Litzler Samuel

BTS "Systèmes Numériques"

1ère Année

Le 14 / 01 / 2019

Année 2018 – 2019

Mini projet Bibliothèque

Objectif :

Nous devons créer une application pour pouvoir gérer les adhérents, ouvrages et emprunts dans une bibliothèque.

I. Etude du système

Entretien :

– J'ai fait appel à vous pour informatiser notre bibliothèque. En effet, nous commençons à avoir un certain nombre de livres et d'adhérents, et il devient difficile pour nous de suivre les prêts et difficile pour les adhérents de rechercher des livres.

– *Pourriez-vous me décrire la façon dont vous fonctionnez actuellement ?*

– Nous fonctionnons avec des notices papier. Une notice est affectée à chaque livre et insérée contre la couverture à l'intérieur du livre. Quand une personne emprunte un livre, elle donne la notice du livre à un assistant qui la range dans le fichier des emprunts. Nous avons aussi une fiche par adhérent. Il faut donc noter sur la fiche de l'adhérent les livres qu'il emprunte et la date de retour lorsqu'il les rend.

– *Qu'y-a-t-il d'écrit sur une notice ?*

– Le titre du livre, l'auteur et l'éditeur par exemple. Mais ça dépend un peu des notices. Quand une personne emprunte un livre, on écrit aussi son nom, son prénom et la date du prêt.

– *Pourquoi dites-vous : « ça dépend un peu des notices » ?*

– Parce qu'il y a plusieurs types de notice en fonction des documents. Nous avons des romans, des bandes dessinées, des livres sur la culture, comme l'histoire, l'art, etc.

– *Pouvez-vous me montrer quelques notices ?*

Notice de livre

Titre : La stratégie de l'ombre
Auteur : Orson Scott Card
Éditeur : L'Atalante
N° ISBN : 2-84172-185-X

Nom et prénom	Date
Mounissamy Viasse	02/05/2004
Abdoul lattif Bilal	05/10/2004
Boueyre Igor	22/10/2004
Khalil Safia	12/12/2004

Notice d'ouvrage d'art

Titre : Doisneau : Paris
Auteur : Robert Doisneau
Éditeur : Flammarion
N° ISBN : 2080304917

Nom et prénom	Date
Tran Richard	02/05/2004
Sessi Amedie	05/10/2004
Oliveira Romeo	22/10/2004
Mebarkia Soufyane	12/12/2004

Notice de bande dessinée

Titre : Londres
Scénario : Régis Loisel
Dessin : Régis Loisel
Couleurs : Régis Loisel
Collection :
Éditeur : Vents d'Ouest
N° ISBN : 2-86967-115-6

Nom et prénom	Date
Fruitier Cédric	02/05/2005
Guyen Noorullah	05/10/2005
Robin Remi	22/10/2006
Souna Karim	12/11/2006

– *Quels sont exactement les différents types de documents que vous possédez?*
– Des romans, des bandes dessinées, des ouvrages sur l'art et l'histoire, des guides de voyage et des revues qui ne peuvent pas être empruntés.
– *Le système doit-il aussi gérer les revues?*
– Oui, pour connaître notre fond, et pour permettre de faire des recherches.
– *Qu'attendez-vous du système?*
– Qu'il permette de mémoriser et de gérer toutes nos notices papier. Qu'il permette d'effectuer des recherches sur notre fond. Qu'il permette de gérer les emprunts.

– *Tout le monde peut-il emprunter des ouvrages?*
– Oui, à condition d'être abonné à la bibliothèque.
– *Donc le système doit aussi gérer les abonnés?*
– Euh ... oui.
– *Un adhérant a-t-il accès au système?*
– Oui, il doit pouvoir effectuer des recherches pour savoir si un ouvrage existe dans la bibliothèque et s'il est disponible. Même un simple visiteur doit pouvoir le faire.
– *Toutes les autres interactions avec le système sont réalisées uniquement par le bibliothécaire?*
– Oui ... ou un assistant. Un assistant doit pouvoir gérer les emprunts et les retours. Il doit aussi pouvoir effectuer des recherches et savoir, le cas échéant, qui à emprunté un ouvrage en cours de prêt. Moi, je dois pouvoir, enplus, modifier le fond documentaire. J'aimerais aussi pouvoir afficher la liste des ouvrages qui auraient dû être rendus et ne le sont pas encore, et qui les a empruntés.
– *Quelle est la durée maximale d'un prêt?*
– Ça dépend, un mois pour les romans et les autres livres, trois semaines pour un guide de voyage et deux pour une bande dessinée.
– *Combien un adhérent peut-il emprunter d'ouvrages?*
– Au maximum trois romans, deux guides de voyage et cinq bandes dessinées. Mais pas plus de cinq ouvrages en tout.
– *Bon, voyez-vous des choses à rajouter*
– Oui, j'aimerais bien qu'un assistant ou moi-même puissions spécifier sur une notice l'état d'un ouvrage. Par exemple avec trois niveaux : bon, moyen et abîmé. Ceci m'aiderai beaucoup pour le remplacement des exemplaires.

Notre réflexion pour pouvoir mettre en place un diagramme en classe est :

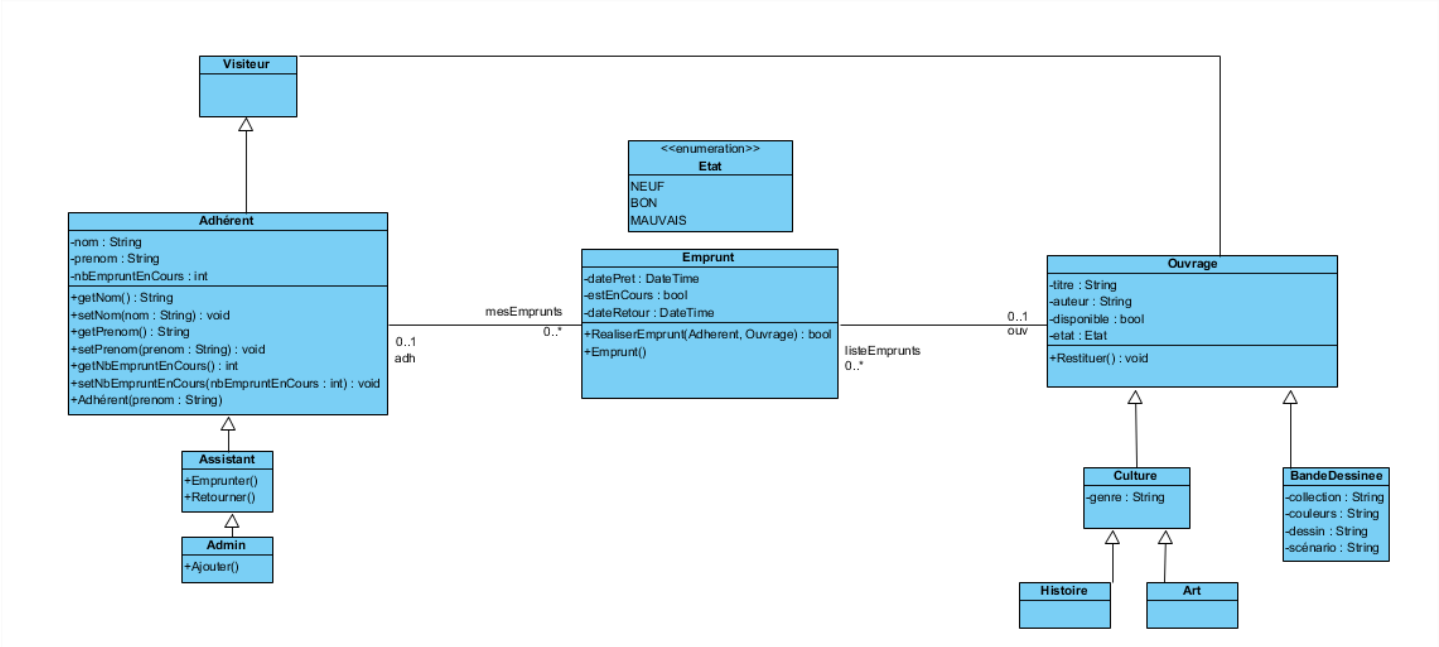
Pour organiser le diagramme des classes, on suit cette règle : les noms sont des attributs ou des classe et les verbes sont les méthodes ou des associations.

Classes candidate (attributs entres parenthèses)

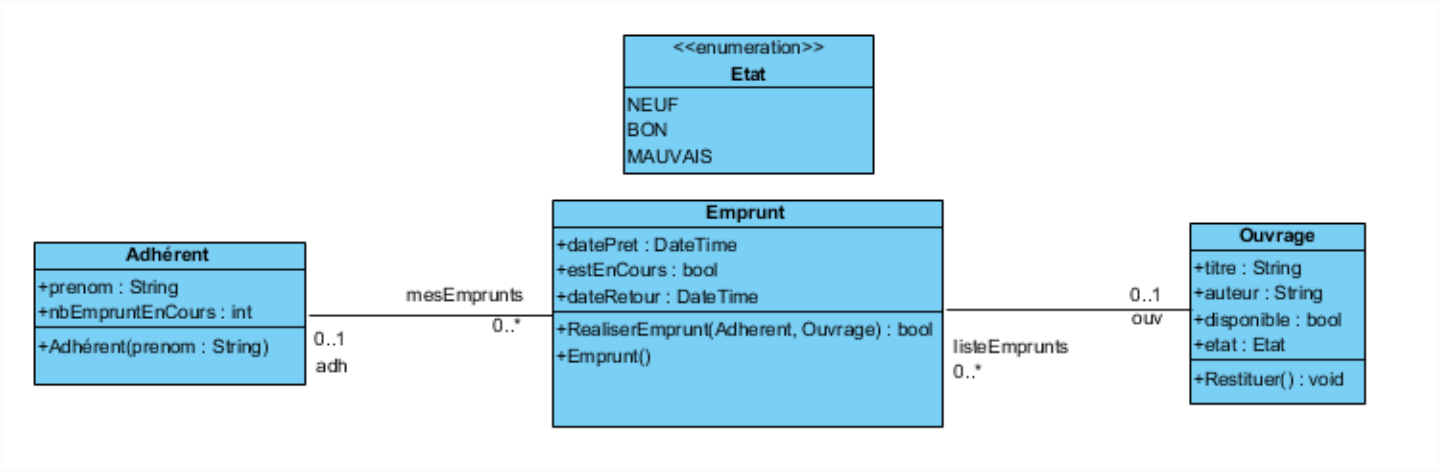
- Livre (titre, auteur, éditeur, N°ISBN, état du livre, [disponibilité pour rénovation])
- Adhérent (nom, prénom, nombre emprunt, [tel, email, naissance,])
- ~~Prêt~~
- ~~Notice~~ (affecté à chaque livres)
- ~~Fiche du livre~~ (lié à adhérent)
- Emprunt (idem prêt) (date du prêt, retour, durée possible emprunt)
- ~~Fiche adhérent~~
- Date (existe : DateTime)
- Roman
- BD (T,A,E,N°, collection, couleurs, dessin, scénario)
- Culture : ouvrage art, histoire, voyage, revues
- Art : déclinaison de livre
- Visiteur
- Bibliothécaire peut modifier le fond, afficher le retard
- Assistant gère les emprunts, retour

II. Diagramme de classe

Voici donc notre diagramme de classe suite à l'analyse.

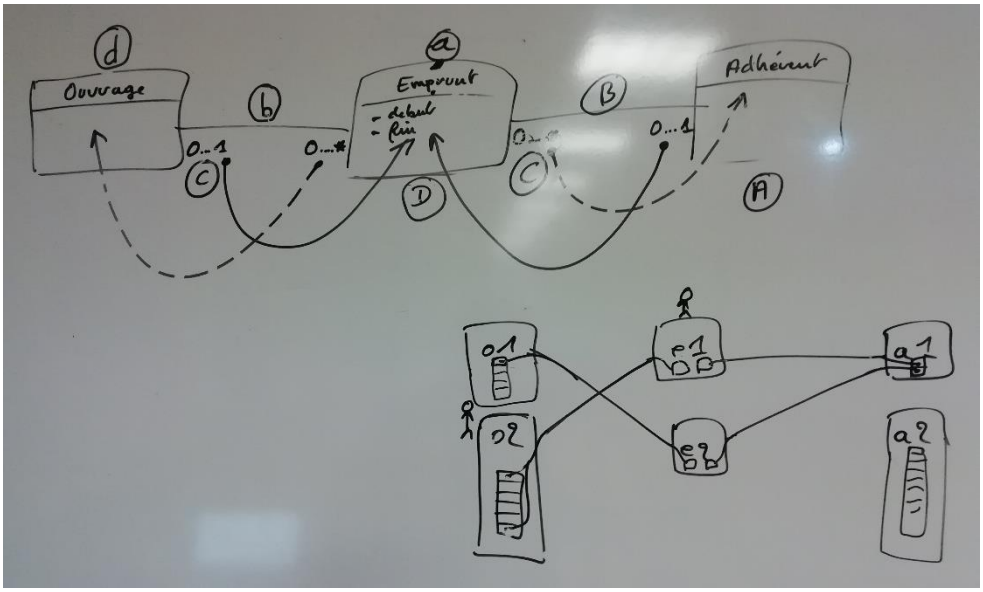


Mais nous allons juste nous concentrer sur le moteur avec ces 3 classes : **Adhérent , Emprunt, Ouvrage.**
Nous simplifions aussi les classes en passant les attributs en public pour ne pas gérer les Getter et Setter.



On fait un schéma pour expliquer le fonctionnement des multiplicitées et des rôles.

Les rôles et multiplicités sont codés dans la classe opposée (démonstration avec les flèches)



III. Programme

Classe Emprunt

```
class Emprunt
{
    // date time
    public DateTime datePret;
    public DateTime dateRetour;
    public bool estEnCours;

    // les rôles
    public Member memb;
    public Ouvrage ouv;

    public Emprunt() { }

    // constructeur par paramètre (si appelé, l'object est crée même s'il est vide il existe)
    public bool RealiserEmprunt(Member x, Ouvrage y)
    {
        if (y.disponible == true)
        {
            // on s'ajoute à la liste d'emprunts du membre x
            x.mesEmprunts.Add(this);
            // l'adhérent de cet emprunt est x
            this.memb = x;

            // on s'ajoute à la liste d'emprunts de l'ouvrage y
            y.listeEmprunts.Add(this);
            // l'ouvrage de cet emprunt est y
            this.ouv = y;

            y.disponible = false;

            this.datePret = DateTime.Now;
            this.estEnCours = true;
            x.nbEmpruntEnCours++;

            return true;
        }
        else { return false; }
    }
```

Classe Adherent

```
    class Adherent
    {
        // Attributs
        public string prenom;
        public int nbEmpruntEnCours;
        //                    \ Rôles /
        public List<Emprunt> mesEmprunts = new List<Emprunt>();

        // constructeurde base à 0 emprunt
        public Adherent() { }

        // constructeur
        public Adherent(String prenom)
        {
            this.prenom = prenom;
            this.nbEmpruntEnCours = 0;
        }
```

Classe Ouvrage

```
enum Etat { Neuf, Bon, Mauvais }

    class Ouvrage
    {
        // Attributs
        public string titre;
        public string auteur;
        public bool disponible;
        public Etat etat;

        //                      \ Rôle /
        public List<Emprunt> listeEmprunts = new List<Emprunt>();

        // constructeur
        public Ouvrage()
        {
            this.etat = Etat.Neuf;
            this.disponible = true;
        }
        public Ouvrage(String titre, String auteur)
        {
            this.titre = titre;
            this.auteur = auteur;
            this.etat = Etat.Neuf;
            this.disponible = true;
        }

        public void Restituer()
        {
            listeEmprunts.Last().estEnCours = false;
            listeEmprunts.Last().dateRetour = DateTime.Now;
            listeEmprunts.Last().memb.nbEmpruntEnCours--;

            this.disponible = true;
        }

    }
```

Programme principale

```
class Program
{
    static void Main(string[] args)
    {

        /* Liste pour crer des ouvrages */
        // templates List<>
        //***    List<Ouvrage> fondOuvrages = new List<Ouvrage>();
        // avec le constructeur par défaut
        //***    fondOuvrages.Add(new Ouvrage() { titre = "tintin", auteur = "Hergé"});
        // avec le contructeur par paramètre
        //***    fondOuvrages.Add(new Ouvrage("asterix", "Boscinny"));
        // boucle pour afficher les titre des objects
        //***foreach (Ouvrage ouvrage in fondOuvrages)
        //***{
        //***Console.WriteLine(ouvrage.titre);
        //***}


        /* Nouvelle liste pour crer des membres */
        //***List<Member> adherent = new List<Member>();
        //***adherent.Add(new Member("Gerald"));
        //***adherent.Add(new Member("Alexandre"));

        //***foreach (Member membre in adherent)
        //***{
        //***Console.WriteLine(membre.prenom);
        //***}

        Ouvrage ouvrage1 = new Ouvrage("tintin", "Hergé"); // neuf par défaut
        ouvrage1.etat = Etat.Bon; // modifier l'état
        Ouvrage ouvrage2 = new Ouvrage("asterix", "Goscinny");

        Adherent membre1 = new Adherent("Samuel");
        Adherent membre2 = new Adherent("Gerald");

        // Emprunt emprunt1 = new Emprunt(membre1, ouvrage2);   // si on défini un constructeur
        // Emprunt emprunt2 = new Emprunt(membre1, ouvrage1);

        List<Emprunt> listeEmprunts = new List<Emprunt>();
        // avec des méthodes
        Emprunt emprunt1A = new Emprunt();
        if (emprunt1A.RealiserEmprunt(membre1, ouvrage2) == true)
            listeEmprunts.Add(emprunt1A);

        // ne va jamais dans le if car ouvrage2 est déjà emprunté
        Emprunt emprunt2A = new Emprunt();
        if (emprunt2A.RealiserEmprunt(membre1, ouvrage2) == true)
            // ajouter à la liste
            listeEmprunts.Add(emprunt2A);


        // date retour 1 secondes après
        Thread.Sleep(1000);
        ouvrage2.Restituer();

        // date retour 2 secondes après
        Thread.Sleep(1000);
        ouvrage1.Restituer();

    }
}
```