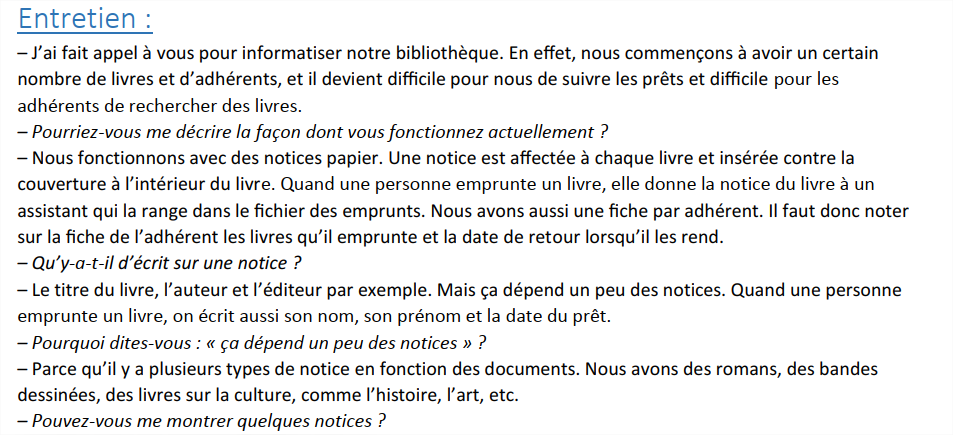
|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | | --- | | Litzler Samuel |   **BTS "Systèmes Numériques"** | |
|  | **1ère Année** |
|  | **Le 14 / 01 / 2019** |
|  | **Année 2018 – 2019** |

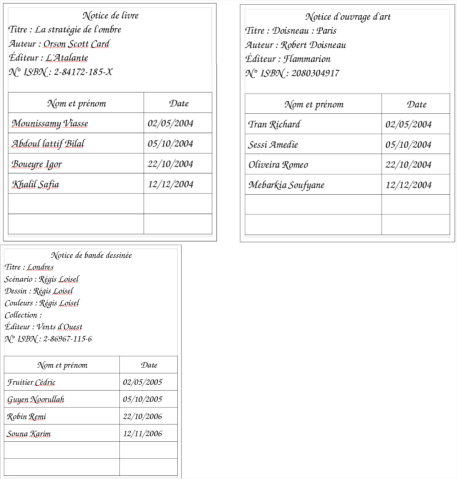
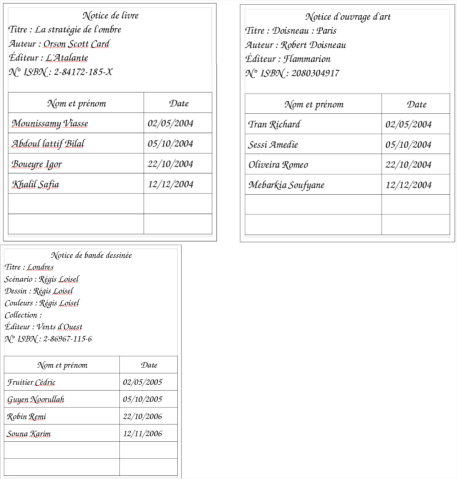
**Mini projet Bibliothèque**

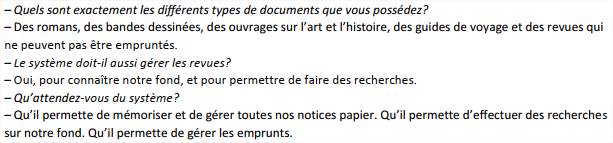
**Objectif :**

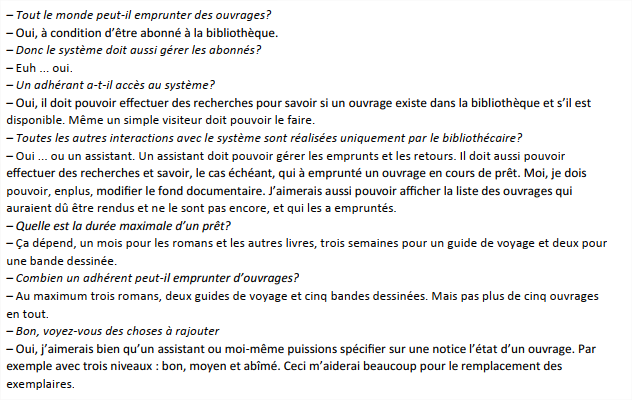
Nous devons créer une application pour pouvoir gérer les adhérents, ouvrages et emprunts dans une bibliothèque.

1. **Etude du système**



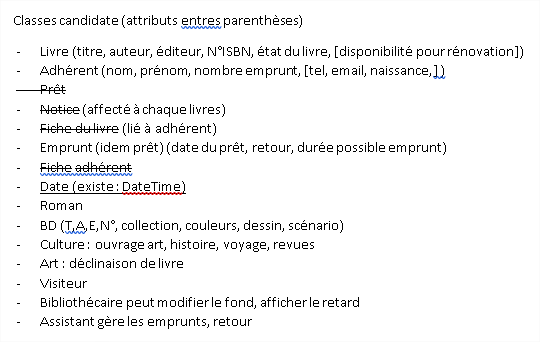






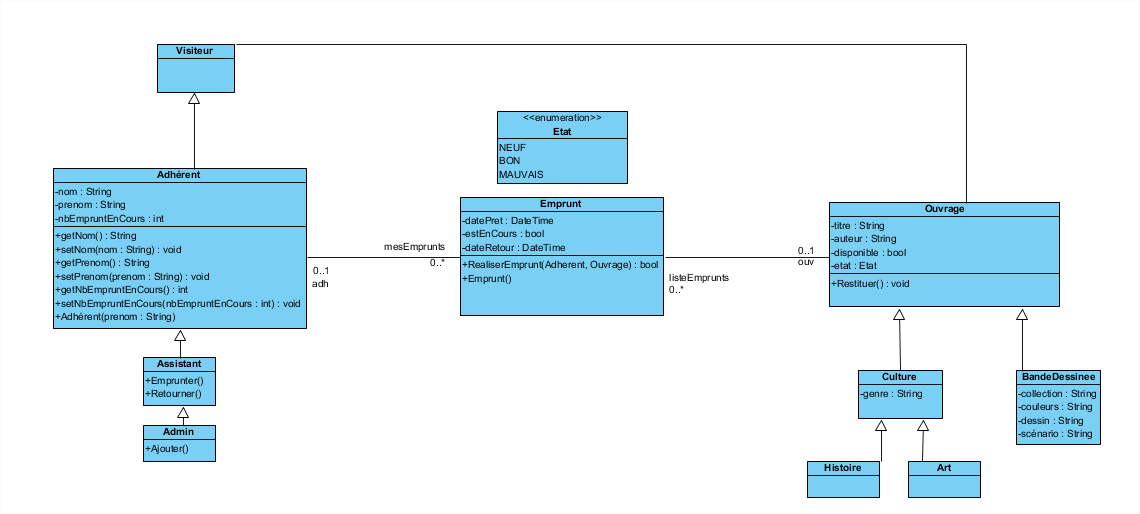
Notre réflexion pour pouvoir mettre en place un diagramme en classe est :

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.



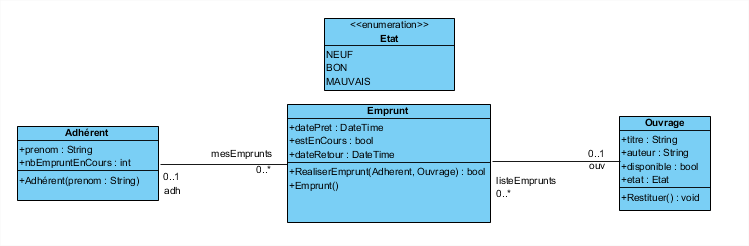
1. **Diagramme de classe**

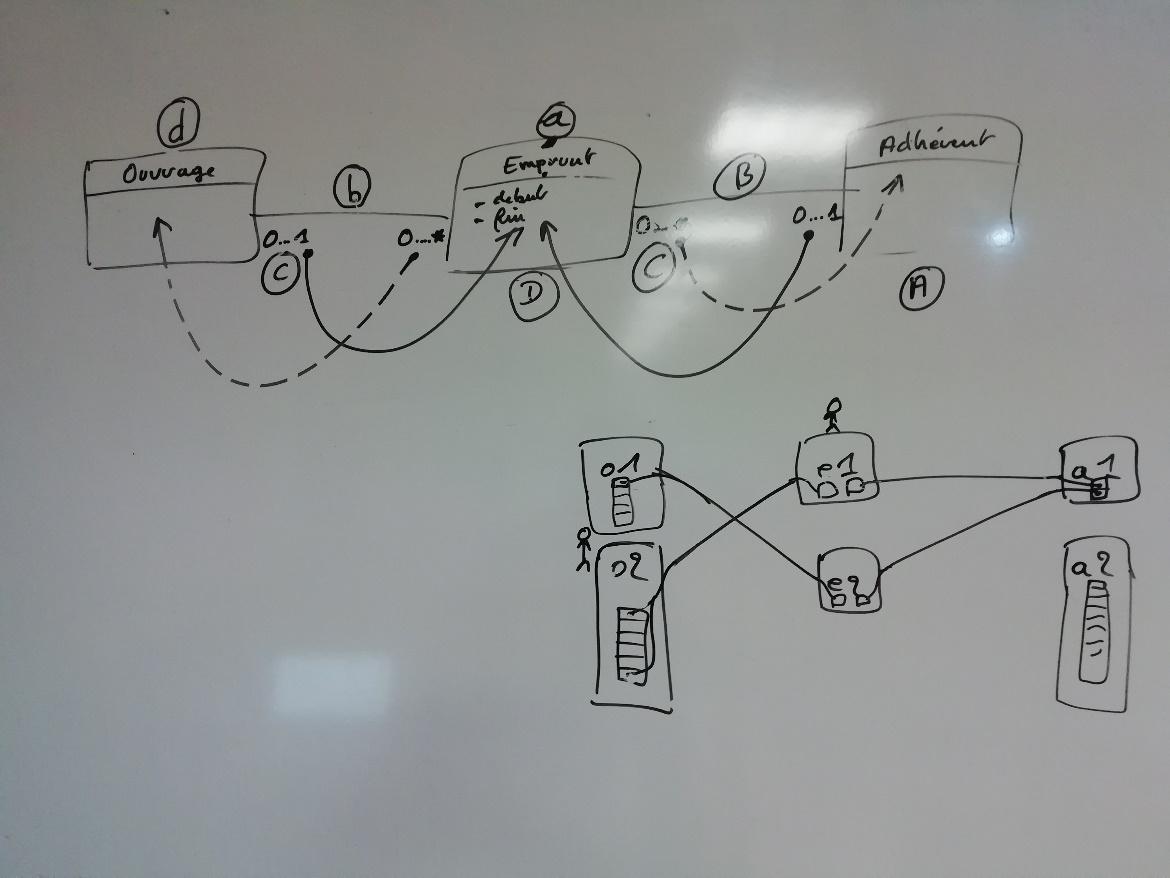
Voici donc notre diagramme de classe suite à l’analyse.



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)

1. **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();

}

}