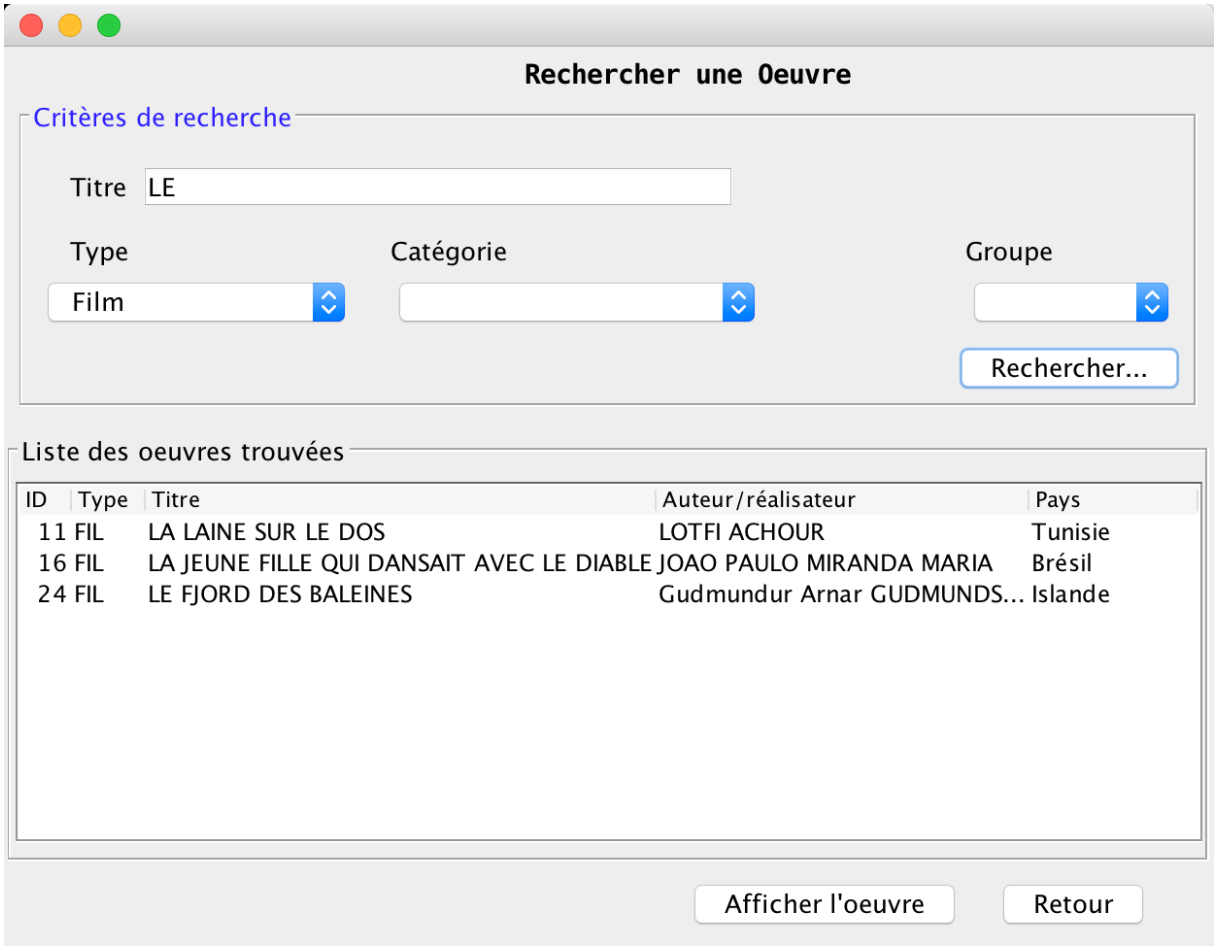
Java Avancé – TP 4
JTable, menus et barre d'outils

Durée : 4h (à rendre pour Lundi 31 mai avant 8h)

On repart de l'interface du TP3 pour compléter notre gestion du catalogue pour le backOffice de VOD. L'objectif de ce cours TP est de vous familiariser avec le composant JTable avec modèle explicite.

PARTIE A – CREATION D'UNE JTABLE

Nous allons **afficher le résultat de la recherche d'œuvre dans une table.** Vous allez dans ce TP supprimer la textArea pour exploiter une JTable :



La table contiendra le résultat des recherches successives : les données de la table seront donc modifiées plusieurs fois (ajout, suppression). Nous allons utiliser une instance de

`AbstractTableModel`. Dans le composant `JTable` en mode design, pour le modèle de la table, mentionner par un code personnalisé que vous allez utiliser une instance de votre modèle (libre à vous de choisir les noms) :

```
maTable.setModel( new TableRechercheModele() );
```

C'est ici qu'on relie la vue (la `JTable`) avec notre modèle de données. Partout où on devra utiliser les données de la table, on récupèrera le modèle avec `maTable.getModel()`, qu'on caste avec la bonne classe.

Q1 – Que renvoie getModel() par défaut dans JTable ?

PARTIE B – MODELE DE LA JTABLE

B.1 – Le modèle de la table va contenir une liste des œuvres avec 5 propriétés qu'on imagine communes à toutes les œuvres : *ID, type d'œuvre, titre, auteur/réalisateur et pays*. Une idée est de **créer une nouvelle classe** avec ces éléments (si vous avez créé une classe Œuvre, peut-être pouvez-vous l'exploiter ici directement). Élaborer cette classe avec un *constructeur* et les *accesseurs* des 5 propriétés. Nous l'appellerons **`ResultatDeRecherche`** pour la suite de l'énoncé.

Q2 – Dans quel package la placez-vous ?

B.2 - Créer ensuite un **modèle de la table** qui étend `AbstractTableModel`, et qui comporte les intitulés des colonnes (tableau de strings) et une liste des données (la classe précédente), par exemple un arrayList.

Implémenter les 3 méthodes obligatoires de `AbstractTableModel`. Utilisez les méthodes et champs des classes Java, qui permettent de connaître les dimensions demandées.

Q3 – Quelles sont ces 3 méthodes ? Pourquoi doivent-elles être obligatoirement implémentées à votre avis ?

B.3 – Testez votre interface : vous pouvez observer que les **noms des colonnes** ne sont pas affichés. Il faut en effet redéfinir, dans notre modèle, la méthode `getColumnName()` pour que la vue affiche les noms des colonnes :

```
@Override
public String getColumnName(int col) {
}
```

Écrivez cette méthode (une ligne).

B.4 – On s'attaque maintenant à la recherche : au lieu d'afficher les résultats dans la textArea après que l'utilisateur ait cliqué sur le bouton *Rechercher*, on va **ajouter une ligne** au modèle que l'on vient de créer.

Écrire la méthode `ajouterLigne()` du modèle qui crée une instance de votre classe **`ResultatDeRecherche`** si des données ont été trouvées.

N'oubliez pas les méthodes `firexxx()` du modèle de table, pour que la `JTable` visualise effectivement les résultats des modifications du modèle (cf cours, `fireTableStructureChanged()` convient ici).

Q4 – Rappelez en quelques mots à quoi servent ces méthodes.

B.5 – Améliorez votre interface pour une **meilleure ergonomie** : effacer les résultats d'une recherche précédente ; caler les entiers à droite, les chaines à gauche ; optimiser la largeur des colonnes…

PARTIE C – AFFICHAGE D'UNE ŒUVRE

Il s'agit ici d'afficher l'œuvre que l'utilisateur a choisi dans la table quand il clique sur le bouton **Afficher**. On utilise pour cela la méthode `getSelectedRow()` de la `JTable`.

Q5 – Que se passe-t-il si l'utilisateur clique sur Afficher sans avoir rien sélectionné ?

On pourra écrire une méthode `getRow()` du modèle, qui retourne l'instance choisie par l'utilisateur.

(FACULTATIF) Gérer la sauvegarde du film après que l'utilisateur l'ait modifié : traitement du bouton « **Valider** ».

PARTIE D (FACULTATIVE) – MENUS ET BARRE D'OUTILS

Vous pouvez compléter votre IHM de menus et barre d'outils, selon vos besoins.

Ce design permet d'alléger les pages, puisque certaines actions auparavant effectuées par les boutons exploitent ici soit les menus, soit la barre d'outils.

TRAVAIL A RENDRE

Comme pour les autres TPs, vous déposerez votre travail + CR de TP avec les questions sur Tomuss.

Donnez aussi un **CR de séance** : dans ce CR non technique, vous mentionnez les difficultés rencontrées liées à votre travail à distance par ex., ou des problèmes de concentration, et votre avancement. Le TP est prévu sur 6h, cours3 et démos compris.