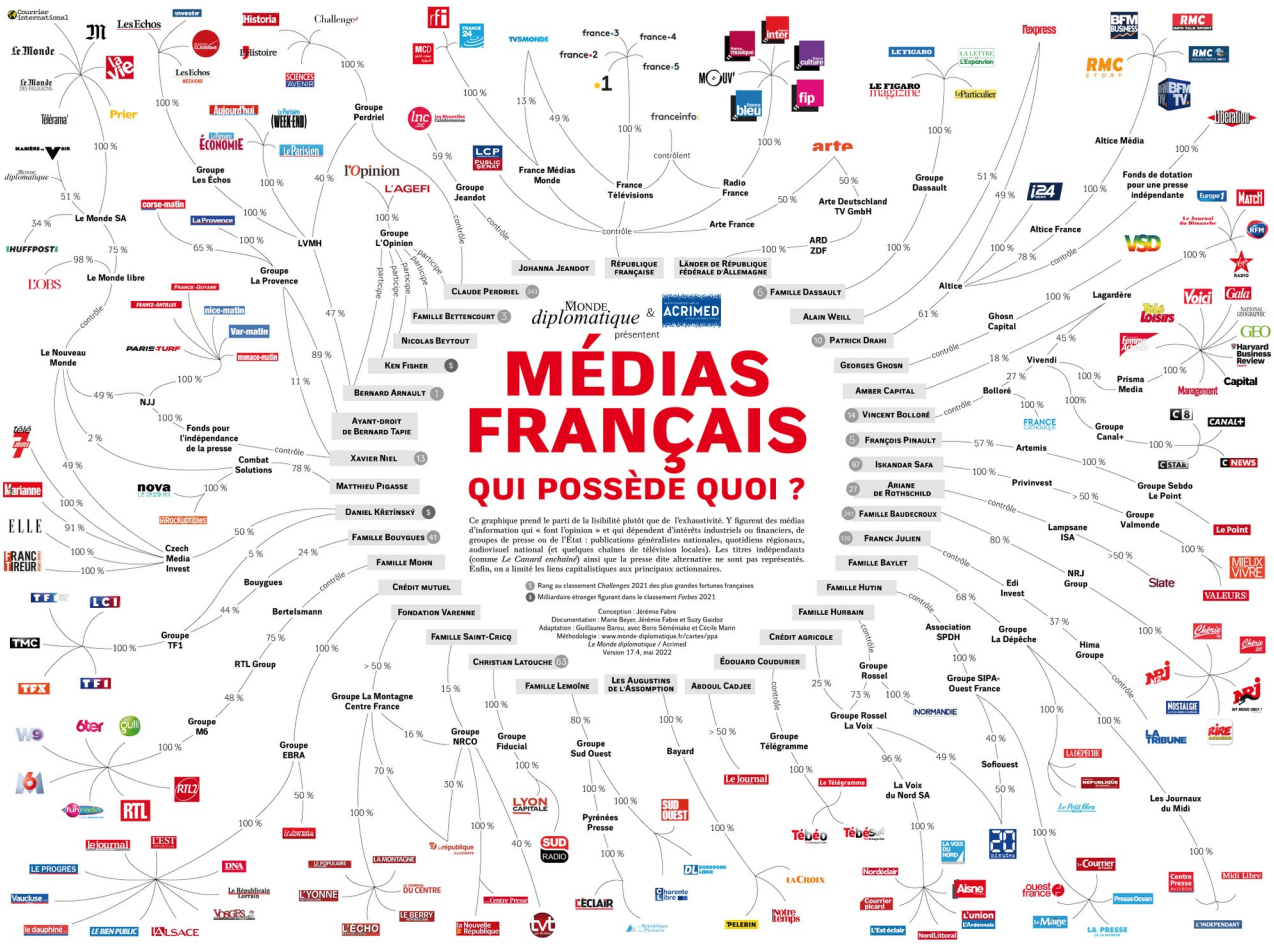
François BALMADIER
ET3 IIM

Année 2024-2025
Polytech Paris-Saclay

Projet Encadré JAVA : Vigie des médias



Introduction :

Dans un contexte où les médias jouent un rôle central dans la formation de l'opinion publique, il devient crucial de comprendre les liens capitalistiques qui unissent les différents groupes de presse à des acteurs économiques ou politiques. En effet, la propriété des médias peut influencer leur ligne éditoriale ou les contenus diffusés. Ce projet Vigie des médias vise à répondre à ce besoin d'analyse et de transparence en proposant une application Java permettant de modéliser les relations de propriété entre les médias français et les personnes ou entités qui les détiennent.

Ce projet, réalisé a la suite du cours Concepts Objet et Java, permet de mettre en œuvre les notions clés de la programmation orientée objet telles que l'encapsulation, le polymorphisme, ou encore la gestion des événements. Il s'inscrit dans une logique d'observation automatisée de l'évolution des parts de propriété dans les médias, ainsi que du suivi des mentions de certaines entités dans les publications. À travers une interface simple en mode console, l'utilisateur peut simuler des événements comme la publication d'un article ou le rachat de parts, et suivre les alertes générées par des modules spécialisés.

Cahier des charges fonctionnel et technique

- Objectifs généraux
  - Offrir une modélisation claire et évolutive des liens de propriété dans le paysage médiatique français.
  - Permettre la navigation dans les données via une interface console intuitive.
  - Simuler et suivre des événements médiatiques et financiers en lien avec les entités modélisées.
  - Détecter automatiquement des situations critiques et alerter via un système de vigie.

Fonctionnalités principales

- Modélisation des entités
  - Personnes physiques ou morales (groupes industriels, individus).
  - Médias : presse écrite, radio, télévision.
  - Liens de propriété (pourcentage de parts).
- Interface utilisateur (console)
  - Visualisation des données :
    - Par ordre alphabétique.
    - Par nombre décroissant de possessions.
  - Commandes interactives :
    - Rachat de parts.

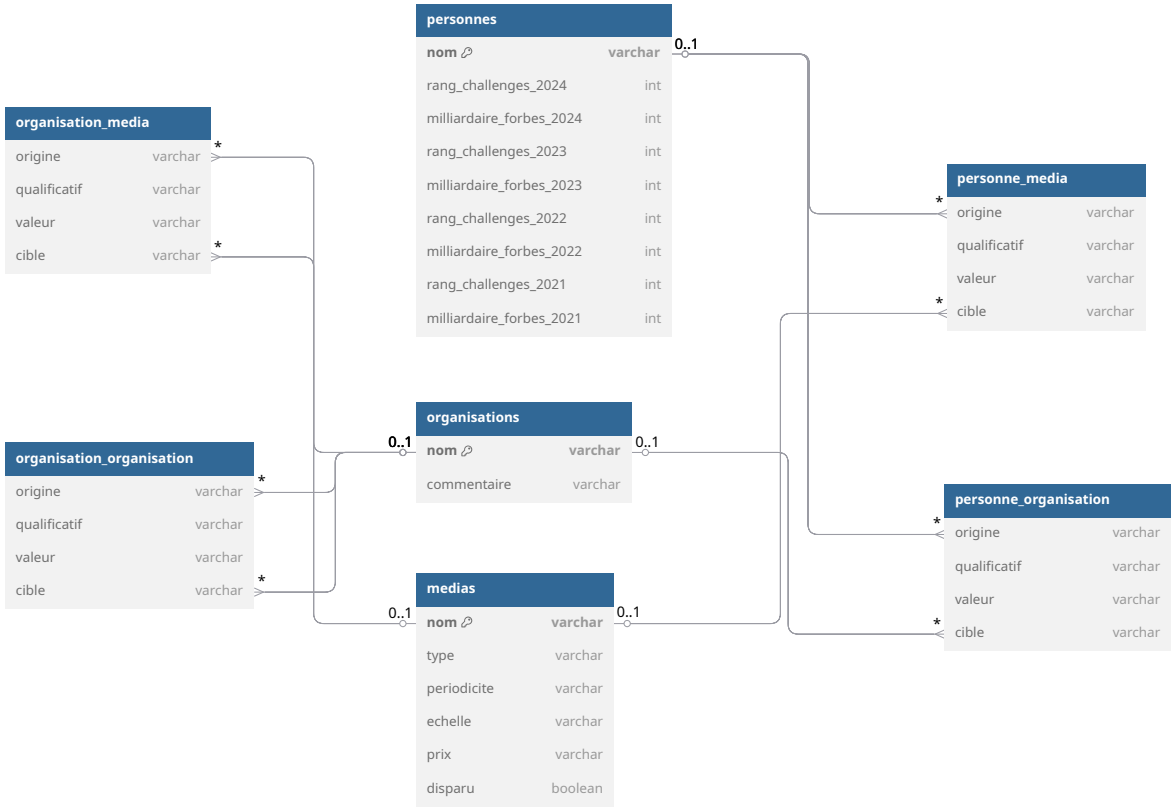
        - Publication ou diffusion (article, reportage, interview).
    - Saisie manuelle d'événements.
- Modules spécialisés
    - Suivi d'une personne :
        - Historique des mentions.
        - Pourcentage de mentions par média.
        - Alerte si la personne est mentionnée dans un média qu'elle possède.
    - Suivi d'un média :
        - Historique des rachats.
        - Nombre de mentions par entité.
        - Alerte si une entité devient nouveau propriétaire ou dépasse un seuil de mentions.
- Vigie des médias
    - Réception des alertes critiques.
    - Historisation des alertes.
    - Affichage dans la console.
    - Propagation d'événements
- Utilisation du patron de conception Observateur.
    - Modules abonnés à des types spécifiques d'événements (mentions, rachats).
    - Traitement et réaction en temps réel.

Contraintes techniques

- Langage : Java
- Architecture : application locale, non distribuée
- Ergonomie : interaction en mode console (interface graphique en option)
- Modélisation orientée objet : encapsulation, héritage, interface, exceptions
- Documentation : commentaires internes + JavaDoc
- Fichiers de données : possibilité d'importer ou simuler des jeux de données (optionnel)
- Exécution : version exécutable + sources + instructions

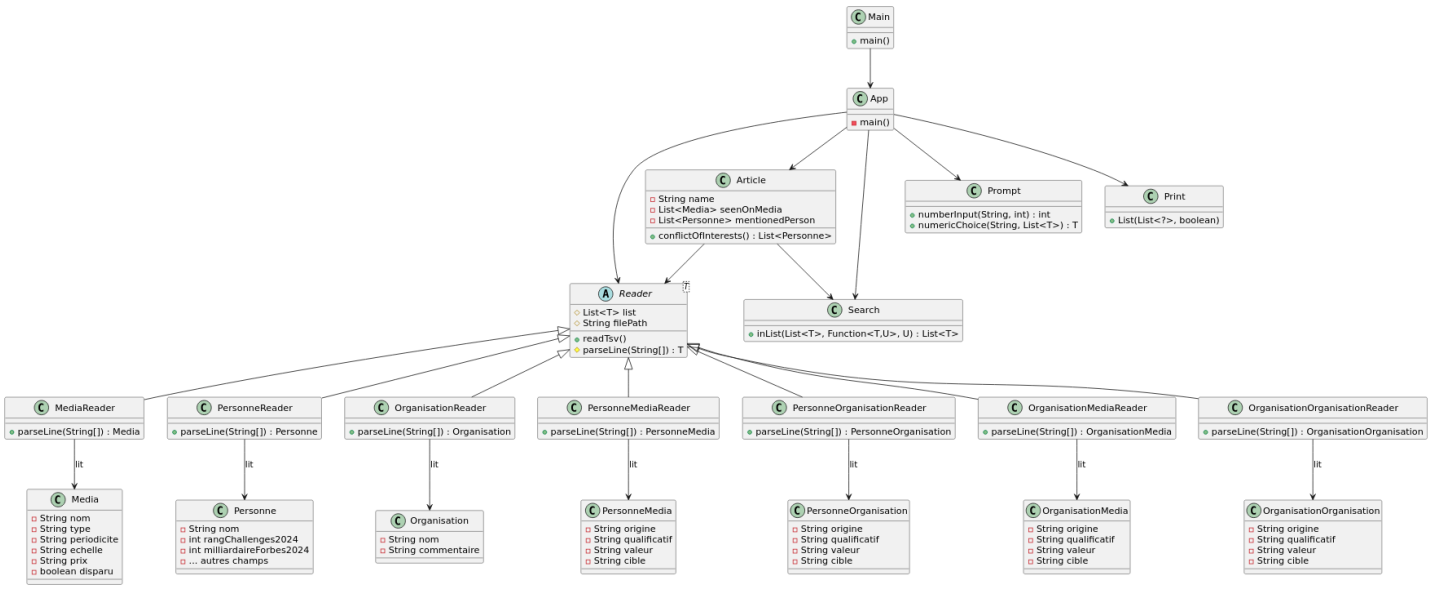
Structure du projet :

Les données qui nous on été proposées dans l'énoncé du projet était structuré par plusieurs fichier tsv (Tab-separated values, un fichier tableur) avec des objets clair (une personne, une organisation et un media) ainsi que des tableau reliant ces objets entre eu. C'est une structure qui ressemble beaucoup a celle des bases de données qu'on avait déjà appréhendé au premier semestre avec le module « Bases de données ».
J'ai essayer de faire une base de donnée SQLITE a partir des données tsv mais je n'ai pas réussi a cause de problème d'importation sur certains fichiers.
J'ai donc décidé de simuler une sorte de base de donnée simplifier dans mon programme en JAVA.
Voilà comment s'articule les différents types de donnés dans ce projet :



J'ai donc commencé par modélisés chaque objet correspondant a un fichier , avec ses différents champs ainsi que ses getters et son toString().

Puis j'ai coder les reader, ce sont des classes qui, a l'initialisation, vont lire le fichier .tsv qui leurs a été associé, le transforme en liste d'objets du type correspondant, et stockent cette liste. Ils ont tous pour héritage la classe Reader, qui est une classe abstraite qui regroupe le champs list et la fonction pour lire un .tsv et transformer chaque ligne en un Array de String. Puis une fonction parseLine(), qui transforme cette Array de String en un objet propre au fichier. La fonction parseLine() est donc abstraite car elle est a implémenter dans les readers spécialisés dans un fichier et dans un objet particulier.



Ensuite, j'ai codé la classe App pour me séparer de la classe main car je ne suis pas encore sur de bien comprendre comment elle fonctionne. Cette classe gère principalement les choix d'action possible en proposant plusieurs actions a chaque fois et il faut sélectionner un numéro pour se déplacer dans l'arbre des actions possibles.
C'est aussi elle qui fait les appels de fonction des autres classes suivant les actions demandées.

Pour éviter d'avoir trop de fonction et pour simplifier quelque partie du code, elle utilise des fonctions utilitaire rangées dans le package utils. Principalement Prompt qui possède quatre fonction pour demander a l'utilisateur un numéro, une chaîne de caractère, un objet dans une liste d'objets et plusieurs objets dans une liste. Mais il y a aussi Print pour afficher les liste d'objets de façon satisfaisante et Search qui permet de chercher un objet dans une liste en fonction d'un de ses paramètres.

Pour finir, la classe Article sert a proposé un système de publication d'article, on lui donne un nom, les médias dans lequel l'article a été publier et les personnalité mentionnée. L'article est ensuite stocké et une vérification des possesseur des média concerné sont identifiés pour vérifier si ils ne sont pas mentionnée dans l'article ce qui engendrerait un conflit d'intérêt qui serait immédiatement notifié a l'utilisateur.

Les difficultés :

La difficulté principale de ce projet a été le temps, j'ai commencé le projet bien trop tard et j'y ai donc consacrée seulement deux week-end, ce qui n'était clairement pas assez.

L'un de mes principale regrets est que la fonction de vérification de conflit d'intérêt ne fonctionne pas, je n'ai pas eu le temps de comprendre pourquoi et comment la réparer. Du coup elle renvoie tout le temps qu'il n'y a pas de conflit d'intérêt même quand je faisais exprès d'en mettre un évident.

J’ai aussi eu du mal a utilisée les choses vues en cours, notamment créer ses messages d’erreurs et les interfaces qui sont des fonctionnalités du langage JAVA dont je n’ai pas encore vu l’utilité.

J’ai eu aussi un peu de mal avec les méthodes générique de part leur syntaxe nouvelle pour moi, mais j’ai bien aimé le concept.

Conclusion :

Ce projet de Vigie des médias a été l'occasion pour moi de plonger concrètement dans la programmation Java. Bien que le temps limité n'ait pas permis d'aboutir à toutes les fonctionnalités prévues, cette expérience m'a apporté une première réelle pratique de la conception orientée objet.

J'ai pu mettre en place une structure de base pour modéliser les relations entre médias et leurs propriétaires, avec un système de navigation dans les données via une interface console.

Mes principaux regrets concerne le module de détection des conflits d'intérêts qui n'a pas pu être finalisé et toute les autres fonctionnalité que je n’ai même pas pu commencer a résoudre. Cette difficulté montre bien l'importance d'une bonne conception dès le départ et surtout d'une gestion rigoureuse du temps de développement.

Au final, Java s'est révélé un langage plaisant et efficace pour ce projet. Malgré quelques bizarreries comme les erreurs cryptiques ou le comportement déroutant d'equals(), j'ai apprécié sa syntaxe et sa productivité. Comparé au C, c'est vraiment mieux de pouvoir se concentrer sur la logique plutôt que sur la gestion bas niveau. Je trouve que Java permet d'écrire du code propre et fonctionnel relativement rapidement.