Adrien Guittard

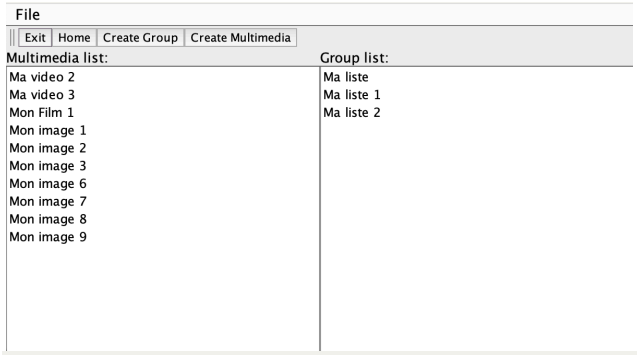
Rapport projet Inf224

L'objectif du projet est de programmer un gestionnaire de multimédia, accessible via une API, qui s'exécute sur un serveur ainsi que l'interface utilisateur permettant d'interagir avec le gestionnaire. La partie serveur est codée en langage C++ et l'interface utilisateur est en Java Swing. Ce système supporte trois types de fichiers multimédias : les vidéos, les images ainsi que les films. Il offre également la possibilité de créer des groupes afin de regrouper des multimédias ensemble et de créer des playlists.

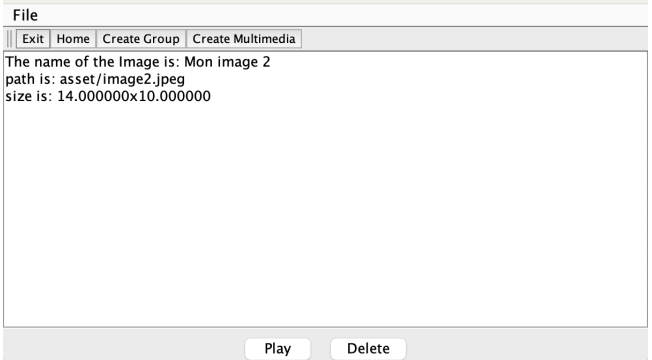
Le serveur s'articule autour d'un élément central : le contrôleur. Cet objet représente le gestionnaire. Il contient la liste des groupes ainsi que la liste des multimédias. Lorsque le serveur est en fonctionnement, le programme possède une instance de cette classe à laquelle on accède via l'API. Au total, 15 types de requêtes ont été implémentés afin de pouvoir effectuer diverses actions sur le contrôleur. Consultez le fichier texte "communication_protocole_template" pour avoir la liste des commandes ainsi que leur utilité.

L'interface utilisateur est divisée en deux parties : la visualisation et le modèle. Ici, le modèle tourne directement sur le serveur (le gestionnaire) et on y accède via des appels à l'API. J'ai donc fait le choix de regrouper toutes les méthodes liées à l'accès au serveur dans un même fichier du répertoire modèle, qui s'intitule "Client". C'est une classe importante qui se retrouve partout dans le reste du code car chaque action sur le modèle passe par un appel à l'API. La vue se décompose en plusieurs pages. Le détail est assez complexe, donc je vais illustrer les fonctionnalités via des captures d'écran pour faciliter la compréhension.

Présentation de l'interface utilisateur et des fonctionnalités implémentées:

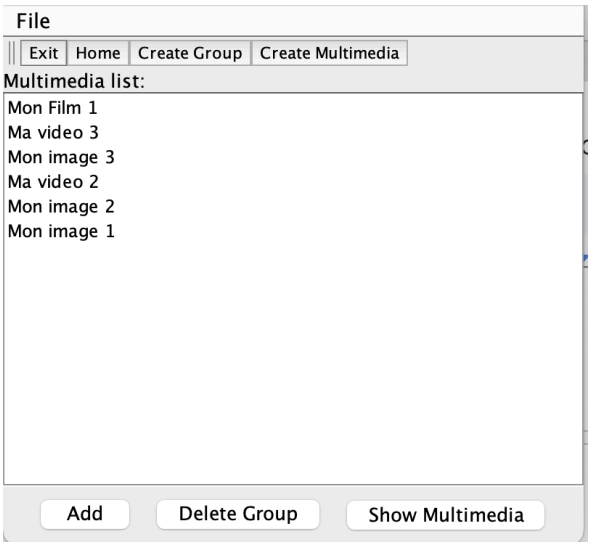


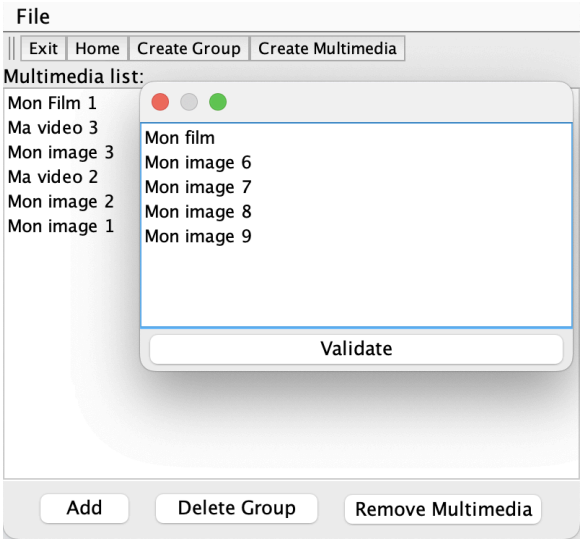
Lors du démarrage, l'utilisateur arrive sur la page d'accueil. On peut voir la liste des groupes ainsi que la liste des multimédias présents dans le gestionnaire. Les éléments des listes sont cliquables et redirigent vers une page dédiée.



En cliquant sur un multimédia, l'utilisateur est redirigé vers une page décrivant les informations concernant le multimédia. Deux boutons permettent de jouer le multimédia sur le serveur ou de le supprimer (de la liste et des autres groupes auxquels il appartient).

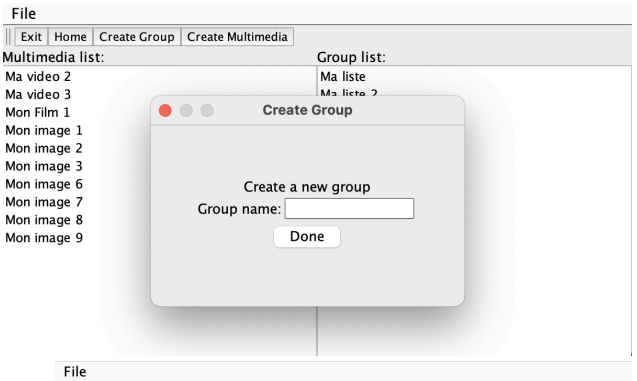
En cliquant sur un groupe, l'utilisateur arrive sur une page contenant encore une liste cliquable contenant les multimédias appartenant au groupe. Trois boutons permettent d'agir sur le groupe. Le bouton "Delete Group" supprime le groupe du contrôleur. Le troisième bouton est un bouton toggle. En mode "Show Multimédia" (comme sur la capture d'écran), si l'utilisateur clique sur un élément de la liste, il est redirigé vers la page du multimédia (capture d'écran précédente). En mode "Remove Multimédia", un clic sur un élément retire l'élément du groupe.

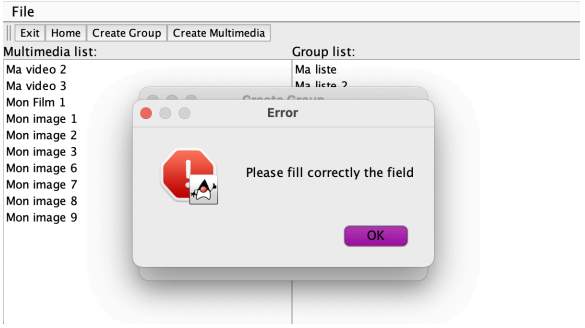


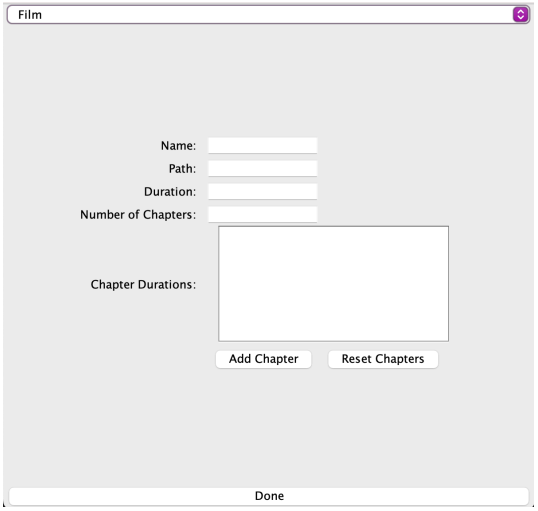


Le bouton "Add" ouvre une pop-up contenant la liste des multimédias qui ne sont pas présents dans le groupe. Un clic sur un élément l'ajoute au groupe. L'affichage du groupe est mis à jour à la fermeture de la pop-up.

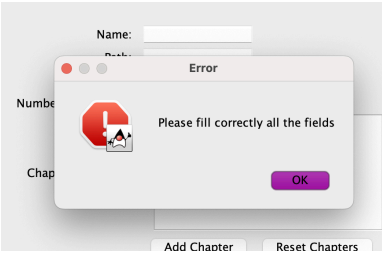
Sur la toolbar, il y a quatre boutons. Le premier quitte le programme, le deuxième ramène à la page d'accueil. Les troisième et quatrième permettent à l'utilisateur de créer un nouveau groupe ou un nouveau multimédia. Dès lors qu'un input est demandé à l'utilisateur, les valeurs entrées sont vérifiées pour afficher une erreur en cas de valeurs incohérentes.



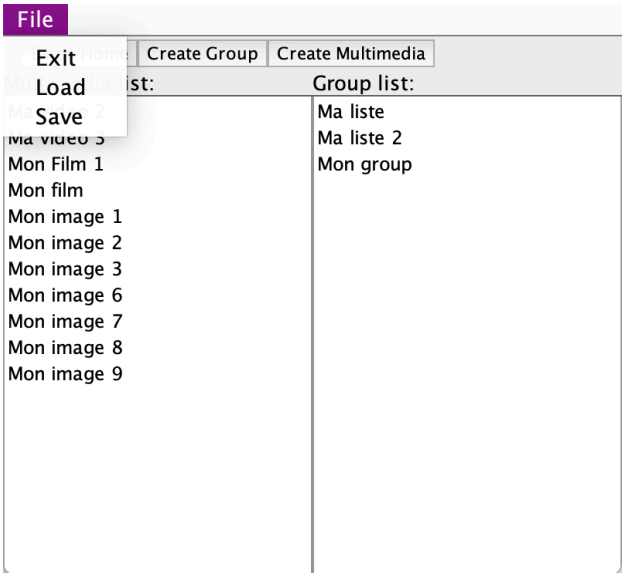


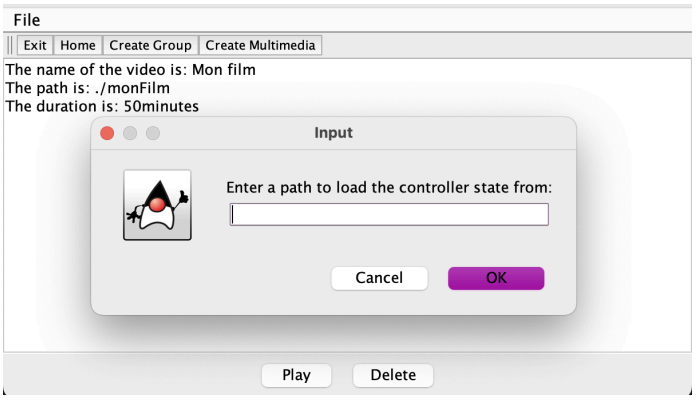


Pour créer un multimédia, une pop-up généralisée pour les vidéos, images et films s'affiche. Tout en haut, un menu déroulant permet de choisir le type de multimédia. Les champs pour chaque type de multimédia sont adaptés et les valeurs entrées sont également vérifiées.



La barre de menu contient trois éléments. Le premier permet de terminer l'exécution. Le deuxième permet de charger l'état du contrôleur (via la sérialisation). Le dernier sert à sauvegarder l'état. Pour les deux derniers éléments de ce menu, une pop-up s'affiche où l'utilisateur peut entrer le chemin d'un fichier pour sauvegarder/charger un contrôleur. De plus, l'état courant est automatiquement sauvegardé à la fin de l'exécution du programme et se charge également directement au lancement de l'application.





Toutes les parties du TP ont été abordées, y compris les questions facultatives. Par manque de temps, je n'ai pas pu correctement commenter le code de la partie sur Swing, mais la structure globale du code reste compréhensible et le programme est décomposé en de nombreuses classes nommées soigneusement. J'ai également fait un effort sur l'organisation des classes en packages pour pouvoir s'y retrouver facilement.

Réponses aux question:

Comment appelle-t-on ce type de méthode et comment faut-il les déclarer ?
Ce sont des méthodes abstraites. Pour les déclarer, il faut ajouter " = 0" derrière la signature de la fonction.

Quelle est la propriété caractéristique de l'orienté objet qui permet de faire cela ?
Le polymorphisme, et ici le downcasting.

Qu'est-il spécifiquement nécessaire de faire dans le cas du C++ ?
Pour pouvoir utiliser le polymorphisme, il faut déclarer les méthodes de la classe de base en virtual. Cela force le compilateur à vérifier les v-tables à la place de simplement exécuter la méthode de la classe de base.

Quel est le type des éléments du tableau : le tableau doit-il contenir des objets ou des pointeurs vers ces objets ? Pourquoi ? Comparer à Java.
Le tableau doit contenir des pointeurs vers les éléments car la taille de tous les éléments d'un tableau doit être la même. L'intérêt du tableau est de pouvoir accéder aux éléments en temps constant précisément grâce au fait qu'ils soient tous de même taille. Pour accéder à l'élément d'index i, je cherche à l'adresse index(base) + i * size(MonObjet). C'est la même chose en Java où les objets sont toujours des références.

Parmi les classes précédemment écrites quelles sont celles qu'il faut modifier afin qu'il n'y ait pas de fuite mémoire quand on détruit les objets ?
Film doit être modifié. Pour pouvoir utiliser un tableau, il faut d'abord réserver de l'espace en mémoire. C'est la seule classe qui utilise le mot-clé new pour l'allocation de mémoire.

*La **copie d'objet** peut également poser problème quand ils ont des variables d'instance qui sont des pointeurs. Quel est le problème et quelles sont les solutions ?*
Une copie en surface de l'objet va simplement recopier la valeur du pointeur et non le pointé. Pour pallier ce problème, on peut redéfinir l'opérateur d'affectation pour copier manuellement les champs concernés ainsi que le constructeur correspondant.

*On rappelle aussi que la liste d'objets doit en fait être une liste de **pointeurs** d'objets. Pourquoi ?*
Pour pouvoir utiliser le polymorphisme comme expliqué plus haut.

Comment peut-on l'interdire, afin que seule la classe servant à manipuler les objets puisse en créer de nouveaux ?
Il faut mettre les constructeurs en private / protected et marquer en friend les classes concernées afin que les autres ne puissent pas faire d'appels aux constructeurs. Pour pouvoir instancier des smart pointers, il faut créer une fonction privée statique qui fait appel au constructeur en interne. Cela est dû au fait qu'il n'est pas possible de déclarer les classes sous-jacentes aux smart pointers en friend (make_shared).