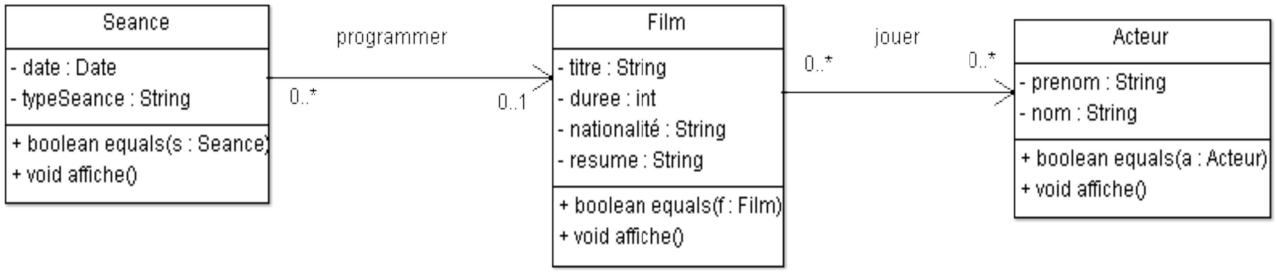
IUT Lannion **Sae 2.01**
Département informatique

Sujet 1 (première partie)
Implémentation Java d'un diagramme de classes

Le diagramme qui suit modélise des informations sur la programmation de films dans une salle de cinéma. Il peut y avoir jusqu'à quatre types de séance par jour : *matinée, début après-midi, fin après-midi* et *soirée*.



L'objectif de cet exercice est de « traduire » ce diagramme, c'est-à-dire d'écrire les classes Java nécessaires à l'implémentation du diagramme. Vous respecterez les consignes suivantes :

- toute classe possédera un constructeur qui reçoit les valeurs initiales en paramètre,
- vous veillerez à encapsuler toutes les variables,
- vous doterez chaque classe de tous les accesseurs nécessaires,
- vous testerez chaque classe de manière unitaire.

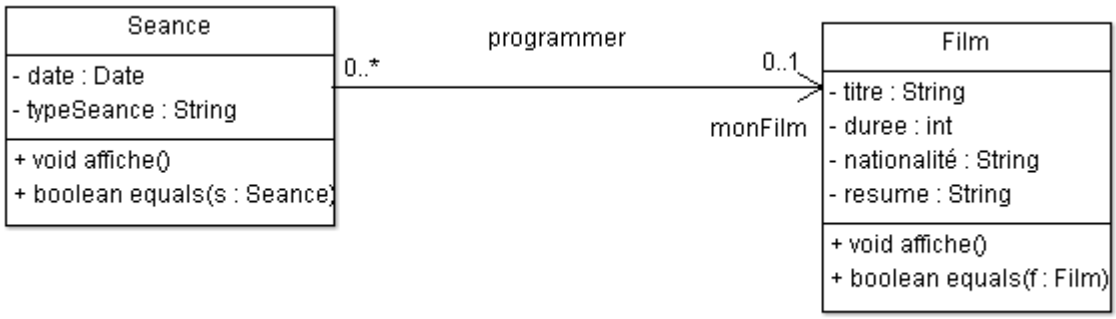
Question 1
Écrivez la classe `Acteur`. Ajoutez-y une méthode pour l'affichage d'un acteur sous la forme *prénom nom* (« Brad Pitt » par exemple). Redéfinissez la méthode `equals` : deux acteurs sont « égaux » s'ils ont le même nom et le même prénom.

Question 2
Écrivez la classe `Film`. **Pour l'instant, ne vous souciez pas de l'association `jouer`**. Outre un constructeur et des accesseurs, la classe `Film` devra contenir une méthode `affiche()` pour afficher les caractéristiques du film, sous la forme suivante :

<*titre*> Film <*nationalité*> de <*duree*> mn Résumé : <*resume*>

Redéfinissez dans cette classe la méthode `equals` : deux films sont « égaux » s'ils ont le même titre.

Question 3
On souhaite maintenant écrire la classe `Seance`, en tenant compte de l'association unidirectionnelle `programmer`.
On implémente cette situation à l'aide d'une variable, dans la classe `Seance`, qui correspond au rôle désignant l'extrémité de l'association (voir schéma ci-dessous). La gestion de l'association consiste en une gestion de ce rôle. L'association `programmer` étant unidirectionnelle, c'est la classe `Seance` qui va être responsable de cette gestion.



Écrivez un constructeur qui reçoit la date et le type de séance en paramètres. Lors de la création d'une instance de séance, on ne connaît pas encore le film qui y est programmé.

Ajoutez la méthode d'instance `public void ajouterProgrammer(Film f)` qui permet d'associer un film à la séance (cette méthode fera appel à une méthode `affecterFilm(Film f)`). Vous devrez programmer les vérifications qui s'imposent : tester que le film passé en paramètre n'est pas null (afficher un message d'erreur sinon).

Écrivez une méthode `public void affiche()` qui affiche la date et le type de la séance ainsi que le film prévu à cette séance (ou bien un message indiquant qu'aucun film n'est programmé à cette séance). Redéfinissez la méthode `equals` **:** deux séances sont « égales » si elles ont la même date et le même type de séance.

Question 4
Dans la classe `Seance`, il faut aussi prévoir le cas où l'on souhaite déprogrammer un film : si le film n'est plus programmé à cette séance, il faut enlever le lien entre la séance et le film. Réfléchissez à une méthode `public void enleverProgrammer(Film f)` et écrivez-la (elle fera appel à une méthode `public void enleverFilm(Film f)`.

Là encore vous programmerez les vérifications nécessaires : vérifier que le film à déprogrammer n'est pas null et vérifier qu'il correspond bien au film programmé à cette séance (afficher des messages d'erreur sinon).

Question 5
Complétez maintenant la classe `Film` de manière à prendre en compte l'association unidirectionnelle `jouer` : un film peut être associé à plusieurs acteurs, il faut donc ajouter à la classe `Film` une variable d'instance qui permet le stockage de plusieurs acteurs (un `ArrayList` par exemple).

Vous ajouterez :

- une méthode `public boolean contientActeur(Acteur a)` pour tester si l'acteur **`a`** passé en paramètre est déjà associé au film (*i.e.* déjà présent dans l'ArrayList `mesActeurs`).
- une méthode `public void ajouterActeur(Acteur a)` pour ajouter un nouvel acteur à l'ArrayList `mesActeurs`.
- une méthode `public void ajouterJouer(Acteur a)` pour ajouter une nouvelle instance de l'association `jouer` en faisant les vérifications nécessaires.

L'ajout d'une instance de l'association `Jouer` doit être vérifié : l'acteur à ajouter doit exister (non null) et ne doit pas être déjà présent dans le film.

Vous modifierez la méthode `affiche()` de la classe `Film` pour qu'elle affiche aussi à l'écran les acteurs du film. Vous écrirez donc une méthode `public void listerActeurs()` pour lister à l'écran les acteurs du film.

Question 6
Pour compléter la gestion de l'association unidirectionnelle `jouer`, écrivez une méthode `public void enleverJouer(Acteur a)`, qui fera appel à une méthode `void enleverActeur(Acteur a)`.
Là encore vous programmerez les vérifications nécessaires.

Question 7
Écrivez une classe `Principale` possédant une méthode `main` chargée de :

1 - Créer ces deux films, et ajoutez-y leurs acteurs :

BIS Film français de 102 mn Résumé : Éric et Patrice sont amis depuis le lycée. Au fil des années, chacun a pris un chemin très différent : d'un côté Éric, hédoniste sans attaches aux multiples conquêtes, et de l'autre Patrice, père de famille «monogame» à la vie bien rangée. Acteurs **:** Franck Dubosc Kad Merad Alexandra Lamy.

PAPA OU MAMAN Film français de 104 mn Résumé : Florence et Vincent Leroy ont tout réussi. Leurs métiers, leur mariage, leurs enfants. Et aujourd'hui, c'est leur divorce qu'ils veulent réussir. Acteurs : Marina Foïs Laurent Lafitte

2 - Créer un ArrayList de séances et y ajouter les quatre séances du 18 mars 2017 :
18/03/2017 – matinée
18/03/2017 – début après-midi
18/03/2017 – fin après-midi
18/03/2017 – soirée

3 - Programmer le film « BIS » le 18 mars 2017, à la séance de la matinée et à la séance de la soirée.

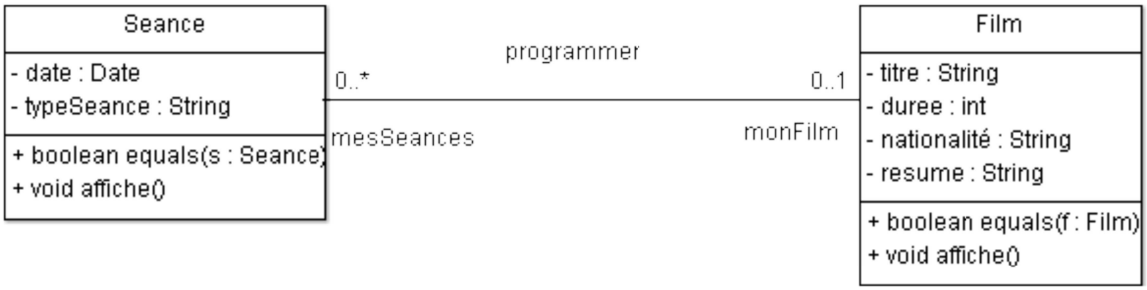
4 - Programmer le film « PAPA OU MAMAN » le 18 mars 2017 en début d'après-midi.

5 - Afficher tout le programme du 18 mars 2017, c'est-à-dire afficher toutes les séances de cette date.

Question 8
On souhaite écrire, dans la classe `Film`, une méthode d'instance qui liste à l'écran toutes les séances où le film est diffusé. Est-ce possible ? Pourquoi ?

De la même manière, peut-on écrire dans la classe `Acteur` une méthode qui liste à l'écran tous les films dans lesquels l'acteur a joué ? Pourquoi ?

Question 9
On propose ce nouveau diagramme, où les associations sont bidirectionnelles.

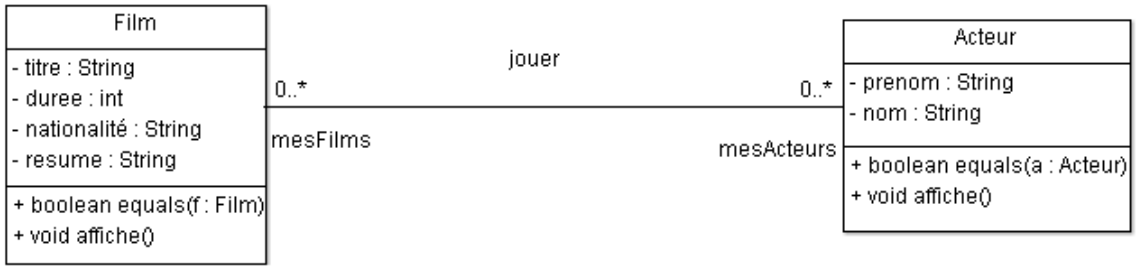


L'implémentation de ce schéma passe par la création de variables qui correspondent aux rôles désignant les extrémités de l'association. La gestion de l'association consiste en une gestion "synchrone" de ces deux rôles. Il faut par contre choisir laquelle des deux classes sera "responsable" de cette gestion.

Ici nous choisissons, arbitrairement, la classe `Seance` comme responsable de la gestion de l'association `programmer`.

Reprenez votre classe `Film` et ajoutez lui une variable d'instance `mesSeances`, capable de stocker plusieurs séances (car 0..*). Écrivez une méthode d'instance `public void ajouterSeance(Seance s)` qui ajoute au film la séance passée en paramètre.
Enfin, dans la classe `Seance`, adaptez les méthodes `ajouterProgrammer()` et `enleverProgrammer()`.

Question 10
De la même manière, modifiez vos classes de manière à respecter le nouveau diagramme.



Question 11
On suppose maintenant que la relation *jouer* est caractérisée par une information, le nom du rôle que joue l'acteur dans le film. Cela se traduit par une classe-association comme le montre le diagramme ci-dessous.

Modifiez votre programme pour tenir compte de ce nouveau diagramme.