

# Documentation Technique - Bibliothèque Numérique

## 1. Introduction

Ce projet a pour objectif de développer une application de gestion de bibliothèque numérique permettant :

- L'ajout, la consultation, la modification et la suppression de médias.
- L'inscription et la gestion des membres.
- La gestion des emprunts, retours et amendes éventuelles.
- La sauvegarde et restauration des données.

Le programme met en avant les principes d'architecture logicielle, le polymorphisme, la gestion d'exceptions et l'utilisation appropriée des collections Java.

## 2. Architecture Générale

Structure du projet :

bibliotheque\_numerique/

```
├── models/          (Classes métier : Media, Livre, CD, etc.)
├── exceptions/      (Exceptions personnalisées)
├── services/        (Logique métier : Bibliotheque, GestionnaireEmprunt, etc.)
├── ui/              (Interface Console : Menu)
├── utils/           (Fonctions utilitaires : Validation, dates)
└── Main.java        (Point d'entrée)
```

Choix d'architecture :

- **HashMap<Integer, Media>** pour les médias et membres → accès  $O(1)$ , idéal pour rechercher par ID.
- **ArrayList<Emprunt>** pour l'historique → nécessite l'ordre chronologique et itération simple.
- **Singleton (Bibliotheque)** : garantit une seule instance centrale de gestion.
- **Polymorphisme (Media + sous-classes)** permet d'afficher et gérer des médias différents de manière uniforme.

### 3. Description des Classes Principales

**\*\*Media (abstraite)\*\* :**

- Représente un média générique avec : id, titre, auteur, année, disponibilité.

Sous-classes :

- **\*\*Livre\*\*** : + nbPages, empruntable

- **\*\*Magazine\*\*** : + numéro édition, date de parution, empruntable

- **\*\*CD\*\*** : + durée (en minutes), empruntable

- **\*\*DVD\*\*** : + format, non empruntable

**\*\*Membre\*\*** :

- Stocke l'identité d'un utilisateur + liste des emprunts en cours + historique.

**\*\*Emprunt\*\*** :

- Associe un média à un membre avec dates d'emprunt/retour et éventuelle amende.

**\*\*Bibliotheque (Singleton)\*\*** :

- Coordonne la gestion globale : ajout, recherche, emprunts, retours, affichage.

### 4. Exceptions Personnalisées

- **\*\*MediaNonTrouveException\*\*** : ID invalide ou inconnu

- **\*\*MediaDejaEmprunteException\*\*** : tentative d'emprunt d'un média déjà emprunté

- **\*\*QuantiteInsuffisanteException\*\*** : utilisé si plusieurs exemplaires devaient être gérés

- **\*\*InvalidMediaException\*\*** : données incorrectes sur un média

- **\*\*MembreNonTrouveException\*\*** : ID membre non existant

Ces exceptions permettent un comportement contrôlé et des messages utilisateurs plus clairs.

## 5. Patterns Utilisés

-----

| Pattern | Rôle |

|-----|-----|

| Singleton | Instance unique de Bibliotheque |

| Polymorphisme | Appel `afficherDetails()` sans connaître le type réel |

| Interface | `Emprutable` garantit le contrat commun pour Livre/CD/Magazine |

| Découplage S-R | Chaque classe a une responsabilité claire (principe SRP SOLID) |

## 6. Guide d'Utilisation

-----

**\*\*Compilation :\*\***

```bash

javac models/\*.java exceptions/\*.java services/\*.java ui/\*.java utils/\*.java Main.java

```

**\*\*Exécution :\*\***

```bash

java Main

```

## 7. Difficultés Rencontrées et Solutions

- Conflits de noms et d'import → solution : organisation stricte des packages.
- Gestion du polymorphisme → affichage détaillé redéfini par média spécifique.
- Problèmes de saisie utilisateur → validation des entrées dans `ValidationUtils`.
- Sauvegarde/restauration → utilisation de `ObjectOutputStream` et `Serializable`.

## 8. Extensions Possibles

- Ajout d'un **systeme graphique (JavaFX)**.
- Ajout de **quantités multiples par média**.
- Intégration d'une **base de données (MySQL / SQLite)**.
- Notifications d'amendes par e-mail.
- Ajout d'un type **Livre Numérique (EPUB/PDF)**.