Exercice JAD's show

Design Pattern en Java

Scénario

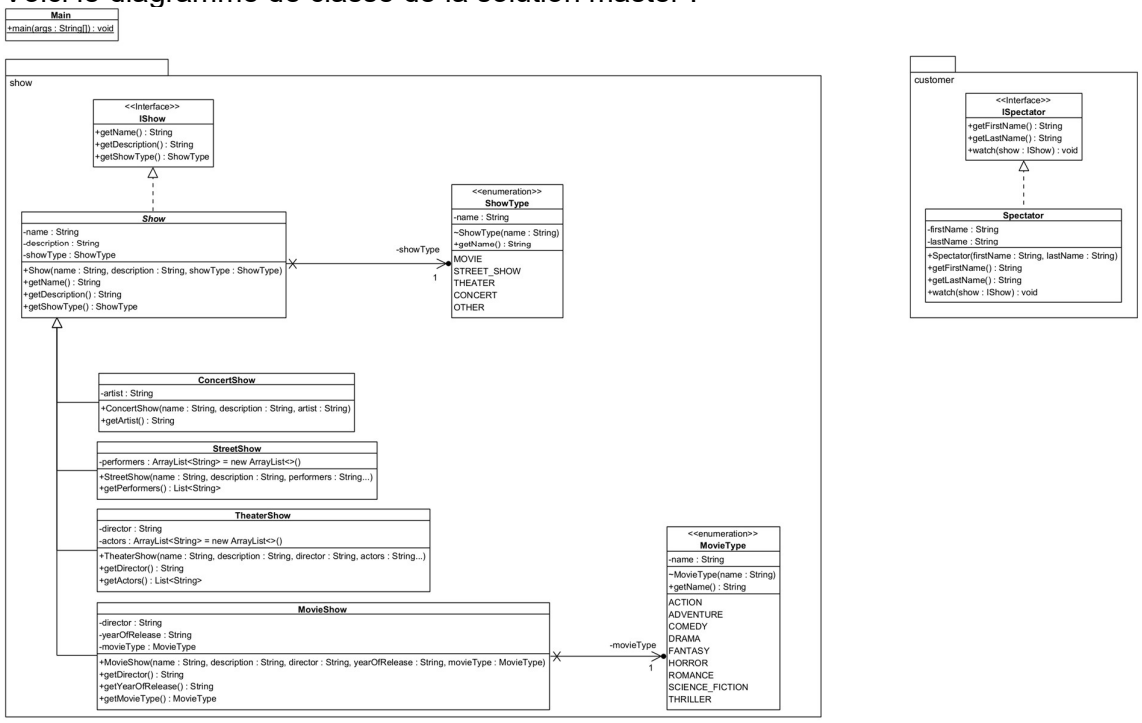
Vous devez reprendre un début de code et le compléter.

Pour démarrer, faîtes un fork du dépôt GitHub suivant et cloner le ensuite sur votre poste.

https://github.com/Jean-Aymeric/jad-s-show

Une fois cela fait, ne regarder pas d'autres branches que celle de master. En effet les autres branches contiennent les solutions. Ne vous gâchez pas le plaisir de les réaliser vous-même.

Voici le diagramme de classe de la solution master :



Si vous lancez le main, vous devez obtenir ceci.

```
Process finished with exit code 0
```

Le programme n'affiche rien, mais il fonctionne.

Exercice JAD's show

Design Pattern en Java

Step 1

Le premier step consiste à utiliser un DP pour permettre aux spectateur `jad` d'assister à chacun des shows.
Vous devez implémenter la méthode `watch()` de la classe Spectator. Vous avez le droit de construire d'autres attributs, méthodes, classes et ou interfaces.

Cependant, la seule contrainte que je vous impose est qu'aucun `switch case` ne devra polluer votre code. Et non il n'est pas non plus autorisé de remplacer ce `switch case` part une succession de `if` tout aussi moches.

Pour vous en sortir, il faudra piocher des les DP (je pense que vous l'aviez deviné), mais lequel.

Vous devez obtenir quelque chose ressemblant à cela :

```
J'ai assisté au film Titre du film de Nom du réalisateur sorti en 2023

J'ai assisté à la pièce de théâtre Titre du spectacle de théâtre de Nom du metteur en scène.
Il y avait : Nom de l'acteur 1, Nom de l'acteur 2, Nom de l'acteur 3,

J'ai assisté au spectacle de rue Titre du spectacle de rue.
Il y avait : Nom de l'artiste 1, Nom de l'artiste 2, Nom de l'artiste 3,

J'ai assisté au concert Titre du concert de Nom de l'artiste ou du groupe
```

Les indices sont écrits en blancs sur fond blanc, il vous suffit de les sélectionner et de les copier pour les lire.

Indice 1 :

Indice 2 :

Step 2

Le premier step consiste à utiliser un DP pour faciliter la création de chacun des shows.

Comme vous vous y attendez, vous allez devoir piocher dans les DP de création.

Je vous laisse deviner lequel (ou lesquels) est (ou sont) le (ou les) plus judicieux.

Votre première idée est certainement de passer par le DP Factory Method.
Le souci est que les constructeurs ont des paramètres différents.

```
public ConcertShow(final String name,
                   final String description,
                   final String artist)

public MovieShow(final String name,
                 final String description,
                 final String director,
                 final String yearOfRelease,
                 final MovieType movieType)

public StreetShow(final String name,
                  final String description,
                  final String... performers)

public TheaterShow(final String name,
                   final String description,
                   final String director,
                   final String... actors)
```

Nous allons découper le problème en deux pour rendre la chose plus compréhensible.

Fabriquez une classe abstraite `ShowFactory` permettant de construire chacun des types de show. Elle disposera de 4 méthodes `make` spécifiques.

Puis transformer vos classes spécialisées de `Show` pour interdire l'instanciation directe hors du package en rendant le constructeur `package`. Pour cela il suffit de supprimer le mot clef `public` devant le constructeur.

Step 3

Bon, vous avez implémenté une factory, mais ce n'est pas vraiment plus simple pour la création. Elle ne passe plus par l'utilisation du constructeur spécifique, mais d'une méthode spécifique. Cependant, vous avez avancé tout de même, car toutes ces méthodes sont sur une unique classe.

L'idée est maintenant de n'utiliser qu'une et une seule méthode `makeShow()`. Mais comme les paramètres de construction sont spécifiques à chaque classe, il va falloir parser les paramètres envoyés à cette méthode.

La signature de la méthode va être la suivante :

```
public static IShow makeShow(final String showDescription)
```

Le paramètre `showDescription` va être analysé et découpé pour permettre d'obtenir les vrais paramètres.

Par exemple, pour obtenir un `MovieShow` il faudra lancer la commande suivante :

```
shows.add(ShowFactory.makeShow(
                "MOVIE:name=Titre du film;" +
                        "description=Description du film;" +
                        "director=Nom du réalisateur;" +
                        "yearOfRelease=2023;" +
                        "movieType=SCIENCE_FICTION"));
```

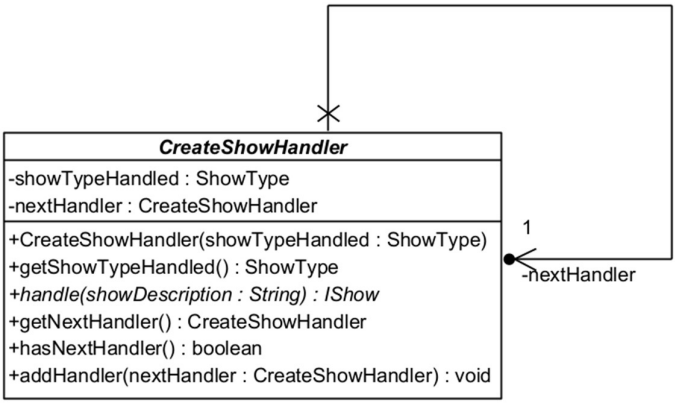
Je vous conseille de vous aider du code que j'ai fourni pour le DP Chain of Responsibility. Inutile dans votre cas, de passer par un interpréter. Cependant, je vous invite fortement à implémenter un Singleton sur votre chaine.

Si vous êtes bloqués, il y a des indices sur la page suivante.

Exercice JAD's show

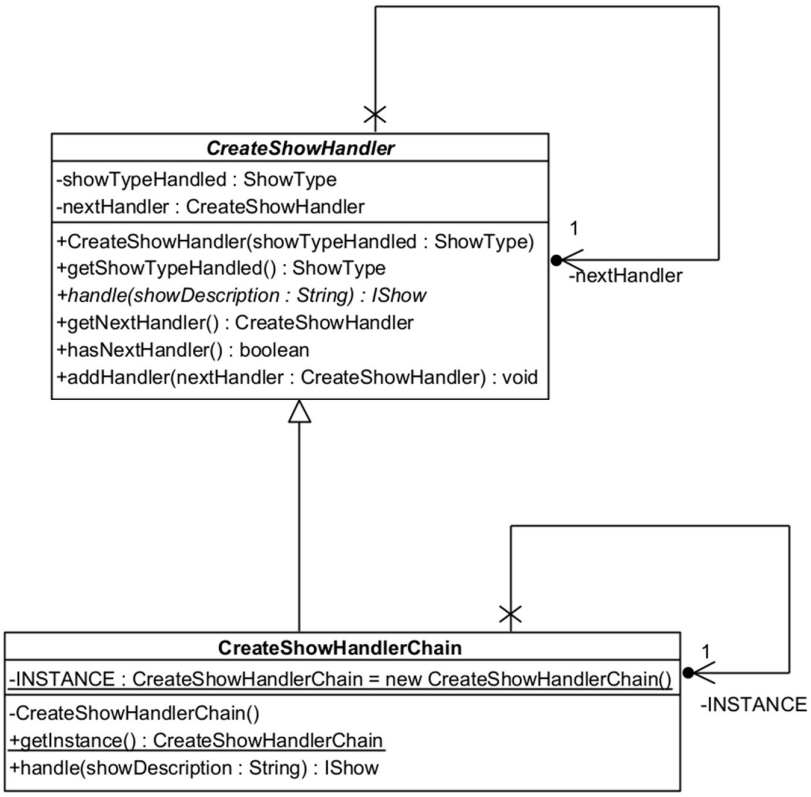
Design Pattern en Java

Indice 1 : Commencez par créer une classe abstraite `CreateShowHandler`.



Cette classe est l'abstraction de vos futurs maillons.

Indice 2 : Créer ensuite le premier maillon de votre chaine.



Et oui c'est sur celle-ci que vous placerez un singleton.

Attaquez ensuite la création des autres maillons. Chacun s'occupant, d'un type de show et découpant le paramètre `showDescription`.