École Centrale Lyon

UE INF
TC2
Rapport

Rapport TD2 : Bibliothèque

Élèves :
Axel Volteau
Céline Zareba

Enseignant :
Daniel Muller
Groupe D2a

11 octobre 2023

Table des matières

1 Diagramme UML **2**

2 Classe bibliothécaire **2**
2.1 Définition de la classe 2
2.2 Relation avec les autres classes 3

3 Classe Conservateur **3**

4 Modification des autres classes **3**

5 Programme test **4**

6 Conclusion **4**

1 Diagramme UML

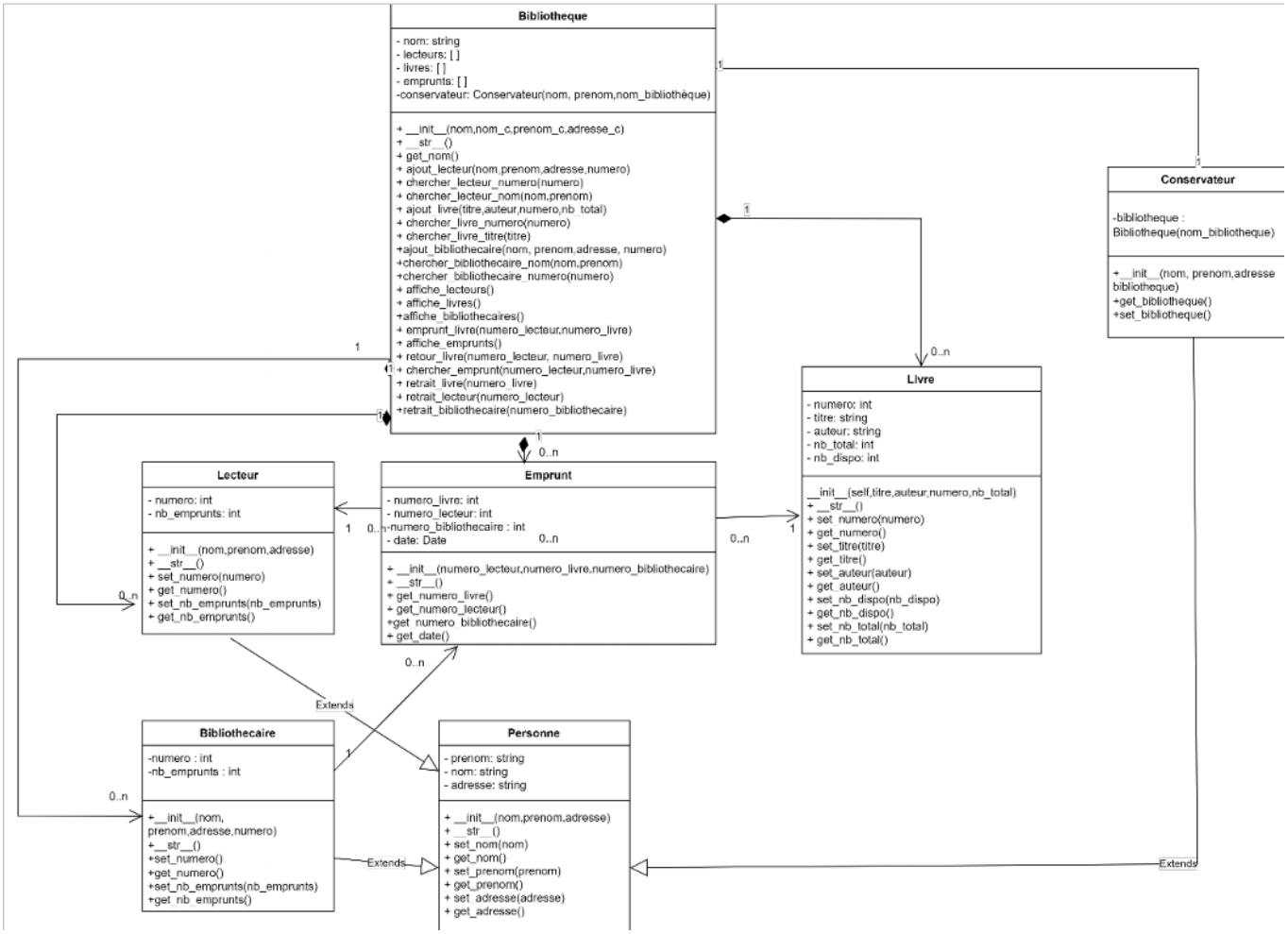

FIGURE 1 – Diagramme UML complet

Le diagramme UML ci-dessus à été complété selon le cahier des charges, avec l'ajout des classes Conservateur et Bibliothécaires, ainsi que l'ajout d'attribut à Bibliothèque, emprunt et de méthodes pour ces mêmes classes.

2 Classe bibliothécaire

2.1 Définition de la classe

On a choisi d'identifier la bibliothécaire par son numéro car celui-ci est unique mais on précise dans le constructeur son nom et son prénom et son adresse pour plus de lisibilité 'réelle' car c'est une personne. De la même façon, on identifie le conservateur par sa bibliothèque associée mais on précise dans le constructeur son nom et son prénom et son adresse. Cela permet aussi de rester cohérent avec les autres constructeurs des classes héritant de Personne. Quand on ajoute un bibliothécaire, on choisit de considérer qu'il n'a effectué aucun emprunt, comme s'il venait d'être embauché.

```python
class Bibliothecaire(Personne):
    def __init__(self,nom,prenom,adresse,numero):
        Personne.__init__(self,nom,prenom,adresse) #utilisation du constructeur de Personne
        self.set_numero(numero)
        self.__nb_emprunt=0 #par defaut le bibliothécaire n'a réalisé aucun emprunt
```

FIGURE 2 – Constructeur de la classe Bibliothécaire en python

2.2 Relation avec les autres classes

On choisit un lien de composition entre les bibliothécaires et la bibliothèque car on considère qu'une bibliothécaire ne peut travailler que dans une seule bibliothèque et que plusieurs bibliothécaires travaillent dans une même bibliothèque.

Nous avons pris la décision d'associer un emprunt à un bibliothécaire, si bien que le bibliothécaire qui à réalisé l'emprunt est le seul à pouvoir enregistrer le retour. De ce fait, à l'image du lecteur il a lui aussi un attribut nb_emprunts qui compte le nombre d'emprunts en cours réalisé par le bibliothécaire

3 Classe Conservateur

Le conservateur est définie comme une Personne, lié à un unique bibliothèque d'ou le lien d'association entre les 2 classes.

```python
class Conservateur(Personne):
    def __init__(self,nom,prenom,adresse,nom_bibliotheque):
        Personne.__init__(self,nom,prenom,adresse) #utilisation du constructeur de Personne
        self.set_bibliotheque(nom_bibliotheque)
```

FIGURE 3 – Constructeur de la classe Conservateur en python

4 Modification des autres classes

On modifie tous les programmes qui concernent les emprunts car la classe Emprunt possède à présent un attribut en plus qui est le bibliothécaire associé à un emprunt On a gardé la cohérence dans le nom des programmes avec un typage (chercher_type_attribut(), retrait_type(),...)

Voici un exemple du constrcteur d'Emprunt modifié :

```python
class Emprunt:
    def __init__(self,numero_lecteur,numero_livre,numero_bibliothecaire):
        self.__numero_lecteur = numero_lecteur
        self.__numero_livre = numero_livre
        self.__date = date.isoformat(date.today())
        self.__numero_bibliothecaire = numero_bibliothecaire
```

FIGURE 4 – Constructeur de la classe Emprunt modifié en python

5 Programme test

Pour le programme de test, les tests ont été réalisé sur le même programme que celui fourni dans le TD2 et ont été fait sur le modèle des autres tests, où chaque nouvelle fonction ou chaque fonction modifié à été testée. Certains tests fournis dans le td2 on été modifié afin de coller aux modifications réalisés comme pour les emprunts par exemples. Ainsi on a vérifié la compatibilité de nos ajouts avec le reste du travail effectué auparavant.

Voici un exemple de test modifié pour correspondre aux méthodes définies et modifiés vis à vis du TD2 :

```
bib = b.chercher_bibliothecaire_numero(106)
if bib != None:
    print('Bibliothécaire trouvé :',bib)
else:
    print('Bibliothécaire non trouvé')

# Quelques emprunts
print('\n--- Quelques emprunts :')
print('-------------------------------')
b.emprunt_livre(1,101,1)
b.emprunt_livre(1,104,2)
b.emprunt_livre(2,101,3)
b.emprunt_livre(2,105,1)
b.emprunt_livre(3,101,2)
b.emprunt_livre(3,104,3)
b.emprunt_livre(4,102,1)
b.emprunt_livre(4,103,2)

# Affichage des emprunts, des lecteurs des livres et des bibliothecaires
print('\n--- Liste des emprunts :')
print('-------------------------------')
b.affiche_emprunts()
print('\n--- Liste des lecteurs :')
print('-------------------------------')
b.affiche_lecteurs()
print('\n--- Liste des livres :')
print('-------------------------------')
b.affiche_livres()
print('\n--- Liste des bibliothécaires :')
print('-------------------------------')
b.affiche_bibliothecaires()
```

FIGURE 5 – Exemple de tests adaptés aux méthodes définies

6 Conclusion

Au long de ce travail, on a pu constaté que l'ajout d'exigences au cahier des charges exigait un travail sur toutes les classes définies précédemment et non uniquement sur les classes à définir.