode qui retourne la liste des tables disponibles à une date et heure donnée. Le principe reste pareil que le cas initial, notamment l'intérêt reste centré sur les commandes, toutefois, nous allons juste nous attarder sur la disponibilité des tables.

Remarques importantes :

Vous pouvez procéder en style procédural **exceptionnellement** pour cette évaluation et non obligatoirement en orienté objet. Toutefois, si vous respectez la structure suivante, vous avez la possibilité d'avoir un **BONUS** :