Projet de programmation Java : Gestion d'un gala

A rendre le 03/01/2022

Une école de commerce souhaite informatiser une partie de la gestion de son gala de fin d'année. Elle vous demande de réaliser une application permettant de gérer les inscriptions et les réservations de tables du dîner. Ce projet est à faire par groupe de 3 étudiants, la création des groupes est de la responsabilité des étudiants.

I. Quelques informations sur l'organisation

Inscription

L'inscription au gala est obligatoire pour avoir accès à la salle ainsi qu'à toute commodité liée au gala. L'application devra permettre aux étudiants de chaque promotion ainsi qu'au personnel de d'école de s'inscrire après identification. Les étudiants devront s'identifier avec leur numéro d'étudiant et les membres du personnel avec un numéro qui leur est attribué à leur arrivée dans l'école.

Réservation des tables

Chaque table du gala possède 8 places. Le comité a décidé de mettre en place trois niveaux de tarifs pour une place à une table de dîner. Le tarif 1 étant le moins élevé est réservé aux étudiants de dernière année, la cinquième année. Le tarif 2 est réservé aux autres étudiants tandis que le tarif 3 sera appliqué au personnel et aux accompagnants.

Personnel

Dix tables sont réservées uniquement pour le personnel. Chaque membre du personnel pourra, au moment de sa réservation, réserver une place supplémentaire pour un accompagnant. Le nombre de places étant suffisant, les réservations seront enregistrées immédiatement. Au moment de la réservation, la personne pourra demander à consulter la composition de toutes les tables réservées au personnel. On affichera alors les noms des membres déjà affectés à chaque table. Si un membre vient accompagné, son nom apparaîtra suivi de la chaîne de caractères « + accompagnant ». La personne pourra alors choisir un numéro de table puis elle précisera le nombre de places qu'elle souhaite réserver (1 ou 2). Elle peut aussi décider de ne pas consulter la composition des tables et choisir directement le nombre de places qu'elle souhaite. Une table lui sera affectée lors de sa réservation. Une fois la réservation effectuée, le montant correspondant aux places sera affiché (on ne s'occupe pas de la gestion du paiement).

Au plus tard 10 jours avant la date du gala, un membre du personnel pourra annuler sa réservation.

Étudiants

Quinze tables sont réservées pour l'ensemble des promotions de l'école. Chaque étudiant pouvant réserver des places pour des accompagnants et le nombre de places au dîner du gala étant limité, le comité a introduit une notion de priorité dans les demandes de réservations. On donnera la priorité aux étudiants de dernière année.

Un étudiant de dernière année pourra réserver jusqu'à 3 places pour 3 accompagnants alors qu'un étudiant d'une année inférieure ne pourra réserver qu'une seule place pour un accompagnant en plus de la sienne.

Les demandes de réservation seront enregistrées suivant leur priorité. Un mois avant le jour du gala, on sélectionnera dans cette file d'attente les étudiants dont on accepte la demande de réservation. Ces étudiants seront placés en attente de confirmation de réservation.

Il devront donc confirmer leur réservation. Au moment de cette confirmation, ils pourront demander à consulter la composition de toutes les tables affectées aux étudiants et choisir un numéro de table sinon une table leur sera attribuée directement. Une réservation sera alors enregistrée et l'étudiant ne sera plus en attente de confirmation. Le système affichera le montant de la réservation.

Après avoir accepté les demandes de réservation de certains étudiants, les étudiants n'ayant pas été sélectionnés faute de place resteront dans la file d'attente en attendant qu'un étudiant annule sa réservation et libère ainsi une ou plusieurs places. Un étudiant peut annuler sa réservation au plus tard 10 jours avant la date du gala.

Les étudiants pourront toujours effectuer des demandes de réservation, ils seront placés dans la file d'attente suivant leur priorité.

II. Architecture du projet

Les objets de base manipulés dans l'application sont les étudiants, les personnels, les tables ainsi que les réservations. Vous devrez créer les classes correspondantes et en ajouter d'autres, si vous le jugez nécessaire.

Un **étudiant** est caractérisé par un numéro d'étudiant de type `int`, un nom, un prénom, un numéro de téléphone et un mail, chacun étant de type `String`. Chaque étudiant est dans une année de formation (de 1 à 5).

Un **personnel** est caractérisé par un numéro de type `int`, un nom, un prénom, un numéro de téléphone et un mail chacun étant de type `String`.

Une **table** est caractérisée par un numéro de table de type `int` et un nombre de places libres de type `int`. Rappelons que toutes les tables sont des tables 8 places. Nous souhaitons pouvoir enregistrer le nom des participants affectés à chaque table lors des réservations.

Une **réservation** est caractérisée par une date de type `LocalDate`, un numéro de table de type `int`, un nombre de places de type `int` et un montant de type `double`.

On affectera une réservation à l'étudiant ou au personnel qui l'aura créée.

Vous utiliserez l'héritage pour mutualiser les caractéristiques communes et les comportements communs entre les objets de type `Etudiant` et de type `Personnel`.

Vous devrez ajouter à chaque classe le ou les constructeur(s) et les méthodes que vous jugerez nécessaires en veillant à identifier les utilisations pouvant entraîner des erreurs. Vous jugerez de la nécessité de redéfinir les méthodes héritées de la classe `Object`.

Architecture Modèle Vue Contrôleur :

Créer quatre classes : `Gala`, `Ihm`, `Controleur` et `Main`.

La classe `Gala`

Cette classe centralise la gestion des objets du modèle. Vous devrez choisir les conteneurs d'objets de l'API Java adaptées à la gestion des étudiants, des personnels et des tables réservées. Il faudra en particulier pouvoir retrouver efficacement un étudiant ou un personnel à partir de son identifiant, enregistrer les étudiants inscrits, les personnels inscrits, les étudiants ayant fait des demandes de réservation et placées en attente suivant la priorité indiquée précédemment (vous pourrez vous documenter sur la classe `PriorityQueue<E>` fournie dans la bibliothèque des collections java), les étudiants dont la demande de réservation aura été acceptée, les tables réservées aux étudiants et celles réservées au personnel. On souhaite pouvoir retrouver une table à partir de son numéro.

Dans la classe `Gala`, on définira les constantes suivantes :

- les constantes correspondant aux 3 tarifs d'une place : tarif 1 : 10 euros, tarif 2 : 15 euros et tarif 3 : 20 euros.
- les constantes correspondant au nombre de tables affectées aux étudiants (15 tables) et au nombre de tables affectées au personnel (10 tables).

Le constructeur de la classe `Gala` prendra en paramètre une date de type `LocalDate` correspondant à la date du gala.

La liste de tous les étudiants pouvant s'inscrire au gala est fournie sous forme d'un fichier «`etudiants.txt`». Celle du personnel est fournie sous la forme d'un fichier «`personnel.txt`».

Le constructeur de la classe `Gala` lit ces fichiers et alimente les objets prévus pour la gestion initiale des étudiants et du personnel.

Il crée également les tables du dîner pour alimenter les collections prévues pour la gestion des tables réservées au personnel (numérotées de 1 à 10) et des tables réservées aux étudiants (numérotées de 11 à 25).

La classe `Ihm`

Cette classe gère toutes les interactions avec l'utilisateur. Elle se chargera d'afficher les informations, de récupérer les données saisies par l'utilisateur à l'aide de la classe `Scanner` et de les transmettre au contrôleur.

La classe `Controleur`

Cette classe traite les demandes de l'utilisateur, agit sur le modèle ou l'interroge en appelant les services d'un objet de type `Gala`.

Le contrôleur connaît donc un objet de type `Ihm` et un objet de type `Gala`.

Le constructeur de la classe `Controleur` prendra en paramètre une date de type `LocalDate` correspondant à la date du gala. C'est le contrôleur qui va créer l'objet de type `Ihm` et créer un objet de type `Gala` lors du premier lancement de l'application.

Cette classe pourra enregistrer l'identifiant de l'utilisateur connecté pour s'en servir lors des

interactions suivantes et en particulier savoir s'il s'agit d'un étudiant ou d'un membre du personnel.

La classe `Main`

Cette classe contient la méthode `main` qui crée un objet de type `Controleur` en passant une date de gala en paramètre.

III. Sauvegarde des données

Nous souhaitons conserver toutes les données du gala de manière à les retrouver à chaque nouveau lancement de l'application. Pour ce faire, lorsqu'un utilisateur quitte l'application, l'objet de type `Gala` contenant toutes les informations devra être sauvegardé par le contrôleur dans un fichier qui sera chargé à chaque lancement de l'application. Le constructeur du contrôleur devra donc créer un objet de type `Gala` lors du premier appel de l'application mais pour les autres appels, il devra charger l'objet `Gala` sauvegardé dans le fichier.

Pour réaliser cela, vous devez utiliser l'interface `IServiceStockage` ainsi que la classe `ServiceStockage` implémentant cette interface qui vous sont fournies et dont la documentation se trouve en Annexe.

IV. Déroulement des inscriptions et des réservations de tables

Lors du lancement de l'application, vous afficherez l'état de l'objet `Gala`, ce qui doit permettre de consulter le contenu des conteneurs d'objets qui le composent.

Ensuite, l'utilisateur doit préciser s'il est étudiant ou personnel puis s'identifier avec son numéro. Si l'identification échoue, le système demande de saisir le numéro de nouveau.

Si l'utilisateur n'est pas inscrit, le système lui propose de s'inscrire ou de quitter l'application. Une fois inscrit, le système affiche le menu donnant accès à la gestion des places du dîner.

Si l'utilisateur est déjà inscrit, le système affiche directement le menu donnant accès à la gestion des places du dîner.

Ce menu proposera 1 – Gérer les places du dîner 2 – Se désinscrire 3 – Quitter

Quand l'utilisateur choisit de gérer les places du dîner, les étapes suivantes diffèrent suivant que l'utilisateur est un étudiant ou un membre du personnel.

Lorsqu'un utilisateur se désinscrit, il faut supprimer les demandes de réservations ou les réservations effectives qui le concernent avant de le désinscrire.

Gestion des places du dîner pour le personnel

Le système affiche le nombre de places que l'utilisateur peut réserver et demande s'il souhaite consulter le plan des tables ou pas.

Si l'utilisateur accepte, le plan des tables s'affiche et l'utilisateur doit choisir un numéro de table. Puis il choisit un nombre de places. Si la table ne contient pas assez de places libres, on affiche un message et on revient au sous-menu.

Si l'utilisateur ne consulte pas le plan des tables, il précise le nombre de places qu'il souhaite et une table libre lui sera attribuée.

Une fois la réservation enregistrée, on quitte l'application.

Si l'utilisateur a déjà fait une réservation, le système affiche le nombre de places réservées et le numéro de table. Dans ce cas, il ne peut plus faire de réservation et ne peut pas non plus modifier sa réservation.

Gestion des places du dîner pour les étudiants

Les étudiants doivent commencer par faire une demande de réservation qui sera placée en attente suivant la priorité définie précédemment. Le système affiche le nombre de places que l'étudiant peut réserver avant de demander combien de places il souhaite réserver. Si le nombre demandé est inférieur ou égal au nombre de places autorisé, la demande sera enregistrée.

Si l'étudiant a déjà fait une demande de réservation, le système affiche le nombre de places demandées. Dans ce cas, il ne peut plus faire de demande de réservation et ne peut pas non plus modifier sa demande.

Dans le mois avant la date du gala, à chaque lancement de l'application, la liste des étudiants dont la demande de réservation aura été acceptée sera mise à jour. Elle dépend du nombre total de places mais aussi des éventuelles désinscriptions d'étudiants.

A partir de ce moment-là, un étudiant ayant fait une demande de réservation et ayant été accepté verra apparaître un autre sous-menu lui demandant de confirmer sa demande de réservation en lui rappelant le nombre de places demandées. Comme pour le personnel, l'étudiant pourra demander à consulter le plan des tables réservées aux étudiants pour choisir son numéro de table. Une fois la confirmation effectuée, on quitte l'application.

Une fois qu'un étudiant a confirmé sa réservation, lorsqu'il choisit « Gérer les places du dîner », le système lui affiche le nombre de places qu'il a réservées et il ne peut plus faire de modifications.

V. Remise du travail

Le projet est à réaliser par trinôme. Il est demandé un rapport au format `pdf`. Ce rapport doit présenter vos choix d'implémentation et vos choix en terme de gestion des erreurs.

Vous préciserez également la répartition du travail au sein du groupe.

Les classes devront être commentées de manière à pouvoir générer la Javadoc de votre projet.

Déposer les fichiers sources de votre projet ainsi que le rapport sous la forme d'une archive au format `.zip`. Le dépôt se fera dans le cours en ligne en cliquant sur **Dépôt du projet Java** correspondant à votre groupe de TP. Date limite du dépôt : lundi 3 Janvier 2022.

Les soutenances du projet seront organisées le mardi 4 Janvier.

Pour l'organisation des soutenances, on vous demande d'inscrire les noms des étudiants de chaque trinôme dans le wiki correspondant à votre groupe **avant le 10 Décembre**.

Cas des trinômes sur plusieurs groupes : Si vous avez constitué un trinôme avec des étudiants d'autres groupes que le votre, inscrivez-vous dans le wiki d'un des groupes. Les enseignants feront ensuite une répartition définitive pour rééquilibrer les groupes. Pour le dépôt du projet, vous déposerez votre projet dans le groupe qui aura été affecté à votre trinôme.

Annexe

Sérialisation

- Pour sauvegarder un objet de type `Gala`, il faut que la classe `Gala` implémente l'interface `Serializable` et que toutes les classes des objets utilisés dans celle-ci l'implémentent également. En particulier, vous ne pourrez pas utiliser de classe interne anonyme car celle-ci ne pourra pas être déclarée `Serialisable`. Vous pouvez toujours utiliser une classe interne si besoin mais il faut la nommer, elle ne pourra pas être anonyme.

Interface IServiceStockage

Method Summary

Modifier and Type	Method	Description
`java.lang.Object`	`charger()`	Lit un objet à partir du flux d'entrée
`void`	`enregistrer(java.lang.Object object)`	Écrit l'objet passé en paramètre dans un flux de sortie

Method Details

enregistrer

`void enregistrer(java.lang.Object object) throws java.io.IOException`

Écrit l'objet passé en paramètre dans un flux de sortie

Parameters: `object` -

Throws: `java.io.IOException` - Erreur liée aux entrées/sorties

charger

`java.lang.Object charger() throws java.io.IOException, java.lang.ClassNotFoundException`

Lit un objet à partir du flux d'entrée

Returns: l'objet lu

Throws:

`java.io.IOException` - Erreur liée aux entrées/sorties

`java.lang.ClassNotFoundException` - La classe d'un objet sérialisé ne peut être trouvée.

Class ServiceStockage

```
public class ServiceStockage
extends java.lang.Object
implements modele.IServiceStockage
```

Constructor Summary

Constructors

Constructor	Description
`ServiceStockage()`	Crée un objet de type ServiceStockage qui va accéder à un fichier "gala.ser" en lecture et en écriture.

Method Summary

Modifier and Type	Method	Description
`java.lang.Object`	`charger()`	Lit un objet dans le fichier à condition que la classe de l'objet implémente l'interface Serializable
`void`	`enregistrer(java.lang.Object object)`	Ecrit l'objet passé en paramètre dans le fichier à condition que la classe de l'objet implémente l'interface Serializable

Constructor Details

ServiceStockage

`public ServiceStockage() throws java.io.IOException`

Crée un objet de type ServiceStockage qui va accéder à un fichier "gala.ser" en lecture et en écriture. Le fichier "gala.ser" est créé s'il n'existe.

Throws: `java.io.IOException` - Erreur liée aux entrées/sorties

Method Details

enregistrer

`public void enregistrer(java.lang.Object object) throws java.io.IOException`

Écrit l'objet passé en paramètre dans le fichier à condition que la classe de l'objet implémente l'interface Serializable

Specified by: `enregistrer` in interface `IServiceStockage`

Parameters: `object` -

Throws: `java.io.IOException` - Erreur liée aux entrées/sorties

charger

`public java.lang.Object charger() throws java.io.IOException, java.lang.ClassNotFoundException`

Lit un objet dans le fichier à condition que la classe de l'objet implémente l'interface Serializable

Specified by:

`charger` in interface `IServiceStockage`

Throws:

`java.io.IOException` - Erreur liée aux entrées/sorties

`java.lang.ClassNotFoundException` - La classe d'un objet sérialisé ne peut être trouvée.