Projet de programmation Java : Gestion de spectacles

A rendre le 05/01/2020

Il est vivement recommandé de lire le sujet intégralement avant de commencer.

Le projet doit vous permettre de vous familiariser avec la bibliothèque des collections de l'API Java. Vous ne devez donc pas utiliser de tableaux pour réaliser ce projet.

L'objectif de ce projet est de gérer la programmation et la billetterie de la maison des Arts et de la Culture de la ville de Tarascon. Cette maison gère les salles de spectacle. Les spectacles programmés sont de deux types : film et pièce de théâtre.

Un spectacle est caractérisé par un numéro, un titre et un ensemble de noms d'interprètes.

Un film est un spectacle défini en plus par le nom de son réalisateur et sa durée exprimée en minutes. Une pièce de théâtre est un spectacle caractérisé en plus par le nom de son metteur en scène et un nombre d'entractes.

La maison des Arts et de la Culture propose deux types de salles, les salles standards dans lesquelles sont projetés les films et les salles de théâtre pour les pièces de théâtre.

Une salle standard porte un numéro, un nom, est caractérisée par un nombre de places et un tarif de place. Dans ce type de salles, les places peuvent être vendues à prix réduit correspondant à 60% du tarif normal (pour les familles nombreuses, les étudiants, les chômeurs...). Une salle enregistre aussi ses créneaux d'occupation. Un créneau est caractérisé par un jour de la semaine (entier dans l'intervalle [1. . .7]), un horaire de début et un horaire de fin.

Une salle de théâtre a les mêmes caractéristiques qu'une salle standard. Elle dispose en plus de places à tarif supérieur, nommées *fauteuils*. On enregistrera donc pour ces salles le nombre de fauteuils et le tarif des fauteuils. Aucune réduction ne peut être appliquée dans ces salles.

A chaque spectacle est associé un ensemble de séances correspondant aux séances planifiées sur une semaine.

Une séance est donc caractérisée un créneau et un nombre de places vendues à tarif normal.

Une séance de film est une séance qui est en plus caractérisée par une salle standard et un nombre de places vendues à tarif réduit.

Une séance qui correspond à une pièce de théâtre est une séance qui est en plus caractérisée par une salle de théâtre et un nombre de places de type fauteuil vendues.

Pour un film, plusieurs séances peuvent être programmées par jour à condition que les salles soient disponibles sur les créneaux demandés. En revanche, une seule séance par jour est programmée s'il s'agit d'une pièce de théâtre. La durée d'une séance est fixée à 3h pour toutes les pièces de théâtre.

1. Créer les classes `Spectacle`, `Film`, `PieceTheatre`, `Salle`, `SalleTheatre` et `Creneau` en utilisant les descriptions fournies ci-dessus. Compléter ces classes avec les constructeurs et les méthodes qui vous semblent nécessaires. Les numéros des films s'incrémenteront de 1 à chaque nouveau film. Le premier film créé aura le numéro 100. Les numéros des pièces de théâtre s'incrémenteront de 1 à chaque nouvelle pièce. La première pièce créée aura le numéro 1000. Les numéros des salles s'incrémenteront de 10 à chaque nouvelle salle. La première salle créée aura le numéro 10. Pour gérer les horaires, on pourra utiliser la classe `Horaire` définie en TD.

   Dans la classe `Salle`, on pourra ajouter un attribut `lesCreneauxOccupes` de type `SortedMap<Integer,Set<Creneau>>` de manière à stocker pour chaque jour de la semaine l'ensemble des créneaux occupés ce jour là. On choisit un ensemble pour ne pas accepter deux fois le même créneau. Lors de la création d'une salle, `lesCreneauxOccupes` sera vide. Ajouter les méthodes suivantes :

- `public boolean estDisponible(Creneau c)` qui teste si la salle est disponible sur le créneau passé en paramètre. Il ne faut pas que les créneaux se chevauchent. Par contre, si une séance termine à un horaire donné, on considère que la suivante peut commencer à ce même horaire.

    - `public boolean ajouterCreneau (Creneau c)` qui teste l'existence du jour de ce créneau comme clé dans `lesCreneauxOccupes`. Si une ligne existe, on ajoute le créneau à l'ensemble associé à la clé. Sinon, on crée un ensemble de créneaux auquel on ajoute le créneau et on ajoute une ligne dans `lesCreneauxOccupes` avec ce nouveau jour et cet ensemble. Avant d'ajouter un créneau, on testera que la salle est bien disponible sur celui-ci.

- `public boolean pasDeCreneauCeJour(int jour)` qui retourne `true` s'il n'existe pas de créneau attribué à la salle pour le jour passé en paramètre et `false` sinon.

2. Créer les classes `Seance`, `SeanceFilm` et `SeanceTheatre` en utilisant les descriptions fournies ci-dessus.

Pour chaque séance, on souhaite connaître le nombre de places standard disponibles, le taux de remplissage de la salle pour la séance. Pour une séance, on aura besoin de consulter le jour, l'horaire de début et celui de fin.

Chaque séance proposera un service de vente de places standard : `public void vendrePlacesTN(int nbre)`. Deux séances seront égales si elles sont programmées sur le même créneau. Le test d'égalité des séances est donc indépendant de la salle dans laquelle la séance est projetée.

Une séance de cinéma proposera en plus un service de vente de places standard à tarif réduit : `public void vendrePlacesTR(int nbre)`.

Une séance de théâtre proposera en plus un service de vente de places de type fauteuil : `public void vendrePlacesFauteuil(int nbre)`. On pourra également consulter le nombre de fauteuils disponibles pour cette séance.

3. La classe `Spectacle` devra également permettre de gérer l'ensemble des séances programmées pour ce spectacle sachant qu'on ne doit pas avoir deux fois la même séance associée à un spectacle. On souhaite pouvoir ordonner les séances d'un spectacle suivant le jour et l'horaire programmés. Choisir le type de données abstrait adapté à la gestion des séances dans la bibliothèque des collections Java.

   Un `Spectacle` proposera les services suivants :

   - rechercher une séance correspondant à un jour et un horaire de début donnés.
   - rechercher les séances correspondant à un jour donné et retourner une liste de séance(s) (dans le cas d'une pièce de théâtre cette liste contiendra au plus une séance).
   - obtenir le taux moyen de remplissage pour ce spectacle.
   - obtenir le chiffre d'affaires de ce spectacle.

   Dans la classe `Film`, on ajoutera également la méthode `public boolean ajouterSeanceFilm (SeanceFilm s)` qui ajoute une séance dans l'ensemble des séances de ce film à condition qu'elle n'existe pas déjà et retourne `true` sinon retourne `false`.

   Dans la classe `PieceTheatre`, on ajoutera également la méthode `public boolean ajouterSeanceTheatre (SeanceTheatre s)` qui ajoute une séance dans l'ensemble des séances de cette pièce à condition qu'il n'existe pas

déjà une séance le même jour et retourne `true` sinon elle retourne `false`.

4. Créer une classe `GestionProgrammationSemaine` qui implémente l'interface `IProgrammationSemaine` fournie ci-dessous et dont le fichier `IProgrammation.java` est fourni. **L'implémentation doit tenir compte des commentaires Javadoc associés à chaque méthode de l'interface**.

```
public interface IProgrammationSemaine {
	public Film rechercherFilm (String titre, String realisateur);
	public void ajouterFilm (String titre, String realisateur, int
	duree);
	public void ajouterInterprete( int numSpectacle, String
	interprete);
	public PieceTheatre rechercherPiece (String titre, String
	metteurEnScene);
	public void ajouterPiece (String titre, String metteurEnScene, int
	nbEntractes);
	public void ajouterSeanceFilm(int idFilm, int jour, Horaire debut,
	int idSalle);
	public boolean existeSeanceCeJour(int idPiece, int jour);
	public void ajouterSeanceTheatre(int idPiece, int jour, Horaire
	debut, int idSalle);
	public double chiffreAffaires(int numSpectacle);
	public double getTauxRemplissage(int numSpectacle);
	public void vendrePlaceFilmTN(int idFilm, int jour, Horaire debut,
	int nbPlacesTN);
	public void vendrePlaceFilmTR(int idFilm, int jour, Horaire debut,
	int nbPlacesTR);
	public void vendrePlacePieceTN(int idPiece, int jour, int
	nbPlacesTN);
	public void vendrePlaceFauteuil(int idPiece, int jour, int
	nbFauteuils);
	public String lesFilms ();
	public String lesPieces();
	public String lesSallesFilm();
	public String lesSallesTheatre();
	public String lesSeancesFilm (int idFilm);
	public String lesSeancesTheatre (int idPiece);
	public int getNbPlacesDispo(int numSpectacle, int jour, int heures,
	int minutes);
	public boolean existeFilm (int idFilm);
	public boolean existePiece (int idPiece);
```

```
    public boolean existeSeanceFilm (int idFilm, int jour, int heures,
    int minutes );
    public boolean existeSalleFilm (int idSalle);
    public boolean existeSalleTheatre (int idSalle);
    public int dureeFilm(int idFilm);
    public boolean salleStandardDisponible (int jour, Horaire debut,
    int duree, int idSalle);
    public void reinitialiserProgrammation();
    public int getNbFauteuilsDispo(int idPiece, int jour);
    public int getNbPlacesDispo(int idPiece, int jour);
    public boolean estUnNumeroFilm(int numSpectacle);
    public boolean estUnNumeroPiece(int numSpectacle);
    public Spectacle getSpectacle( int numSpectacle);
}
```

Ajouter à la classe `GestionProgrammationSemaine` les attributs permettant la gestion des salles, des salles de théâtre, des films et des pièces de théâtre. On pourra nommer ces attributs `lesSalles`, `lesSallesTheatre`, `lesFilms` et `lesPieces`. Choisir le type de données abstrait adapté pour ces attributs dans la bibliothèque des collections Java. Vous noterez que la plupart des méthodes accèdent aux salles, aux films et aux pièces de théâtre à partir de leur numéro. Vous pourrez donc stocker des paires d'informations de type (clé, valeur) pour optimiser l'accès aux salles, aux films et aux pièces de théâtre connaissant leur numéro.
Le constructeur de la classe `GestionProgrammationSemaine` se chargera de créer 4 salles standard et 4 salles de théâtre et de les ajouter à `lesSalles` et `lesSallesTheatre`.

5. Créer une classe `Main` contenant la méthode `main`. Créer un objet de type `GestionProgrammationSemaine` puis ensuite demander à l'utilisateur de choisir parmi les actions suivantes, et cela en boucle jusqu'à ce qu'il décide de quitter:
    - réinitialiser programmation (0)
    - ajouter un film (1)
    - ajouter une pièce de théâtre (2)
    - ajouter un nom d'interprète à un spectacle (3)
    - ajouter une séance pour un film (4)
    - ajouter une séance pour une pièce de théâtre (5)
    - vendre des places pour un film (6)
    - vendre des places pour une pièce de théâtre (7)

- consulter le chiffre d'affaires et le taux de remplissage d'un spectacle (8)
- quitter (9)

Vos classes doivent respecter le principe d'encapsulation. Vous ajouterez et redéfinirez les méthodes que vous jugez nécessaires. Vous devez utiliser les concepts vus en cours pour éviter la duplication du code et mutualiser les caractéristiques communes et les comportements communs de certaines classes. Vous devez gérer les erreurs susceptibles de survenir dans les différentes méthodes et commenter ces méthodes de manière à préciser les erreurs pouvant être levées (commentaires Javadoc). Vous serez particulièrement vigilant aux erreurs de saisie de l'utilisateur.

Le projet est à réaliser en binôme. Créer une archive (.zip) avec vos fichiers sources et la déposer dans le cours en ligne en cliquant sur **Dépôt du projet Java** correspondant à votre groupe de TP. Vous déposerez un seul projet par binôme. **Date limite du dépôt : dimanche 5 Janvier 2020.**

Vous pouvez constituer un binôme avec un étudiant d'un autre groupe que le votre. Dans ce cas, envoyez un mail aux deux enseignants de TP concernés pour les tenir informés.

La constitution d'un trinôme pourra être tolérée mais sera limitée à un trinôme par groupe dans le cas d'un nombre impair d'étudiants dans ce groupe. Vous devez en faire la demande auprès de votre enseignant de TP par mail.

Des soutenances seront organisées la semaine du 6 Janvier. Durée : 15 min par groupe. Le créneau précis de votre passage vous sera communiqué via le forum du cours en ligne. On vous demandera prochainement d'inscrire les groupes sur un wiki pour faciliter l'organisation.

Pendant votre soutenance, vous devrez présenter les choix de conception que vous aurez fait ainsi que le détail de la répartition du travail dans le binôme.

Vous ferez une démonstration du fonctionnement de votre application en suivant le scénario suivant :

1. Ajouter un film (titre : f1, réalisateur : r1, durée 120).

2. Ajouter un film (titre : f2, réalisateur : r2, durée 90).

3. Ajouter un film (titre : f1, réalisateur : r1, durée 120)
   → affichage d'une erreur « Film déjà existant » avec retour au menu.

4. Ajouter une pièce de théâtre (titre : p1, metteur en scène : m1, nombre d'entractes : 3).

5. Ajouter un interprète
   → affichage des films existants et des pièces existantes
   → choix idFilm : 30
   → affichage d'une erreur « Numéro de film inexistant » avec retour au menu.

6. Ajouter un interprète
   → affichage des films existants et des pièces existantes
   → choix idFilm : 101
   → saisie interprète : i1

7. Ajouter une séance pour un film
   → affichage des films existants (ou message « Aucun film » si aucun film avec retour au menu principal)
   → choix idFilm 100
   → saisie jour : 1
   → saisie heures : 10
   → saisie minutes : 30
   → choix idSalle : 10

8. Ajouter une séance pour un film
   → affichage des films existants
   → choix idFilm 101
   → saisie jour : 1
   → saisie heures : 11
   → saisie minutes : 15
   → choix idSalle : 10
   → affichage de l'erreur « Salle indisponible sur ce créneau ».

9. Ajouter une séance pour un film
   → affichage des films existants
   → choix idFilm 101
   → saisie jour : 2
   → saisie heures : 14
   → saisie minutes : 0
   → choix idSalle : 40

10. Ajouter une séance pour un film
    → affichage des films existants
    → choix idFilm 101
    → saisie jour : 2
    → saisie heures : 15
    → saisie minutes :30
    → choix idSalle : 40

11. Ajouter une séance pour une pièce
    → affichage des pièces existantes
    → choix idPiece 1000
    → saisie jour : 1
    → saisie heures : 20
    → saisie minutes : 30

→ choix idSalle : 50

12. Ajouter une séance pour une pièce
→ affichage des pièces existantes
→ choix idPiece 1000
→ saisie jour : 1
→ affichage de l'erreur « Il existe déjà une séance programmée ce jour »

13. Vendre des places pour un film
→ affichage des films existants
→ choix idFilm 101
→ affichage des séances pour ce film (ou message « Aucune séance » si aucune séance avec retour au menu principal)
Les séances du film : [Séance du 2 14h0 15h30

Nombre de places vendues: 0

Nombre de places vendues au tarif réduit: 0

En salle numéro 40,

Séance du 2 15h30 17h0

Nombre de places vendues: 0

Nombre de places vendues au tarif réduit: 0

En salle numéro 40]

→ choix jour : 2
→ choix heures : 14
→ choix minutes :0
→ affichage du nombre de places disponibles dans la salle numéro 40
→ saisie nombre de places tarif normal : 2
→ saisie nombre de places tarifs réduit : 3

14. Vendre des places pour un film
→ affichage des films existants
→ choix idFilm 101
→ affichage des séances pour ce film
Les séances du film : [Séance du 2 14h0 15h30

Nombre de places vendues: 2

Nombre de places vendues au tarif réduit: 3

En salle numéro 40,

Séance du 2 15h30 17h0

Nombre de places vendues: 0

Nombre de places vendues au tarif réduit: 0

En salle numéro 40]

→ choix jour : 2
→ choix heures : 14

→ choix minutes :0
→ affichage du nombre de places disponible dans la salle numéro 40
→ saisie nombre de places tarif normal : n1
→ saisie nombre de places tarifs réduit : n2 (on choisit n1 et n2 tels que n1 + n2> au nombre de places disponibles dans la salle 40.
→ affichage de l'erreur « Pas assez de places ».

15. Vendre des places pour un film
→ affichage des films existants
→ choix idFilm 101
→ affichage des séances pour ce film
Les séances du film : [Séance du 2 14h0 15h30

Nombre de places vendues: 2

Nombre de places vendues au tarif réduit: 3

En salle numéro 40,

Séance du 2 15h30 17h0

Nombre de places vendues: 0

Nombre de places vendues au tarif réduit: 0

En salle numéro 40]

→ choix jour : 2
→ choix heures : 10
→ choix minutes :0
→ affichage de l'erreur « Séance inexistante» retour au choix idFilm pour saisir le numéro du film de nouveau et affichage des séances.
choix jour : 2
→ choix heures : 15
→ choix minutes :30
→ affichage du nombre de places disponible dans la salle numéro 40
→ saisie nombre de places tarif normal : 5
→ saisie nombre de places tarifs réduit : 2

16. Vendre des places pour une pièce de théâtre
→ affichage des pièces existantes
→ choix idPieces 1000
→ affichage des séances pour cette pièce
Les séances de la pièce : [Séance du 1 20h30 23h30

Nombre de places vendues: 0

Nombre de places vendues au tarif fauteuil: 0

En salle numéro 50]

→ choix jour : 1
→ affichage du nombre de places standard disponibles : 50
→ affichage du nombre de fauteuils disponibles : 25
→ saisie nombre de places standard : 25

→ saisie nombre de fauteuils : 10

17. Vendre des places pour une pièce de théâtre
    → affichage des pièces existantes
    → choix idPieces 1000
    → affichage des séances pour cette pièce
    Les séances de la pièce : [Séance du 1 20h30 23h30

    Nombre de places vendues: 25

    Nombre de places vendues au tarif fauteuil: 10

    En salle numéro 50]

    → choix jour : 1
    → affichage du nombre de places standard disponibles : 25

    → affichage du nombre de fauteuils disponibles : 15
    → saisie nombre de places standard : 2
    → saisie nombre de fauteuils : 16

    → affichage de l'erreur « pas assez de fauteuils.

18. Consulter le chiffre d'affaires et le taux de remplissage d'un spectacle
    → affichage des films existants et des pièces de théâtre existantes
    → choix du numSpectacle : 101
    → affichage du chiffre d'affaires et du taux de remplissage.

19. Réinitialiser programmation

20. Ajouter une séance pour un film

    → affichage de l'erreur « Aucun film »

A tout moment, on pourra tester des saisies erronées telles que la saisie d'une chaîne de caractères non entière à la place d'un entier, la saisie d'un idFilm inexistant, la saisie d'un idPiece inexistant, la saisie d'un numSpectacle inexistant, la saisie d'un jour erroné ou d'un horaire erroné ou ne correspondant pas à une séance. Le programme ne doit pas s'interrompre mais doit afficher un message et proposer une nouvelle saisie de la valeur ou bien un retour au menu principal.

A chaque fois que le programme doit afficher une liste de films ou de pièces ou de séances. On sera vigilant au cas où cette liste est vide.