Programmation Orientée Objet
IG2I, L3 – 2015-2016

CTP – 26 Novembre 2015

Diego Cattaruzza, Philippe Kubiak, Maxime Ogier

Conditions de l'examen

- **durée** : 2h30;
- ordinateurs personnels autorisés;
- documents et pages internet autorisées;
- accès à Moodle autorisé;
- les moyens de communication (mails, messageries instantanées, Facebook, Tweeter, téléphones portables, etc.) sont INTERDITS.

Rendu

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt du CTP 2015/2016** une archive compressée de votre répertoire de projet. Cette archive doit être nommée **CTP_Nom_Prenom.zip** avec votre nom et votre prénom. Vérifiez bien que les fichiers sources que vous avez réalisés lors du CTP y sont présents.

Notation

Cette épreuve de CTP est évaluée sur 20 points. 2 points sont réservés à la bonne présentation du code (indentation correcte, respects des conventions Java dans les noms de classe, attributs, méthodes et variables, commentaires au format Javadoc).

Consignes

Le sujet est composé de 4 parties liées entre elles. Vous devez commencer par répondre aux questions des parties 1 et 2. Puis vous pouvez traiter les parties 3 et 4 dans l'ordre que vous voulez (3 avant 4 ou 4 avant 3). La partie 5 est un bonus pour ceux qui auraient fini en avance.

Vous êtes bien sûr autorisés à rajouter les attributs ou méthodes que vous jugez nécessaires même si ce n'est pas dit explicitement dans le sujet.

Sujet

On se propose ici d'ajouter de nouvelles fonctionnalités à l'interface graphique créée dans le TP sur la gestion du cinéma (TP5). L'idée est de rajouter une nouvelle fenêtre dans laquelle on a un objet **JSlider** qui permet de sélectionner une année, et un objet **JTree** qui affiche les films sortis par année. Puis, un clic sur un film affiche une boîte de dialogue qui contient les titres des autres films dans lesquels a joué l'acteur principal du film sélectionné.

1 Modèle objet et requêtes sur la base de données (4 points)

Question 1. Ajoutez une nouvelle classe **InfoFilm** (dans le paquetage **modele** si vous l'avez créé). Ajoutez dans cette classe les attributs suivants : le titre du film, son année de sortie et un objet de type **Acteur** qui représente l'acteur principal du film. Ajoutez un constructeur par données, les accesseurs et mutateurs nécessaires ainsi qu'une méthode **toString()**. Ajoutez une méthode principale (**public static void main(String[] args)**) et testez le bon fonctionnement de cette classe.

Question 2. Ajoutez dans la classe **RequeteCinema** une méthode **public List<Integer> ensAnneesSorties() throws SQLException** qui renvoie une liste contenant toutes les années de sortie des films. Dans votre requête vous pouvez utiliser la fonction SQL **YEAR(d)** qui permet de récupérer l'année de la date **d** sous forme d'un entier. Vous pouvez, à titre d'exemple, vous **inspirer** de la requête suivante :

```
SELECT YEAR(f.sortie) AS annee FROM film f
```

Remarque : Il est aussi possible de récupérer la date complète sous forme d'un objet **String** et de récupérer la sous-chaîne qui contient l'année.

Question 3. Ajoutez dans la classe **RequeteCinema** une méthode **public List<InfoFilm> ensFilmsAnnee (int annee) throws SQLException** qui renvoie une liste contenant les objets **InfoFilm** pour tous les films sortis l'année **annee**. À nouveau, dans votre requête vous pouvez utiliser la fonction SQL **YEAR(d)** qui permet de récupérer l'année de la date **d** sous forme d'un entier. Vous pouvez, à titre d'exemple, vous **inspirer** de la requête suivante :

```
SELECT f.titre, YEAR(f.sortie) AS annee, a.nacteur, a.nom, a.prenom
FROM film f
INNER JOIN acteur a ON f.nacteurprincipal = a.nacteur
WHERE YEAR(f.sortie) = ?
```

Testez le bon fonctionnement de vos requêtes.

2 Mise en place de la partie graphique (4 points)

Question 4. Ajoutez une nouvelle fenêtre **Filmographie**, qui ressemblera à la fenêtre proposée à la Figure 1 après avoir traité les questions suivantes. Placez-y les objets graphiques suivants :

- 3 **JLabel** pour l'année minimale, l'année maximale et l'année courante;
- 1 **JSlider** pour faire le choix de l'année;
- 1 **JTree** pour afficher les titres de films par année.

Ajoutez dans la fenêtre **GestionVideoClub** un nouveau bouton qui permette d'ouvrir cette nouvelle fenêtre (même si pour le moment on affiche pas les bonnes informations dans cette nouvelle fenêtre).

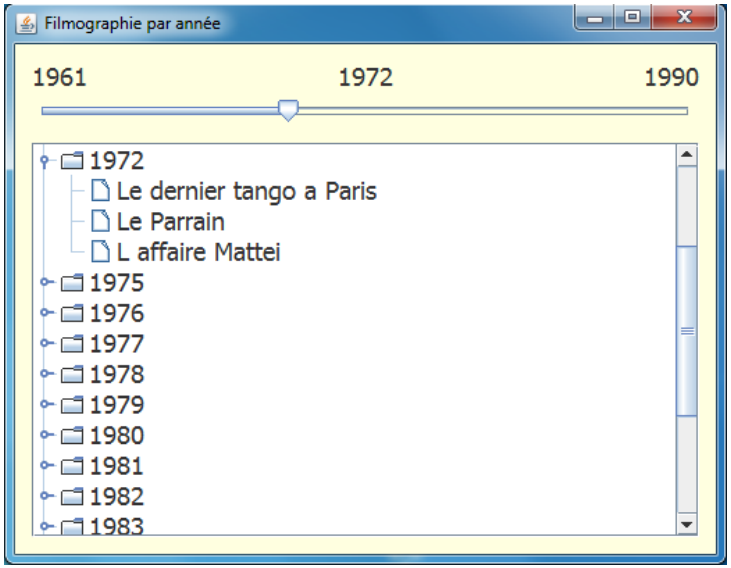

Figure 1: Exemple de fenêtre pour la filmographie par année.

Question 5. Remplissez les trois **JLabel** avec les valeurs :

- de l'année minimale des années de sortie des films;
- de l'année maximale des années de sortie des films;
- de l'année courante initialisée à la moyenne de l'année minimale et de l'année maximale.

Configurez l'objet **JSlider** en faisant appel à ses méthodes **setMinimum()**, **setMaximum()** et **setValue()**. On rappelle que pour transformer un entier **i** en chaîne de caractère, on peut faire appel à la méthode **String.valueOf(i)**.

Question 6. Le **JLabel** central doit indiquer l'année courante pointée par le curseur du **JSlider**. Pour cela, ajoutez un écouteur sur l'événement **StateChanged** du **JSlider** de telle sorte que quand l'utilisateur déplace le curseur, le label de l'année courante soit mis à jour. Vous pouvez utiliser la méthode **getValue()** du **JSlider**. Cette méthode renvoie la valeur courante associée au curseur.

Question 7. Remplissez l'objet **JTree** de telle sorte que :

- le nœud racine contienne "Films par année de sortie";
- les nœuds de premier niveau contiennent les années de sortie;
- les nœuds de second niveau (les feuilles) contiennent les objets **InfoFilm** de telle sorte que sous le nœud d'une année on ait les films sortis cette année là.

Pour les nœuds de second niveau on veillera à ce que seul le titre du film soit affiché. En fait, pour l'affichage d'un nœud, le **JTree** fait appel à la méthode **toString()** de l'objet associé au nœud. Ainsi, il faudra que la méthode **toString()** de la classe **InfoFilm** renvoie l'information que l'on souhaite afficher.

Testez ensuite que tout fonctionne correctement à l'ouverture de la fenêtre **Filmographie**.

3 Affichage dynamique du JTree (5 points)

L'idée ici est de faire en sorte que le mouvement du curseur (sur le **JSlider**) implique une mise à jour de l'affichage du **JTree**. Seule l'année demandée est déployée au niveau du **JTree**, et la vue sur le **JTree** est déplacée sur cette année.

Question 8. Ajoutez dans la classe **Filmographie** une méthode **fermerTout** qui referme tous les nœuds de premier niveau de l'arbre (on doit donc voir la racine et les années, mais pas les titres de films). Vous pouvez écrire cette méthode de manière récursive ou itérative.

Quelques indications :

- on peut obtenir le nœud racine de l'arbre en récupérant le modèle lié au **JTree** (avec **getModel()**), puis en faisant appel à la méthode **getRoot()** sur le modèle; on récupère alors un objet de type **Object** que l'on peut caster en **DefaultMutableTreeNode**;

- on peut récupérer le nombre de nœuds fils d'un nœud en faisant appel à la méthode **getChildCount()** sur le nœud père;
- on peut récupérer le $i^{ème}$ fils d'un nœud père en faisant appel à la méthode **getChildAt(i)** sur le nœud père (les fils sont numérotés à partir de 0);
- on peut récupérer le chemin (de type **TreePath**) qui identifie un nœud **node** de la manière suivante : **new TreePath(node.getPath())**; pour cela il faut que **node** soit de type **DefaultMutableTreeNode**;
- pour fermer un nœud dans l'arbre, on peut faire appel à la méthode **collapsePath(TreePath path)** en passant en paramètre le chemin qui identifie ce nœud; ceci ferme le nœud, i.e. on ne voit plus les fils, mais le nœud reste visible.

Question 9. Ajoutez dans la classe **Filmographie** une méthode **private DefaultMutableTreeNode chercherNoeud (String nodeStr)** qui recherche dans l'arbre un nœud dont l'affichage est la chaîne de caractère **nodeStr**. Si le nœud est trouvé on renvoie ce nœud, sinon on renvoie **null**.

Quelques indications :

- on peut récupérer un objet de type **Enumeration<DefaultMutableTreeNode>** qui permet de parcourir tout le sous-arbre à partir d'un nœud en appelant la méthode **breadthFirstEnumeration()** sur ce nœud (qui doit être de type **DefaultMutableTreeNode**);
- les objets de type **Enumeration** fournissent une méthode **hasMoreElements()** et une méthode **nextElement()** qui permettent de faire le parcours de l'énumération;
- la méthode **getUserObject()** de la classe **DefaultMutableTreeNode** renvoie l'objet associé au nœud.

Question 10. Ajoutez dans la classe **Filmographie** une méthode **private void ouvrirNoeudAnnee (String annee)** qui ferme tous les nœuds de premier niveau de l'arbre, recherche le nœud dont l'affichage est **annee**, et l'ouvre s'il est présent. Vous ferez aussi en sorte que le nœud ouvert soit bien visible dans la vue du **JTree**. Faites appel à cette méthode dans l'écouteur sur le **JSlider** et vérifiez que tout fonctionne correctement. Si jamais vous rencontrez des difficultés avec l'écouteur sur l'événement **stateChanged**, vous pouvez implémenter à la place des écouteurs sur les événements **mouseDragged** et **mousePressed** du **JSlider**.

Quelques indications :

- pour ouvrir un nœud dans l'arbre, on peut faire appel à la méthode **expandPath(TreePath path)** (sur le **JTree**) en passant en paramètre le chemin qui identifie le nœud à ouvrir;
- on peut récupérer un objet de type **Rectangle** dans lequel des nœuds vont être affichés; pour cela la classe **JTree** fournit une méthode **getPathBounds(TreePath path)**;
- on peut récupérer le **Rectangle** visible dans la vue du **JTree** en appelant la méthode **getVisibleRect()** ;
- pour s'assurer que le rectangle contenant les nœuds à afficher soit bien dans la vue, il est préférable de spécifier la hauteur de ce rectangle (attribut **height**) de telle sorte qu'il ait la même hauteur que celle du rectangle visible;
- pour faire défiler la vue sur un rectangle, la classe **JTree** fournit une méthode **scrollRectToVisible(Rectangle rect)**.

4 Informations sur les films joués par l'acteur principal (5 points)

L'idée ici est de faire en sorte que lorsque l'utilisateur clique sur un titre de film dans le **JTree**, une boîte de dialogue informative s'ouvre et affiche les titres des autres films dans lesquels a joué l'acteur principal du film sélectionné. Par exemple, si on sélectionne le film *Le maître de guerre* dans lequel l'acteur principal est *Clint Eastwood*, alors la boîte de dialogue affichera les autres films dans lesquels *Clint Eastwood* est l'acteur principal, à savoir : *Le bon, la brute et le truand* et *Pour une poignée de dollars*. Cet exemple est présenté dans la Figure 2.

Question 11. Ajoutez dans la classe **InfoFilm** un attribut de type **Set<String>** qui contiendra les titres des autres films dans lesquels joue l'acteur principal (l'acteur qui est défini dans **InfoFilm**).

Question 12. Afin de remplir ce nouvel attribut avec les titres des autres films, vous avez deux possibilités (choisissez-en une seule).

1. Ajouter dans la classe **RequeteCinema** une méthode **public Set<String> autresFilmsActeur (Acteur a, String titre) throws SQLException** qui renvoie un **Set** contenant les titres des films (sauf **titre**) dans lesquels l'acteur **a** est l'acteur principal.
2. Ajouter dans la classe **InfoFilm** une méthode **public void setAutresFilmsActeur (List<InfoFilm> tousLesFilms)** qui, à partir de la liste de tous les objets **InfoFilm** retrouve les autres films dans lesquels joue l'acteur, et ajoute leur titre à l'attribut de type **Set<String>** de la classe **InfoFilm**.

Question 13. Faites maintenant en sorte que lorsque l'utilisateur clique sur un film dans le **JTree**, une boîte de dialogue de type informative s'ouvre si l'acteur principal a joué dans d'autres films. Le titre de cette boîte de dialogue doit

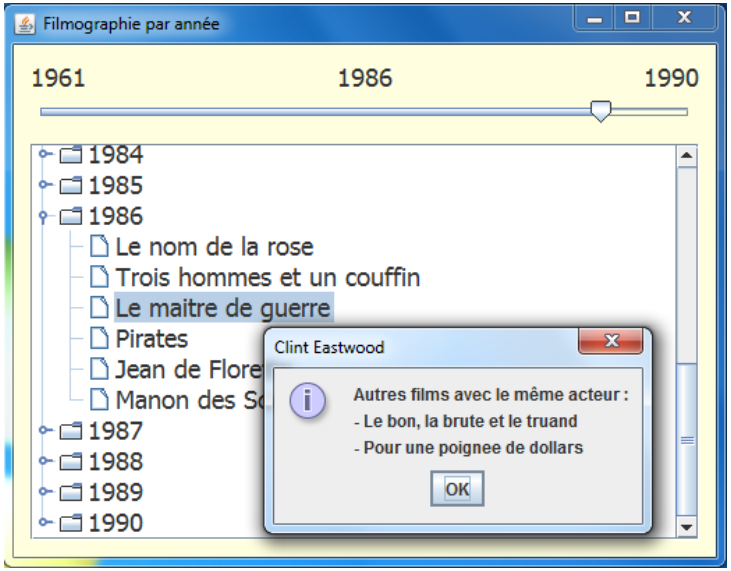

Figure 2: Lorsque l'utilisateur clique sur le film *Le maitre de guerre*, une boîte de dialogue affiche les autres films dans lesquels joue *Clint Eastwood*.

contenir le nom et le prénom de l'acteur principal, et la boîte de dialogue doit contenir les autres titres de films dans lesquels a joué l'acteur principal.

5 Bonus : info-bulle (2 points bonus)

Question 14. Reprenez la question précédente, en utilisant une info-bulle pour l'affichage des autres films de l'acteur principal.

Quelques indications :

- pour autoriser l'affichage des info-bulles sur un objet **tree** de type **JTree**, on peut écrire la ligne suivante : **ToolTipManager.sharedInstance().registerComponent(tree)**;
- pour afficher une info-bulle, on peut faire appel à la méthode **setCellRenderer(TreeCellRenderer tcr)** sur le **JTree**; et dans cette méthode on passe un objet qui hérite de **DefaultTreeCellRenderer** (le plus simple est de faire une classe anonyme);
- dans cette classe anonyme, il faut redéfinir la méthode **getTreeCellRendererComponent()** et faire appel dans cette méthode à **this.setToolTipText(String s)** afin de décider du texte à écrire dans l'info-bulle (si ce texte est égal à **null**, l'info-bulle n'est pas affichée);
- regardez bien dans la Javadoc quels sont les paramètres de la méthode **getTreeCellRendererComponent()** afin de vous en servir pour récupérer le nœud et savoir s'il s'agit d'une feuille ou non.