# **Rapport de Projet – Vigie des Médias**

**Auteur : Maël Tallard** **Lien GitHub :**<https://github.com/banzaiaiai/projet_java>

## **Introduction – Philosophie du Projet**

Ce projet a été réalisé dans le cadre de ma première année d'apprentissage du langage Java, et représente également mon premier projet individuel. Ce fut une expérience très enrichissante, bien que ponctuée de nombreuses difficultés, que j’aborderai plus en détail dans la section dédiée.

Mes compétences en Java étant encore limitées au moment de ce développement, j’ai principalement appliqué les concepts de programmation que j’avais appris auparavant, ce qui explique par exemple l’absence d’héritage dans le code. J’ai choisi de me concentrer sur le traitement des fichiers .tsv, car il me paraissait essentiel de pouvoir les modifier facilement et de manière lisible.

Étant donné que la base de données utilisée est de petite taille et que le projet était réalisé en Java, j’ai privilégié une approche simple et compréhensible, plutôt qu’une optimisation poussée du code.

## **Description des Réalisations**

### **1. La classe Menu**

La classe Menu constitue la colonne vertébrale de l’application. J’ai fait en sorte qu’elle soit la plus claire et intuitive possible. Une première version, basée sur de nombreux blocs if / else if, s’est révélée rapidement illisible. J’ai donc refactoré cette structure en sous-menus numérotés, permettant à l’utilisateur de naviguer facilement.

Un retour vers le menu précédent est toujours possible, ce qui évite tout risque de blocage et améliore l’expérience utilisateur.

C’est via ce menu que l’on accède à l’affichage et à la gestion des données.

### **2. La classe Affichage**

Cette classe permet d’afficher l’intégralité du jeu de données de manière simple et lisible. J’ai également ajouté une méthode affichageParticulier, afin de permettre à l’utilisateur de consulter un ensemble spécifique d’informations à tout moment, sans surcharger l'affichage par défaut.

Cette méthode s’est révélée particulièrement utile lors des phases de test, et j’ai fait en sorte qu’elle soit la plus générique possible pour pouvoir être réutilisée dans différents contextes.

### **3. La classe Action**

La classe Action regroupe les méthodes permettant d’afficher les entités liées à une entité particulière. Par exemple, les méthodes affichePersonneParticulier, afficheMediaParticulier ou afficheOrganisationParticulier suivent toutes une logique similaire, bien que je n’aie pas réussi à les factoriser complètement – ce que je considère comme un point à améliorer.

Le principe de base consiste à identifier les entités situées dans la quatrième colonne d’un fichier .tsv, en fonction de la valeur de la première colonne (ex. : organisation-media, personnage-media, etc.). Le fichier organisation-organisation a nécessité un traitement particulier, afin de pouvoir afficher les liens indirects masqués entre organisations.

### **4. La classe Achat**

La mise en place de cette classe a été l’une des parties les plus complexes du projet. Elle permet de simuler l’achat de parts d’un média ou d’une organisation.

Son fonctionnement est le suivant :

* Vérification que le vendeur possède un pourcentage suffisant.
* Mise à jour des parts du vendeur et de l’acheteur dans tous les fichiers concernés.
* Application des modifications via la classe Modification.

Ce choix de séparation entre Achat et Modification m’a paru pertinent, afin de garder un code clair et lisible. À noter également que, selon la logique métier que j’ai choisie, un individu ne peut pas acheter un média directement à une organisation : il doit obligatoirement passer par une autre organisation.

### **5. La classe Publication**

Cette classe a été ajoutée à la fin du projet, pour simplifier la gestion des publications d’événements. La méthode effectuerPublication, initialement incluse dans le menu, a été déplacée ici pour plus de clarté.

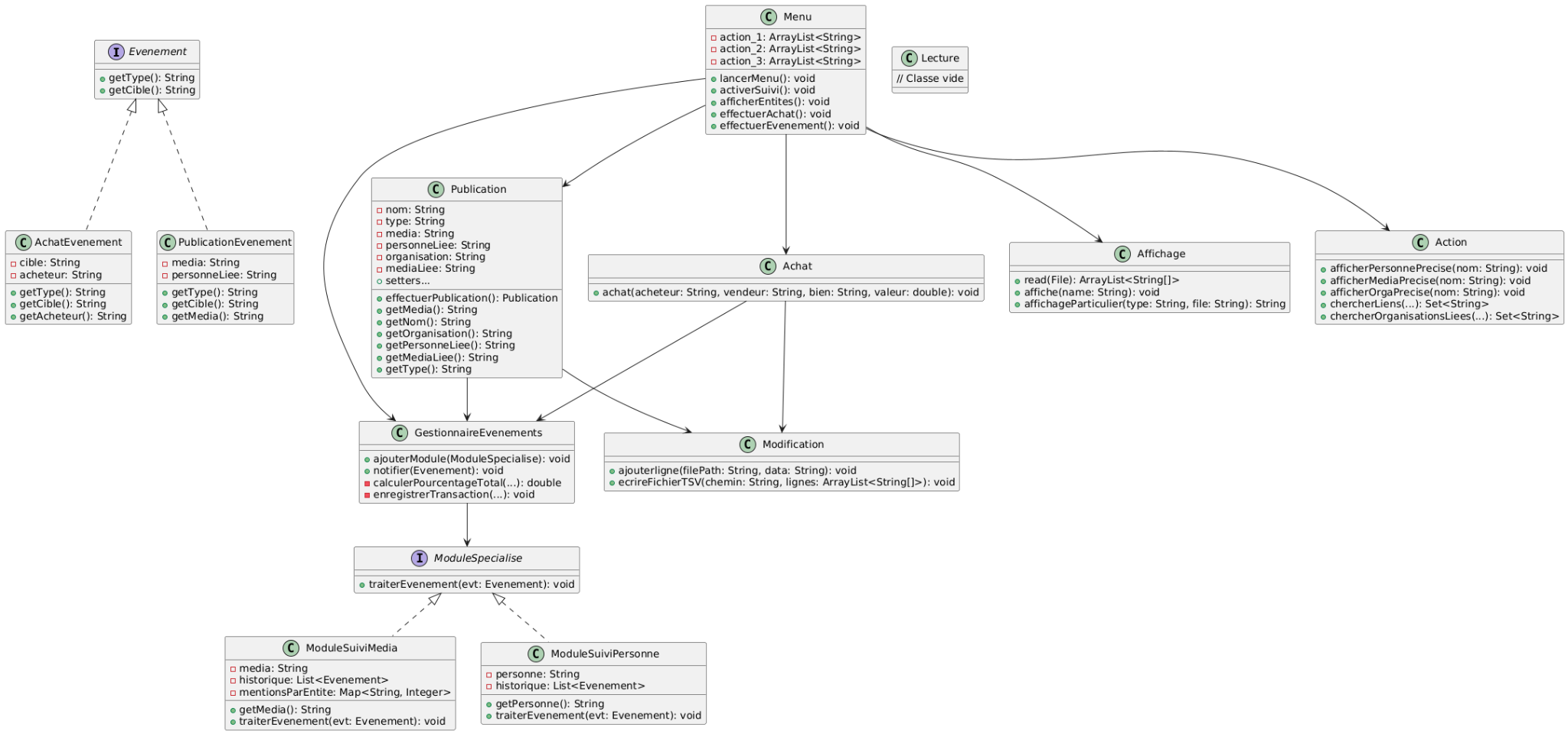
L’écriture dans les fichiers .tsv se fait via la méthode ajouterLigne de la classe Modification.

### **6. Gestion des événements**

Le suivi des événements est assuré par un ensemble de classes et d’interfaces, notamment ModuleSuiviMedia et ModuleSuiviPersonne. Ces modules sont appelés lors d’un achat ou d’une publication, et permettent de notifier l’utilisateur si l’entité concernée est surveillée.

L’objectif est de garantir une traçabilité et un contrôle sur les modifications apportées à la base de données. Cette architecture permet également une extension future avec d’autres types de modules de suivi.

## **Modélisation – Diagramme de Classe**



## **Difficultés Rencontrées**

Comme mentionné plus tôt, la gestion des achats a représenté un véritable défi, notamment en ce qui concerne les choix arbitraires à faire sur les fichiers à modifier. J’en reconnais les limites, mais j’ai fait au mieux avec les outils et les connaissances dont je disposais.

La gestion des alertes a également été difficile à mettre en place. Je n’ai compris le fonctionnement attendu qu’assez tard dans le développement, ce qui explique certaines lacunes dans la couverture fonctionnelle sur cette partie.

## **Informations Techniques pour l’Exécution du Projet**

La navigation dans les menus se fait en entrant les chiffres affichés (1, 2, 3, etc.). Lorsque l'utilisateur doit entrer une chaîne de caractères, une question lui est posée, lui laissant la possibilité de consulter les données disponibles pour l’aider à faire son choix.

Dans le cas où l’utilisateur entre une donnée inexistante, la saisie est refusée – sauf pour les publications, où l’utilisateur est libre d’ajouter toute entrée.

## **Conclusion**

Ce projet m’a permis de découvrir en profondeur la programmation en Java, ainsi que la gestion de fichiers de données structurées. J’ai pu mettre en œuvre de nombreux concepts appris au cours de ma formation, tout en développant une solution fonctionnelle, même si certaines parties restent perfectibles.

Merci d’avoir lu ce rapport.

**Maël Tallard**