janvier 2026

# ECM 2A S7
# Programmation Objet

# Compte rendu
# Projet C++ : Gestion de bibliothèques


SURLEAU Guillaume - guillaume.surleau@centrale-med.fr
BREMONT Edouard - edouard.bremont@centrale-med.fr

Sommaire

# ETAPE 1 : Identification des objets et relation entre-eux

---

**__ Description de chaque classe __**

1. *class Bibliotheque*

La classe Bibliotheque représente l'entité qui gère le stock (de livres) et les adhérents.

- attributs : *nom*, *adresse*, *code*
- contenu : une liste de livres (*stock*), une liste d'adhérents (*inscrits*)

Responsabilités (Méthodes) : *afficher* ses livres (la totalité ou par catégorie), *acheter*, *supprimer*, *prêter* (si libre), *recupérer* un livre d'un adhérent, *demander* un livre à une autre bibliothèque, *rendre* des livres prêtés.

2. *class Livre* <<Abstract>>

La classe Livre représente l'objet central, le contenu des bibliothèques et des adhérents.
Comme il existe plusieurs types de livres spécifiques, cette classe sera considérée comme **abstraite** (ou virtuelle).

- attributs communs : *code* (unique), *auteur*, *titre*, *éditeur*, *numéro isbn*, *public visé* (adulte, ado, jeunesse, tout public), *état* (libre, emprunté, prêté), bibliothèque *propriétaire* (code)

Les spécialisations (classes fille) de la classe :

- *BandeDessinee* : possède un *dessinateur*
- *RecueilPoesie* : possède un *indicateur* (vers, prose, ou les deux)
- *Roman* : possède un *genre* (Littérature, Policier, SF, Amour, etc.)
- *PieceTheatre* : possède un *siècle*
- *Album* : possède un type d'*illustrations* (photos, dessins ou les deux)

3. *class Adherent*

La classe *Adhérent* représente l'utilisateur du service proposé par une bibliothèque (inscrit à).

- attributs : *nom*, *prénom*, *adresse*, *numéro d'adhérent*
- association : la *bibliothèque* à laquelle il est inscrit (code de la bibliothèque)
- état : liste des livres empruntés actuellement (*emprunts*), nombre *max d'emprunts* autorisés
- actions possibles (Méthodes) : peut *emprunter* un livre "'libre" s'il est autorisé, peut *rendre* un livre à la bibliothèque auquelle il est inscrit.

4. Autres classes utiles dans le projet

*class Erreur* (hérite d'exception)

La classe *Erreur* gère les exceptions i.e les actions impossibles qui peuvent se produire dans les interactions entre *Adherent*, *Bibliotheque* et *Livre* (exemple : un adhérent veut emprunter un livre déjà emprunté ou prêté). Utile pour tester le programme (voir *main.cpp*) en renvoyant un message personnalisé.

*class Liste* <<Template>>

La classe *Liste* est une classe Template permettant de gérer dynamiquement (supprimer, ajouter, chercher un élément) à la fois les livres (*stock*) d'une bibliothèque et la liste des adhérents (*inscrits*) qui sont des listes chaînées d'éléments différents.

**__ Description des relations existantes entre les objets __**

Relation d'héritage (pour les types de livre) :
Les classes `BandeDessinee`, `RecueilPoesie`, `Roman`, `PieceTheatre`, et `Album` héritent de la classe `Livre`.

Relation d'Agrégation :

- Entre les bibliothèques et les livres : Une *Bibliotheque* possède une collection de *Livre*.
- Entre les bibliothèques et les adhérents : Une *Bibliotheque* possède une liste d'inscrits (*Adherent*).
- Entre les adhérents et les livres : Un *Adherent* possède une liste de *Livre* qu'il emprunte.

Relation d'Association :

- *Adherent* — *Bibliotheque* : Un *Adherent* est inscrit à une *Bibliotheque* spécifique.
- *Adherent* — *Livre* : Un *Adherent* emprunte une liste de *Livre* tout en respectant son quota et si l'état des livres ("Libre" ou non).
- *Bibliotheque* — *Livre* : Une *Bibliotheque* achète des livres disponibles en vente.
- *Bibliotheque* — *Bibliotheque* : Les bibliothèques interagissent entre elles (échange, prêt inter-bibliothèques). Une bibliothèque peut "demander un livre à une autre" et doit "rendre ceux qui lui sont prêtés et non empruntés par ses adhérents".

Relation de dépendance :
La classe template *Liste* est utilisée par *Bibliotheque* et *Adherent* pour gérer dynamiquement les collections de livres et d'adhérents.

**__ Vision globale du déroulement d'une exécution __**

On imagine que dans le scénario (*main.cpp*), il y a deux ou trois bibliothèques (Bib1, Bib2, Bib3) existantes, trois adhérents (A1, A2, A3) chacun à une bibliothèque (respectivement Bib1, Bib2, Bib3) et une quinzaine de livres (L1, ..., L15) disponibles à l'achat.

1. Les bibliothèques commencent par acheter les livres :

Bib1 achète L1 à L5
Bib2 achète L6 à L10
Bib3 achète L11 à L15
Ensuite on peut afficher les stocks de chaque bibliothèque (la totalité ou par catégorie) pour voir si le contenu est correct.

2. Les adhérents commencent à emprunter des livres chacun respectivement à la bibliothèque à laquelle il est inscrit :

Adh1 emprunte L1 et L4
Adh2 emprunte L7, L8 et L10
Adh3 emprunte L12

3. Les adhérents peuvent rendre des livres pour tester l'état des livres de chaque bibliothèque

Adh1 rend L1 à Bib1
Adh2 rend L7 et L8 à Bib2
Adh3 rend L12 à Bib3

4. Les bibliothèques peuvent supprimer certains livres qui partent au pilon ou sont perdus par les adhérents

ex : Bib1 supprime L5 (part au pilon) et L4 (Adh1 a perdu le livre)

5. Les bibliothèques se prêtent des livres entre-eux si disponible :

Bib1 demande à Bib2 les livres L8 et L10. L8 est prêté mais L10 n'est pas prêté car toujours emprunté par Adh2.
Bib3 demande à Bib 1 les livres L2 et L3 qui seront prêtés.

# ETAPE 2 : Réalisation en C++

---

## __ Le travail réalisé __

Pour accéder à tous les fichiers, suivez le lien suivant où vous trouverez le répertoire github du projet : repertoire du projet.
Une fois dans le répertoire, le projet se trouve dans le dossier *projet_cpp*.

Le diagramme UML correspondant, étant assez grand, est consultable dans le README du dossier projet.

Nous avons réalisé une première version sans saisie externe qui permet de tester notre programme dans le terminal.
Le programme principal se trouve dans le fichier *main.cpp* : où son exécution permet de tester l'instanciation des objets et chaque fonctionnalité utile pour une gestion de bibliothèques en respectant le cahier des charges du projet (paramètres à donner pour une fonctionnalité, contraintes) :

Instances :
- Création des bibliothèques
- Création des livres
- Création des adhérents

Fonctionnalités :
- Achat de livres (pour les bibliothèques)
- Affichage du stock (complet ou par catégorie) (pour les bibliothèques)
- Emprunt de livre (pour les adhérents)
- Retour de livre (pour les adhérents)
- Suppression de livre (pour les bibliothèques)
- Echange de livre (entre bibliothèques)

## __ Répartition du travail __

Globalement, nous avons travaillé en équipe pour implémenter chaque fonctionnalité.
En discutant ensemble, nous avons pu trouver les différents cas possibles pour chaque action et ainsi les tester dans notre programme principal.