Projet : Gestion de plan de pièces de théâtres

Ce projet a pour but de traiter une pièce de théâtre qu’on pourra mettre en entrée et gérer les répétitions. Cette pièce de théâtre est composée d’actes, eux-mêmes composés de scènes qui contiennent des répliques.

**Ce que fait notre programme :** Le programme mène à la répartition des rôles à des acteurs en fonction des scènes et des actes de la pièce pour qu’ils puissent répéter leurs rôles. Chaque rôle a ses propres répliques. De plus, il faut prendre en compte la contrainte des acteurs (qu’on aura saisi aussi). En effet, si à une certaine date, l’acteur n’est pas disponible alors il ne pourra pas répéter, donc la scène dans laquelle il interprète devra être jouée un autre jour.

En résumé, il faut intégrer la pièce de théâtre, et créer un planning de répétitions (dont les dates seront précisées par l’utilisateur et le traitement des prises de plages se fera automatiquement) des différentes scènes avec une répartition des rôles en prenant en compte des contraintes des acteurs.

**Le programme principal offre un menu (géré par Gestion.java) à l’écran qui permet d’interagir avec :**

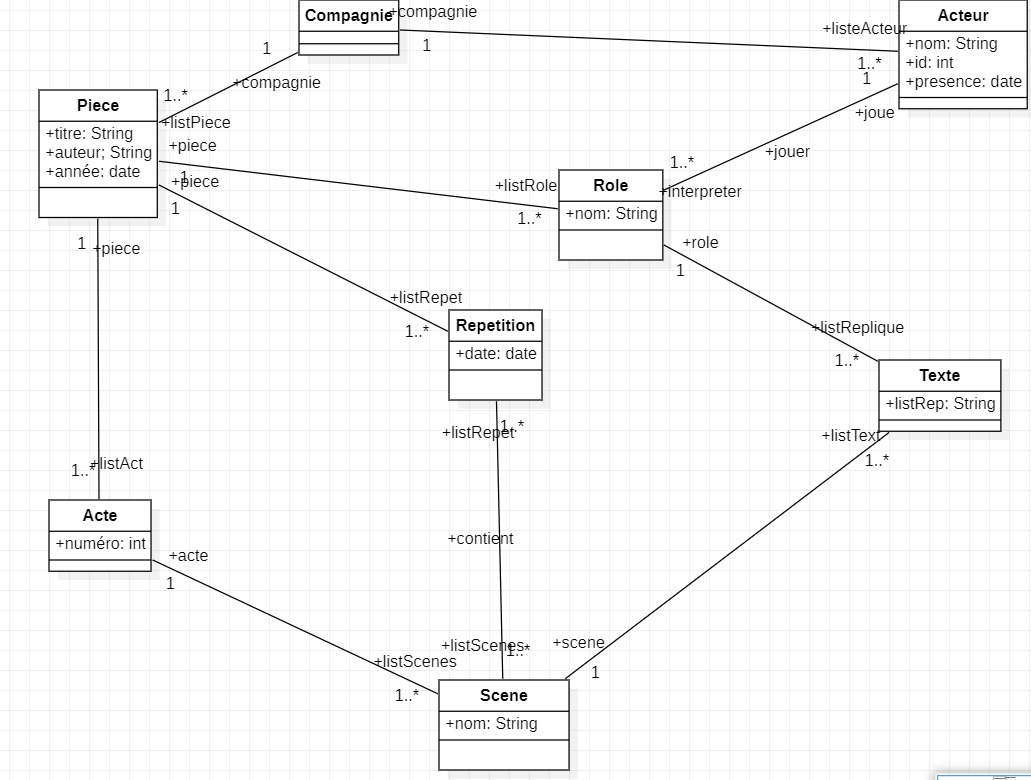
1. Fonctionnalités liées aux acteurs

2. Fonctionnalités liées aux pièces de théâtre

3. Fonctionnalités liées aux rôles

4. Fonctionnalités liées aux répétitions

5. Quitter

*Diagramme du projet montrant les liens entre toutes les classes et leurs attributs qu’on a créés.*

**Méthodes principales du programme :** Il faut noter que Compagnie.java est notre classe contenant la majorité des méthodes modelant ce programme. En effet, voici une liste de ces méthodes et leur fonctionnement :

* ajoutActeur(Acteur a): Ajoute un acteur à la liste des acteurs de la compagnie.
* ajoutPiece(Piece p): Ajoute une pièce à la liste des pièces de la compagnie.
* genid(Vector<Acteur> listeActeur): Génère un identifiant unique pour un acteur.
* genid\_r(Vector<Repetition> listeRep): Génère un identifiant unique pour une répétition.
* getPieceByTitle(String title): Récupère une pièce par son titre.
* getActeByNum(int num, Piece p): Récupère un acte par son numéro et sa pièce associée.
* getSceneByNum(int num, Acte a): Récupère une scène par son numéro et son acte associé.
* getActorIdByRole(Role r): Récupère l'identifiant d'un acteur par son rôle.
* getActeurById(int a): Récupère un acteur par son identifiant.
* getRoleByName(String name, Piece piece): Récupère un rôle par son nom et sa pièce associée.
* getActeurByName(String nom): Récupère une liste d'acteurs par leur nom.
* ajouter\_role(int a, String p, String r): Ajoute un rôle à un acteur pour une pièce donnée.
* removeSceneActe(Piece p, Acte a, Scene s): Supprime une scène d'un acte donné dans une pièce donnée.
* addJour(int a): Ajoute les jours de la semaine où un acteur est disponible.
* removeJour(int a): Enlève les jours de la semaine où un acteur est disponible.
* traitementTexte(File fichier, Piece piece): Traite un texte de théâtre pour extraire les informations d'actes, de scènes et de répliques.

**Autres points concernant le programme :** Même si Compagnie.java reste la classe de référence pour nos méthodes, nous avons laissé certaines méthodes propres à leurs classes telles que les trois méthodes ci-dessous dans Repetition.java:   
choixjour: Choix du jour de répétition en fonction des disponibilités des acteurs.   
choix\_rep\_scene: Imposition des jours de répétition en fonction des disponibilités des acteurs. fixer\_dates: Fixation des dates de répétition pour chaque occurrence des jours sélectionnés.   
De plus, Gestion.java est un fichier spécialement pour notre menu où on fait appel aux sous options des menus, avec les méthodes qui vont avec.

**Conclusion :** Bien que nous ayons eu des difficultés à coder ce projet, la répartition des tâches parmi notre trinôme nous a permis de surmonter les problèmes. Nous avons eu des difficultés notamment avec traitementTexte() qui ne fonctionne pas comme on le voudrait. Cependant, malgré ce seul défaut le programme réalise exactement ce qu’on voulait initialement. Nous avons pu apprendre à changer notre manière de penser grâce à ce projet, puisque le travail de découpage via les classes et leurs méthodes nous a permis d’appréhender le projet de manière plus efficace, tandis que si on avait fait la programmation non orientée objet, le travail aurait été plus laborieux et certainement moins efficace.