Compte Rendu n°9 du Génie Logiciel

DERMOUCHE Mohammed Ryad
Université Paris Cité
L3 Informatique et Applications UFR Maths-Info

Première partie : Gestion des individus dans le cours de Génie Logiciel

1. Description des classes

Ce chapitre présente la modélisation UML des individus prenant part au cours de Génie Logiciel. Le diagramme de classe met en avant trois classes principales : **Humain**, **Étudiant**, et **Enseignant**, ainsi que leurs attributs, méthodes, et relations.

1.1 Classe Humain

- **Attributs :**
  - `nom : String` : Représente le nom de l'individu.
  - `IDMoodle : String {id}` : Identifiant unique de l'utilisateur dans Moodle.
- **Méthodes :**
  - `operation1(params) : returnType` : Méthode générique publique.
  - `operation2(params)` : Méthode générique protégée.
  - `operation3()` : Méthode générique protégée.
- **Particularités :** La classe `Humain` est une classe abstraite spécialisée en deux sous-classes : **Étudiant** et **Enseignant**. Cette relation est marquée comme `{incomplete, disjoint}`, ce qui signifie qu'une instance ne peut appartenir qu'à une seule sous-classe.

1.2 Classe Étudiant

- **Attributs :**
  - `groupe : Integer` : Groupe auquel l'étudiant appartient.
  - `notesTD : Integer[*]` : Liste des notes obtenues pour les travaux dirigés.
  - `notesQuiz : Integer[*]` : Liste des notes des quiz.
  - `noteExam : Integer` : Note obtenue à l'examen.
- **Méthodes :**
  - `suivreLeCours()` : Permet de suivre un cours.
  - `suivreLeTD()` : Permet de suivre un TD.

- `apprendre()` : Action d'apprentissage.
- `reviser()` : Préparation avant un examen.

1.3 Classe Enseignant

- **Attributs :**
  - `groupes : Integer[1..4]` : Groupes supervisés par l'enseignant (limité entre 1 et 4).
- **Méthodes :**
  - `donnerCours()` : Permet d'enseigner un cours.
  - `encadrerTD()` : Encadrement des travaux dirigés.
  - `corriger()` : Correction des travaux ou examens (protégé).
  - `preparer()` : Préparation de cours ou TD (protégé).

2. Relations entre les classes

- **Héritage :** `Étudiant` et `Enseignant` héritent de la classe abstraite `Humain`.
- **Association :** Une relation bidirectionnelle existe entre `Étudiant` et `Enseignant`, où :
  - `Étudiant` : Apprend de l'`Enseignant`.
  - `Enseignant` : Enseigne à un ou plusieurs `Étudiants`.

3. Diagramme de Classe

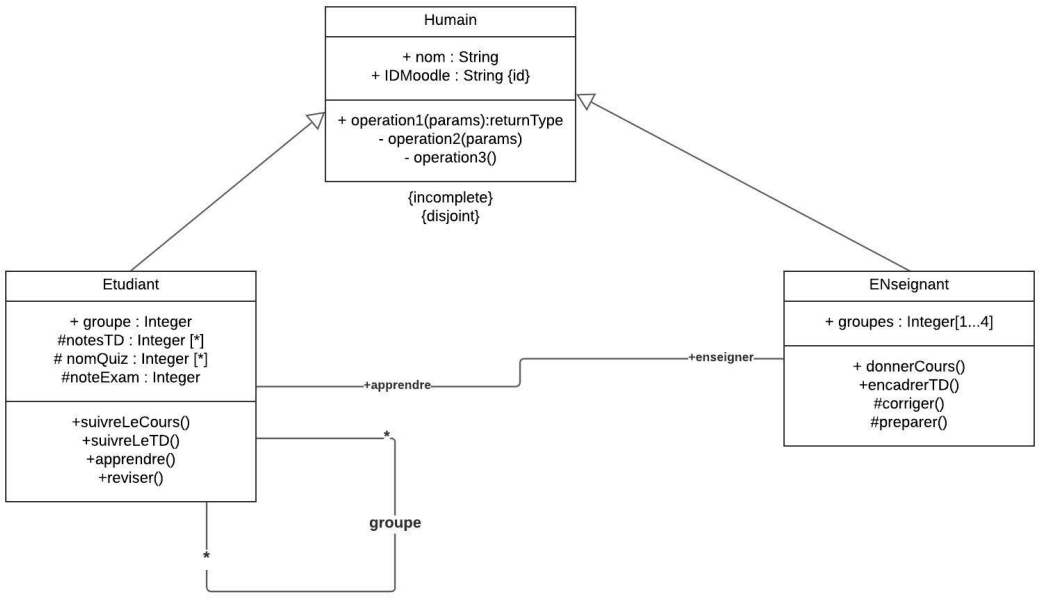

Figure 1: Diagramme de classe UML des individus dans le cours de Génie Logiciel.

Deuxième partie : Modélisation des Classes pour la Bibliothèque Municipale

1. Introduction

Cette section présente le diagramme de classe pour le système de gestion numérique d'une bibliothèque municipale. Ce modèle a été conçu en prenant en compte les interactions étudiées dans les cas d'utilisation et les diagrammes de séquence des semaines précédentes. Les classes principales, leurs attributs, méthodes, et relations sont représentés pour fournir une vue d'ensemble complète du système.

2. Description des classes

2.1 Bibliothèque

La classe `Bibliothèque` représente l'entité principale qui gère les livres, les emprunteurs, les bornes, et les interactions avec la base de données.

- **Attributs :**
  - `nom : String` : Nom de la bibliothèque.
- **Méthodes :**
  - `ajouterLivre(livre : Livre) : void` : Ajoute un livre à la bibliothèque.
  - `retirerLivre(livre : Livre) : void` : Retire un livre de la bibliothèque.

2.2 Livre

La classe `Livre` modélise les informations essentielles sur chaque livre.

- **Attributs :**
  - `auteur : String` : Auteur du livre.
  - `éditeur : String` : Éditeur du livre.
  - `ISBN : String` : Numéro ISBN unique.
  - `dateRendu : Date` : Date prévue de retour.
  - `nbRenouvellement : Integer` : Nombre de renouvellements possibles.

2.3 Emprunteur et Spécialisations

La classe `Emprunteur` représente les utilisateurs qui empruntent des livres. Elle est subdivisée en deux classes spécialisées : - `Étudiant` : Peut emprunter jusqu'à 5 livres. - `Personnel` : Peut emprunter jusqu'à 10 livres.

- **Attributs :**
  - `numero(id) : Integer` : Identifiant unique.
  - `nom : String` : Nom de l'emprunteur.
  - `nbLivresEmpruntés : Integer` : Nombre de livres actuellement empruntés.
  - `nbLivresEmpruntables : Integer` : Limite de livres empruntables (définie dans les sous-classes).
- **Méthodes :**
  - `emprunter(livres : Livre[nbLivresRestants]) : void` : Emprunte des livres.
  - `rendre(livre : Livre) : void` : Rend un livre.

2.4 Bibliothécaire

La classe `Bibliothécaire` représente le personnel responsable de la gestion des livres et des emprunts.

- **Attributs :**
  - `nom : String` : Nom du bibliothécaire.
  - `dateEmbauche : Date` : Date d'embauche.
  - `typeContrat : String` : Type de contrat (CDD, CDI, etc.).
- **Méthodes :**
  - `ajouterRéférence(livre : Livre) : void` : Ajoute un nouveau livre dans la base.
  - `enregistrerRetour(livre : Livre) : void` : Enregistre un retour de livre.

3. Relations entre les classes

Les relations suivantes sont identifiées :

- La `Bibliothèque` est en association avec plusieurs livres, emprunteurs, et bornes.
- La classe `Livre` est liée à la classe `Emprunteur` via une relation d'association.
- La classe `Emprunteur` est spécialisée en `Étudiant` et `Personnel` grâce à une relation d'héritage.
- La `Borne` implémente l'interface `Enregistrer` pour gérer les emprunts et retours de livres.

4. Diagramme de Classe

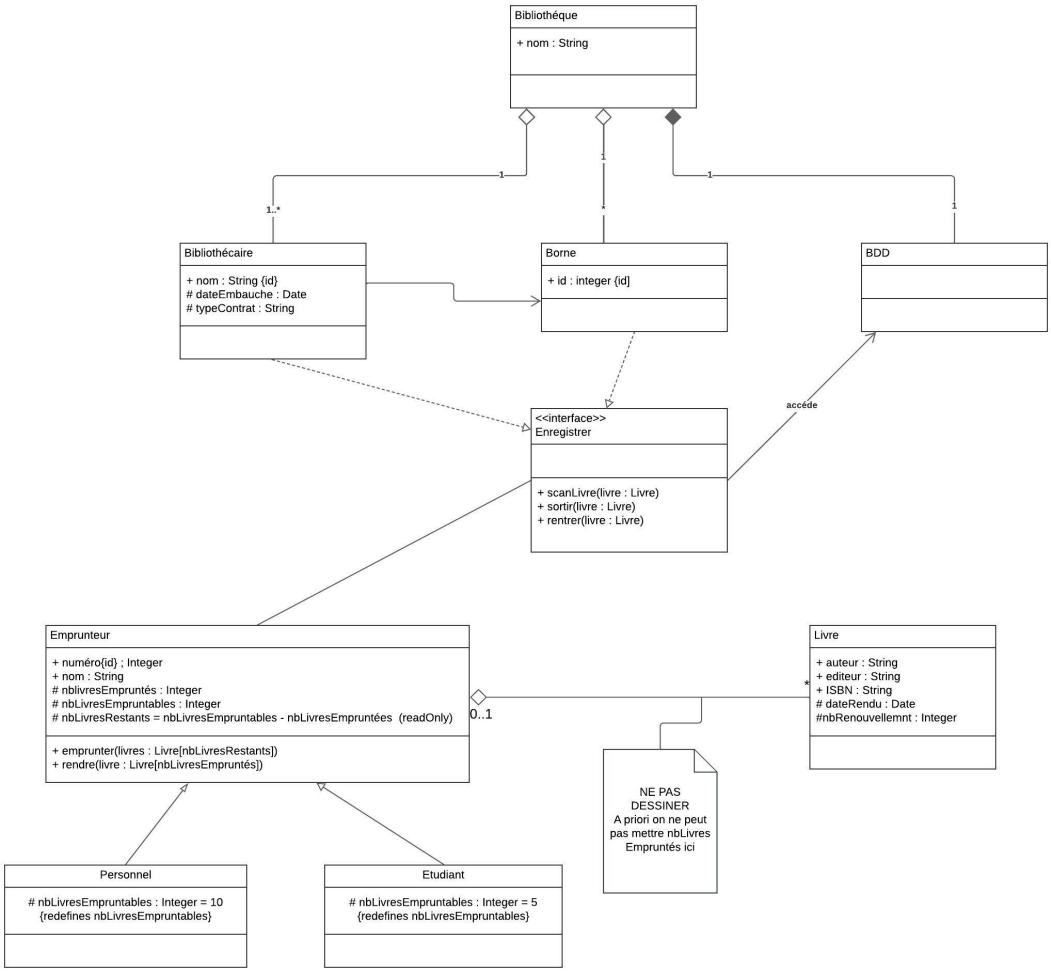

Figure 2: Diagramme de classe UML pour la gestion d'une bibliothèque municipale.

5. Réflexion sur la modélisation

Le modèle aurait pu être construit différemment en débutant par un diagramme de classe pour obtenir une vue d'ensemble claire des entités et des relations. Cependant, l'approche actuelle, en commençant par les cas d'utilisation et les diagrammes de séquence, a permis de garantir une couverture fonctionnelle précise.

5.1 Complexité du modèle

La complexité du modèle est modérée. Les relations entre les emprunteurs, les livres, et la bibliothèque sont simples et compréhensibles. L'utilisation d'une interface pour gérer les emprunts et restitutions via les bornes centralise les fonctionnalités et réduit la complexité.

5.2 Informations manquantes

Quelques améliorations pourraient enrichir le modèle :

- Gestion des exemplaires multiples pour un même livre.
- Gestion des pénalités pour retard dans les restitutions.
- Fonctionnalités de réservation de livres indisponibles.
- Suivi des stocks globaux et des emprunts actifs.

Ces ajouts permettraient une couverture fonctionnelle plus complète et une meilleure gestion des cas réels rencontrés dans une bibliothèque.