IUT2 Université Grenoble Alpes
Département INFO

S1107 – Programmation Orientée Objet

Projet : Système d'information pour la gestion d'une bibliothèque

DUT AS Informatique
2021-2022

ALBANESE Rémy
BONACORSI Camille
LAURAC Alexandre

SOMMAIRE

I/ INTRODUCTION

Notre équipe a été sollicitée par la société CPO pour analyser et concevoir un système de gestion d'une bibliothèque municipale. Plusieurs cas d'utilisation étaient à programmer : la création d'un nouveau lecteur, la création d'un nouvel ouvrage, la création d'un ou plusieurs exemplaires d'un ouvrage, la consultation d'un lecteur, la consultation d'un ouvrage, la consultation des exemplaires d'un ouvrage, l'emprunt d'un exemplaire, le retour d'un exemplaire et la relance d'un lecteur lors d'un retard.

Afin de réaliser ce projet, nous avons décidé d'utiliser la programmation orientée objet couplée au processus unifié, notamment avec le langage UML (Unified Modeling Language). En effet, ce fonctionnement permet de se focaliser sur les interactions et relations entre différents objets, ici physiques comme un lecteur ou un ouvrage, en les faisant passer d'objets réels à virtuels.

Pour ce faire nous disposions de différents outils. Premièrement nous nous sommes servis du logiciel Visual Paradigm et de sa fonctionnalité « team » pour partager, commenter nos modifications. Il nous a permis de réaliser toute la partie de modélisation de ce projet. Notamment avec les diagrammes de cas d'utilisation, de classe, de séquence de haut niveau, de séquence détaillés et d'état.

Ensuite, pour l'implémentation du code, nous avons utilisé le langage de programmation orienté objet Java et l'environnement de développement intégré NetBeans avec le logiciel de gestion de versions décentralisé Git. Ce dernier nous a permis de partager et gérer les différentes modifications du code effectuées. En nous appuyant sur la démarche de développement UML, nous avons réalisé notre projet sur deux itérations, la première étant centrée sur la gestion de la bibliothèque, et la seconde étant centrée sur la gestion des emprunts. Pour chaque itération, nous suivions le plan suivant : expressions des besoins et analyse, puis conception avec les diagrammes de séquence détaillés et enfin implantation en codant les algorithmes de méthodes en Java.

CONVENTIONS DU GROUPE PROJET

Les conventions de nommage que nous avons utilisées dans ce projet ont été établies à l'aide de la partie 7 "Démarche de développement" du cours de COO, p.7 et 8 intitulées "Dans un projet : fixer des notations communes" et constituent une adaptation de ces notations. Les conventions sont les suivantes :

- les noms d'attributs de classe (types de base ou objets) :

ils ont été choisis pour désigner de façon explicite leur rôle et sont la plupart du temps des noms complets.

Ex : nomLecteur, nomEditeur, numExemplaire pour des types "de base" ; lecteur, ouvrage, emprunt pour des objets.

Ils sont cohérents entre le diagramme de classes et le code Java.

- les noms de variables dans le corps des méthodes (notamment celles des cas d'utilisation) :

* types de base : ils reprennent le nom de l'attribut de classe correspondant

ex : dans les primitives d'entrée-sortie ('String numISBN = ihm.saisirNumOuvrage(listISBN)

* les objets : ils reçoivent dans les différentes méthodes de classe des noms simples, constitués d'une seule lettre, voire deux :

ex : Ouvrage o, Lecteur l, Exemplaire ex, Emprunt em

Ils sont cohérents entre les diagrammes de séquences et le code Java

* les collections d'objets : ayant des noms explicites comme attributs de classe, elles prennent dans le corps des méthodes un nom du type 'collectNom', avec Nom le nom de la classe des objets qu'elles contiennent

- pour la mise à jour des rôles d'association, les conventions suivantes sont utilisées :

* ajouterNom : méthode publique pour ajouter un élément à l'objet destination d'un rôle multivalué

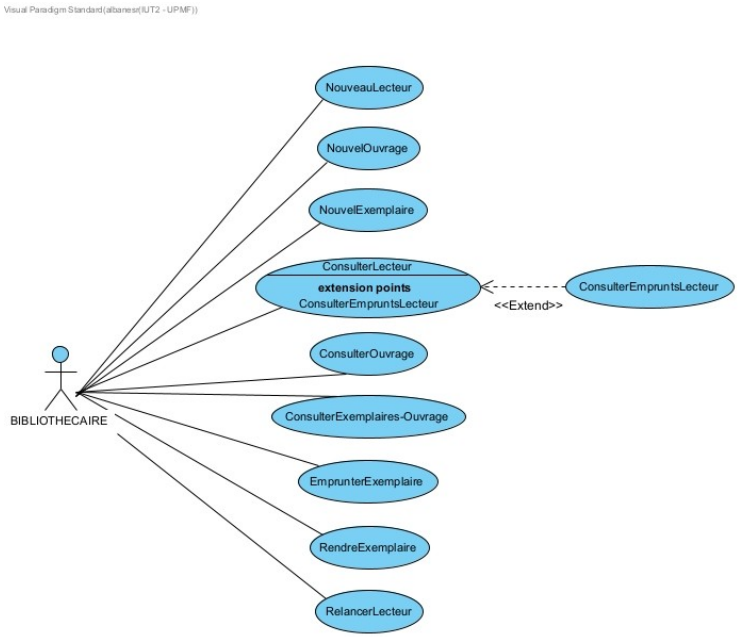
* setNom : méthode publique pour affecter l'objet destination d'un rôle monovalué

* lierNom : setters privés pour ajouter un élément à l'objet destination d'un rôle mono ou multivalué

* unNom : méthode publique pour accéder à un objet via un rôle d'association

I/ EXPRESSION DES BESOINS ET ANALYSE

Diagramme des cas d'utilisation :



Nouveau Lecteur

Paramètres saisis :
nomLecteur : String, prénomLecteur : String, dateNaissance : Date, mail : String
Précondition :
La date de naissance doit être antérieure ou égale à la date du jour.
Résumé :
Ce cas d'utilisation permet d'ajouter un nouveau lecteur dans le système en insérant son nom, prénom, date de naissance, mail. Un numéro de lecteur lui est attribué séquentiellement.

Nouvel Ouvrage

Paramètres saisis :
titre : String, nomEditeur : String, dateParution : Date, nomsAuteurs : Set(String) / ou ArrayList<String>, numISBN : Integer, publicVisé : Public (enfant, ado, adulte)
Précondition :
Le numéro ISBN ne doit pas déjà exister. La date de parution doit être antérieure à la date du jour.
Résumé :
Ce cas d'utilisation permet d'ajouter un nouvel ouvrage dans le système en insérant son titre, son éditeur, ses auteurs, sa date de parution, son numéro ISBN et le type de public auquel il s'adresse.

Nouvel Exemplaire

Paramètres saisis : numISBN : Integer (le même que l'ouvrage), dateRecep : Date, nbExemplairesEntrés : Integer, nbNonEmpruntables : Integer
Précondition :
L'exemplaire correspond à un ouvrage géré par le système. La date de réception de l'exemplaire est postérieure à la date de parution de l'ouvrage correspondant et est antérieure ou égale à la date du jour.
Résumé :
Ce cas d'utilisation permet d'ajouter un ou plusieurs exemplaires d'un ouvrage déjà possédé par la bibliothèque en insérant son numéro ISBN, sa date de réception, le nombre d'exemplaires, sa possibilité d'être emprunté. Le système lui attribue un numéro d'exemplaire séquentiellement.

Emprunter Exemplaire

Paramètres saisis : numISBN : Integer, numExemplaire : Integer, numLecteur : Integer
Précondition : Le lecteur doit exister et avoir moins de 5 emprunts en cours. L'exemplaire doit exister, être empruntable et disponible. L'âge du lecteur correspond au public visé par l'ouvrage.
Résumé : Ce cas d'utilisation permet l'emprunt d'un exemplaire par un lecteur en insérant son numéro de lecteur, le numéro ISBN de l'ouvrage et le numéro d'exemplaire.

ConsulterLecteur étendu par ConsulterEmpruntsLecteur

Paramètre saisis :
numLecteur : Integer
Précondition :
Le lecteur doit exister pour être consulté.
Résumé :
Ce cas d'utilisation permet la consultation du numéro, nom, prénom, age et mail d'un lecteur donné, ainsi que ses emprunts en cours. Pour chaque emprunt les informations suivantes sont affichées : titre, ISBN, numéro d'exemplaire de l'ouvrage, ainsi que les dates d'emprunt et de retour.

ConsulterOuvrage

Paramètres saisis :
numISBN : Integer
Précondition :
L'ouvrage doit exister pour être consulté
Résumé :
Ce cas d'utilisation permet la consultation du titre, numéro ISBN, auteurs, nom d'éditeur et date d'édition d'un ouvrage donné.

ConsulterExemplairesOuvrage

Paramètres saisis :
numISBN : Integer
Précondition :
L'ouvrage doit exister.
Résumé :
Ce cas d'utilisation permet d'afficher, pour l'ouvrage saisi, son titre et son numéro ISBN. Pour chaque exemplaire existant son numéro est affiché et, dans le cas où il est emprunté, le numéro, nom, prénom du lecteur, ainsi que les dates d'emprunt et de retour sont également affichés.

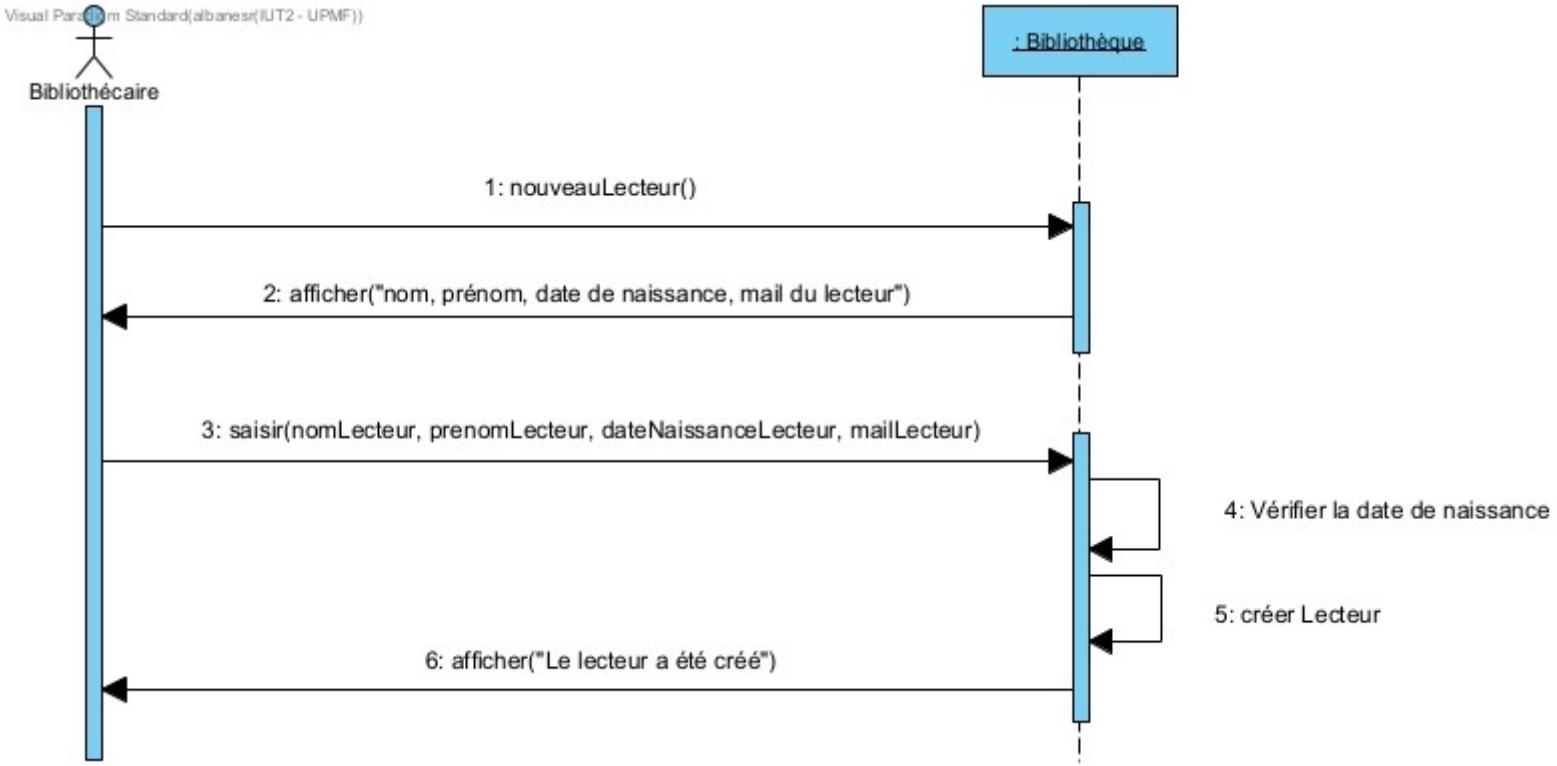
Rendre Exemplaire

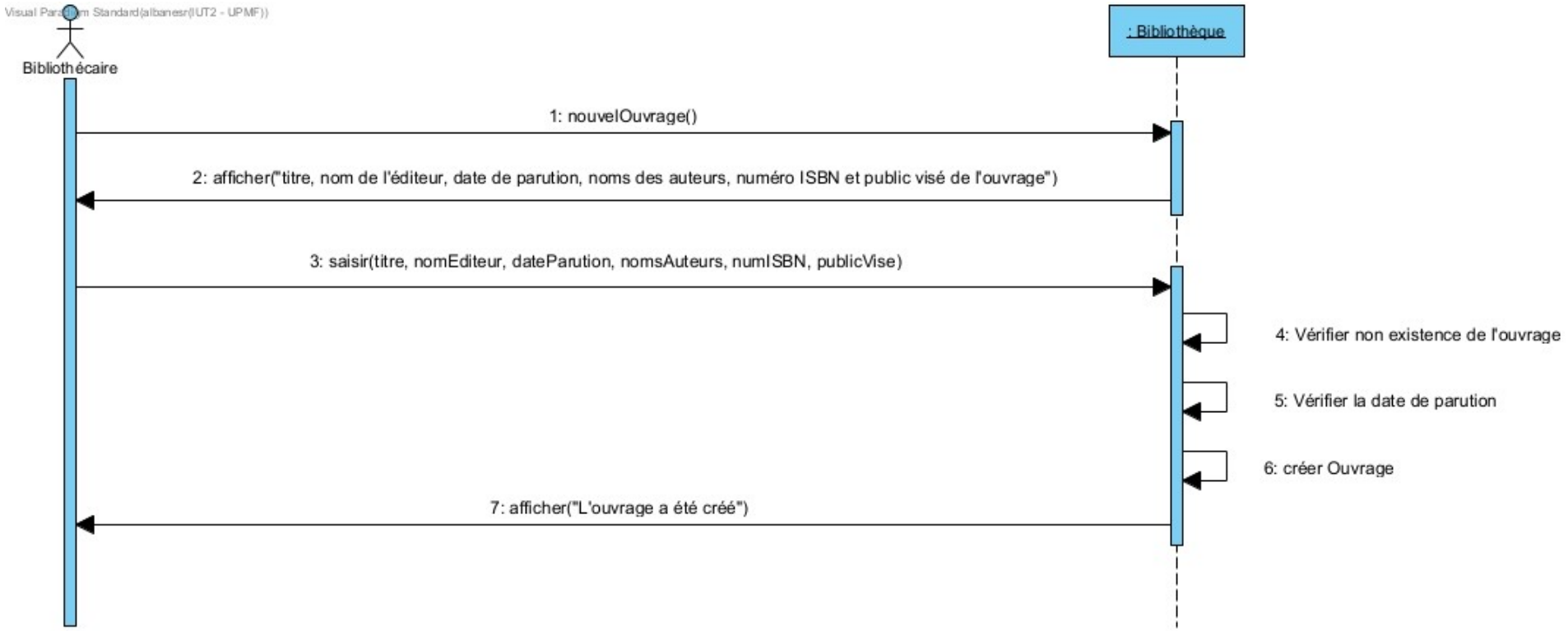
Paramètres saisis : numISBN : Integer, numExemplaire : Integer, numLecteur : Integer
Précondition : L'ouvrage doit exister. L'exemplaire doit être emprunté et exister, et le numLecteur doit exister.
Résumé : Ce cas d'utilisation permet le retour de l'exemplaire d'un ouvrage qui était emprunté en insérant le numéro lecteur, le numéro ISBN de l'ouvrage et son numéro d'exemplaire.

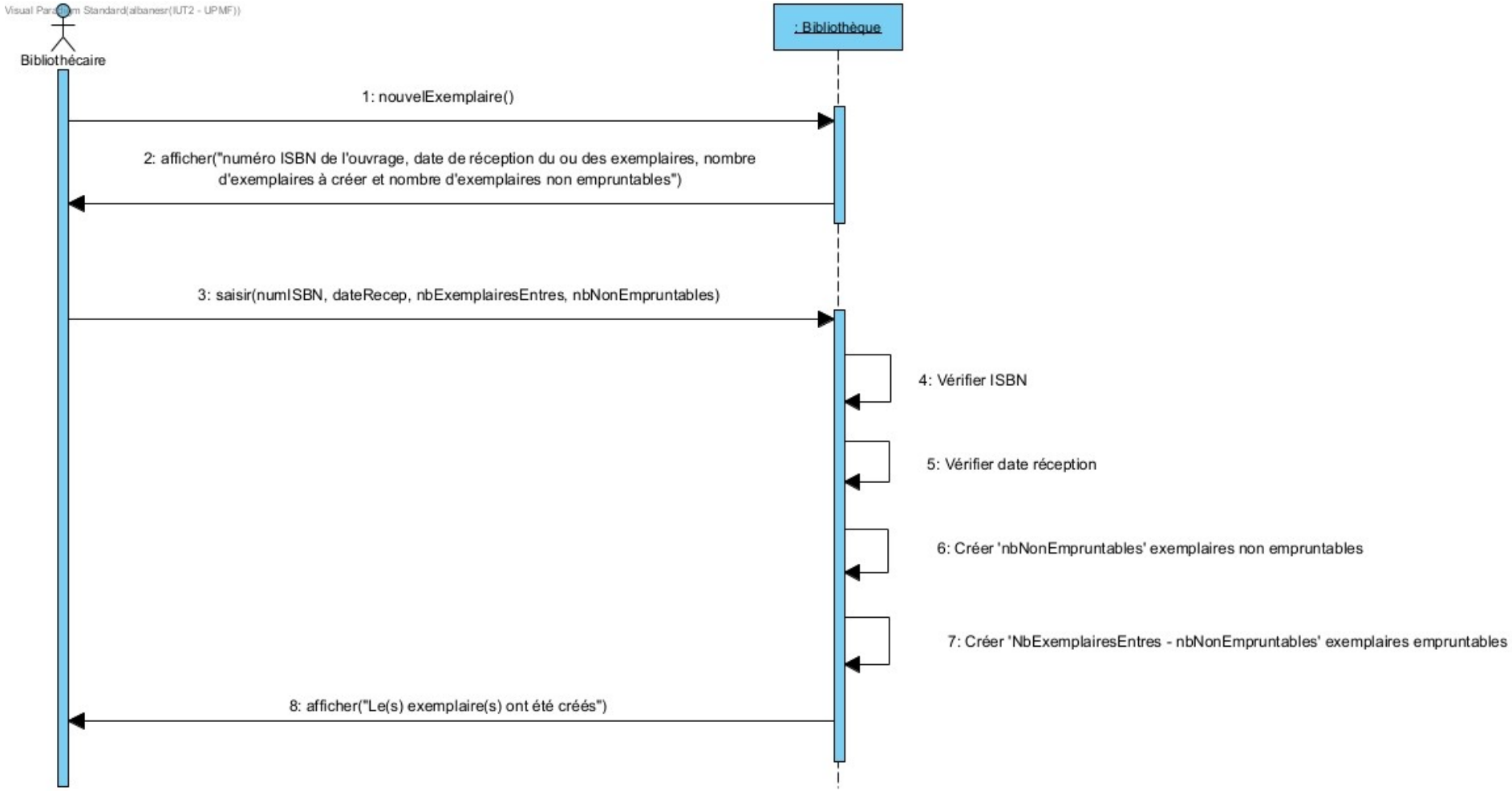
Relancer Lecteur

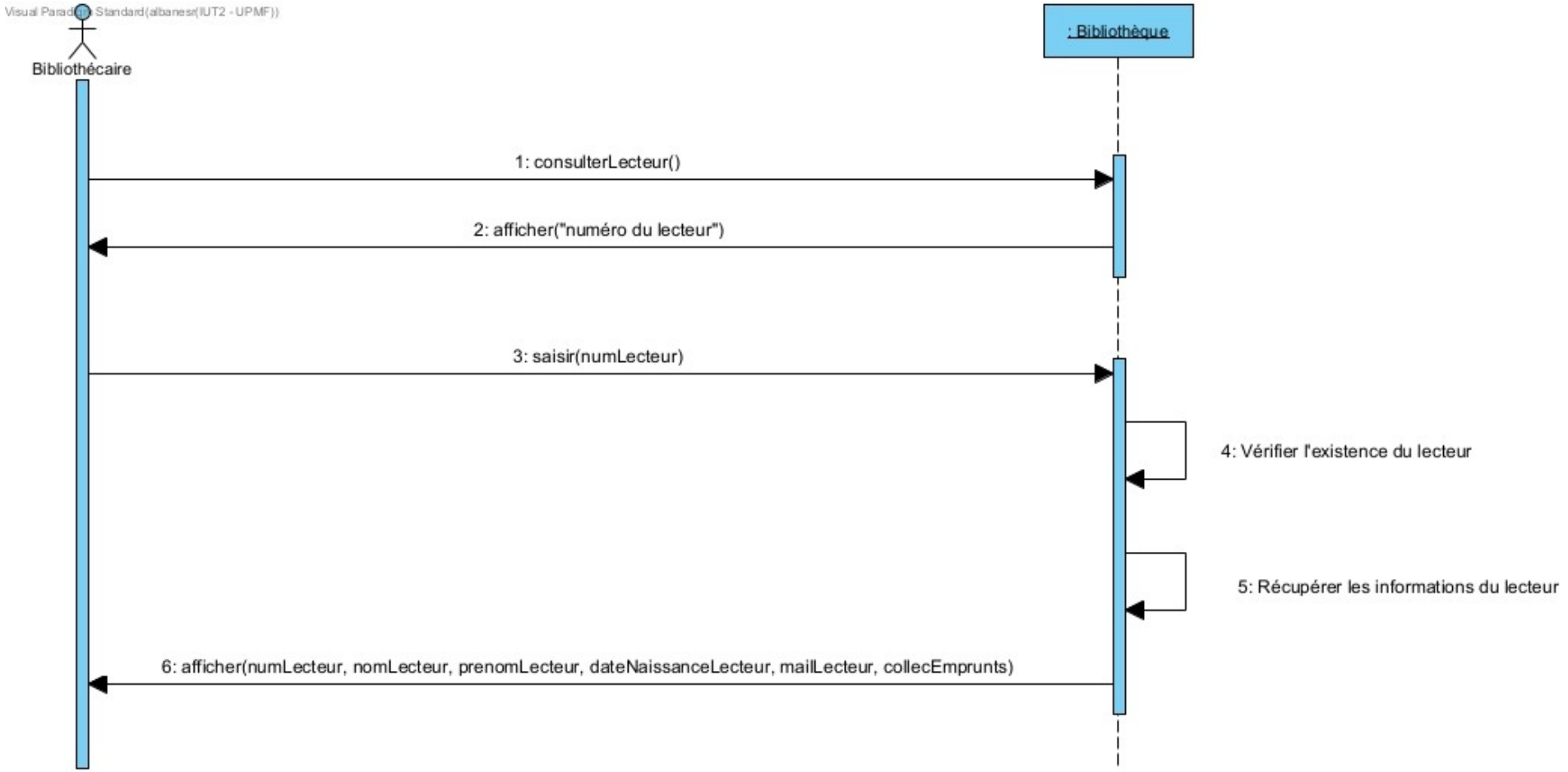
Paramètres saisis :
/
Précondition :
/
Résumé :
Ce cas d'utilisation affiche, pour chaque exemplaire non retourné avec un retard de 15 jours,: numéro, nom, prénom, age, mail du lecteur qui l'a emprunté et titre, ISBN, numéro d'exemplaire ,dates d'emprunt et de retour de l'exemplaire concerné.

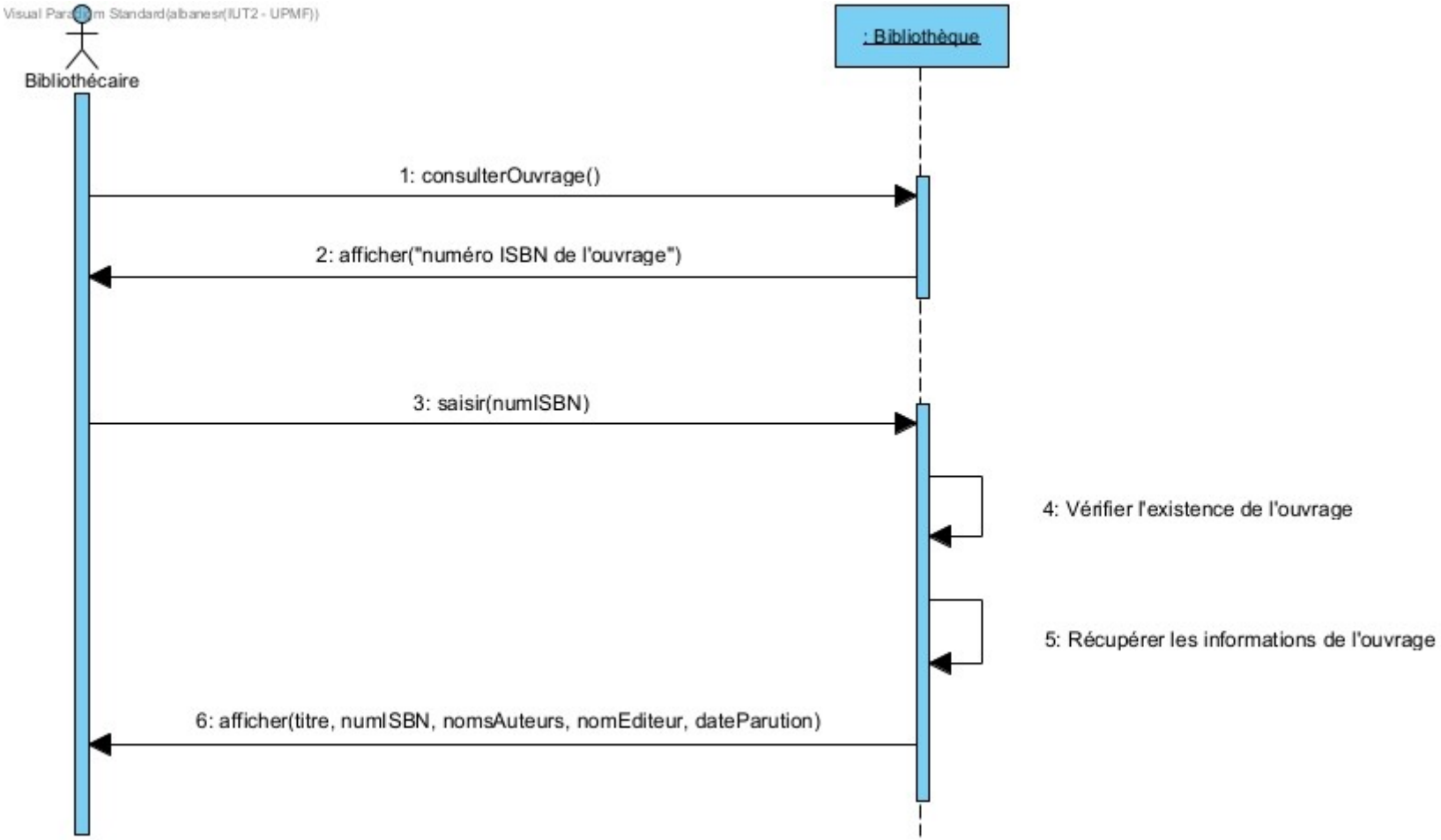
Diagrammes de séquence de haut niveau

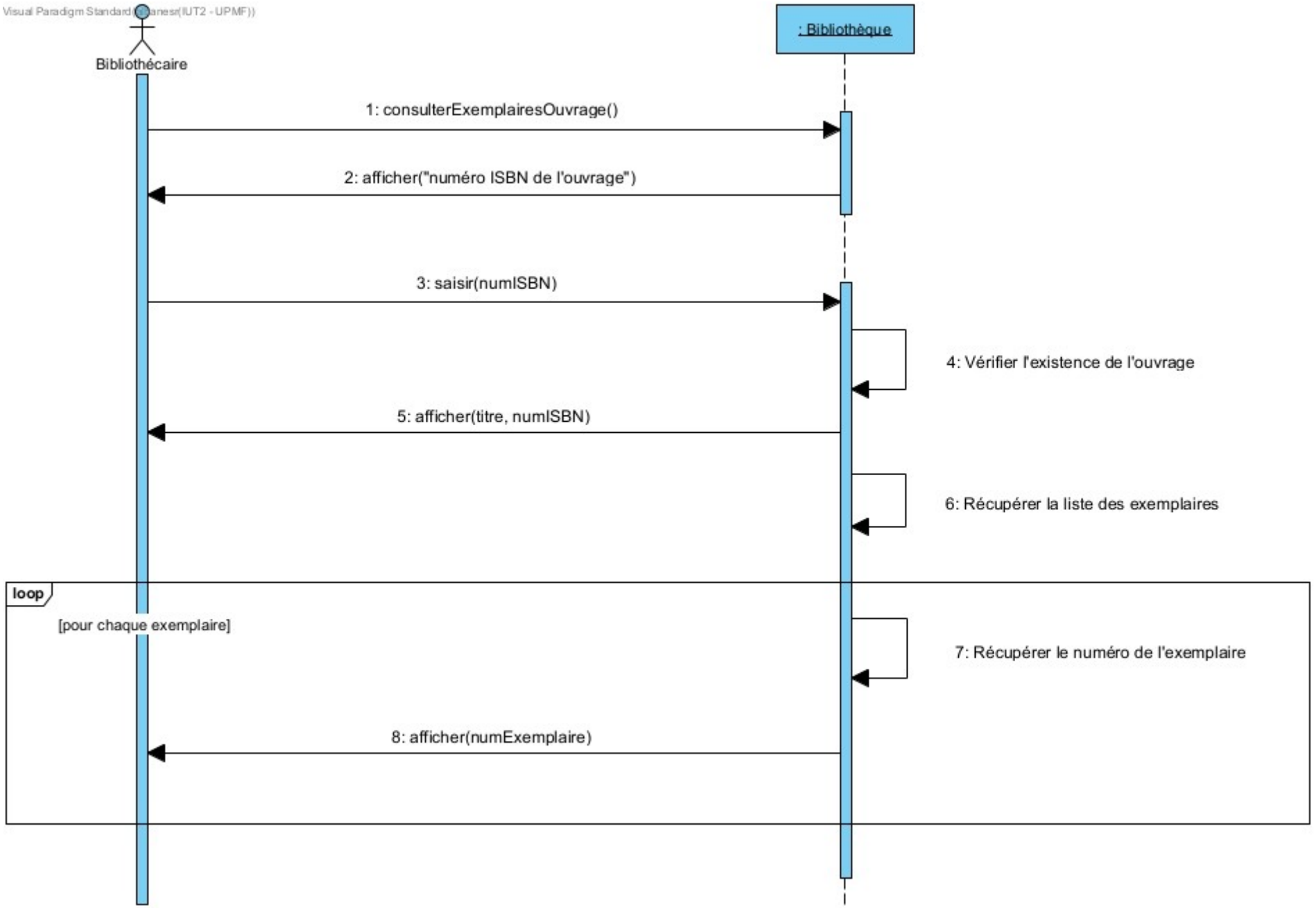


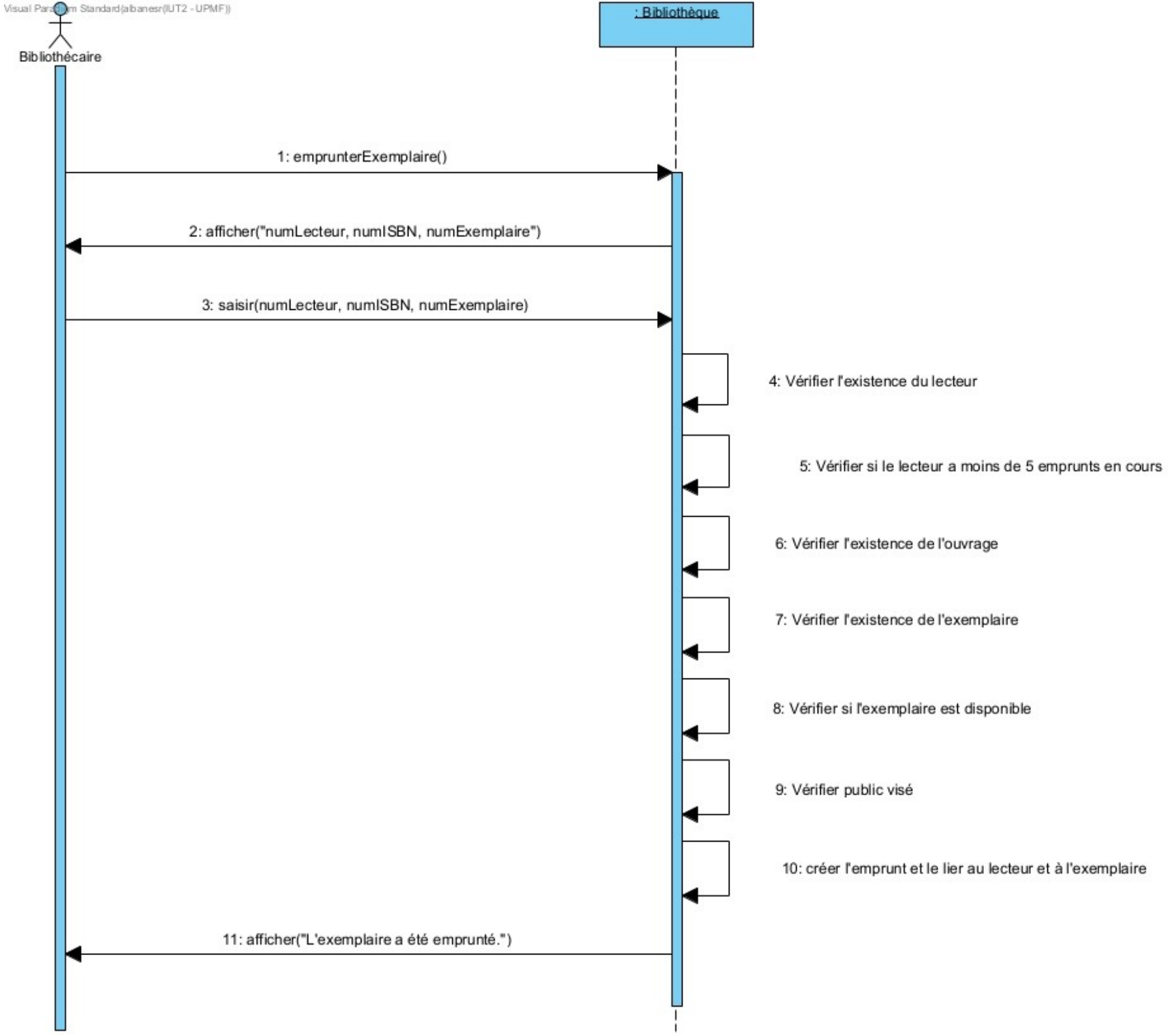

Visual Paradigm Standard(albanesr(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: nouvelOuvrage()
2: afficher("titre, nom de l'éditeur, date de parution, noms des auteurs, numéro ISBN et public visé de l'ouvrage")
3: saisir(titre, nomEditeur, dateParution, nomsAuteurs, numISBN, publicVise)
4: Vérifier non existence de l'ouvrage
5: Vérifier la date de parution
6: créer Ouvrage
7: afficher("L'ouvrage a été créé")

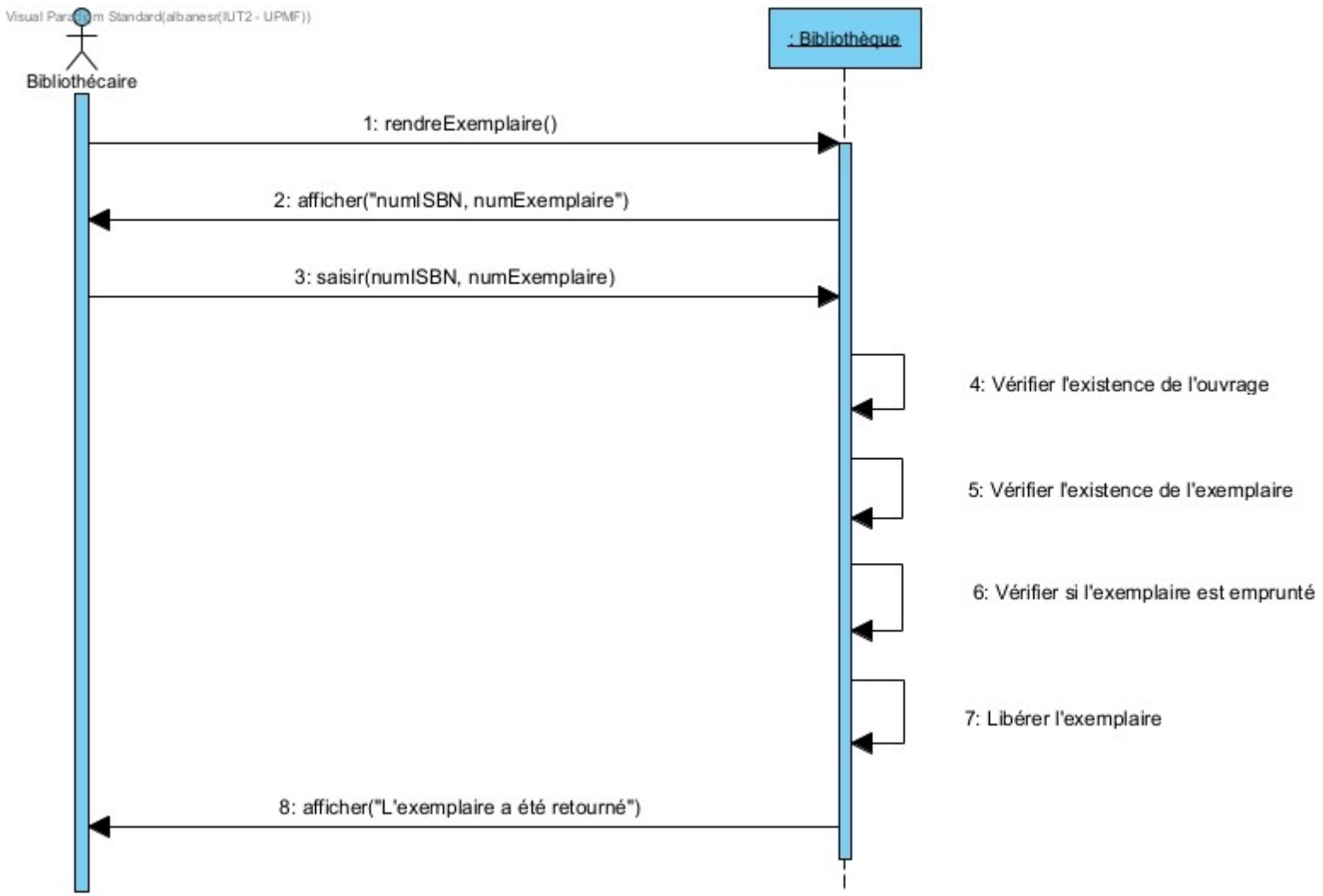

Visual Paradigm Standard(albanesr(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: nouvelExemplaire()
2: afficher("numéro ISBN de l'ouvrage, date de réception du ou des exemplaires, nombre d'exemplaires à créer et nombre d'exemplaires non empruntables")
3: saisir(numISBN, dateRecep, nbExemplairesEntres, nbNonEmpruntables)
4: Vérifier ISBN
5: Vérifier date réception
6: Créer 'nbNonEmpruntables' exemplaires non empruntables
7: Créer 'NbExemplairesEntres - nbNonEmpruntables' exemplaires empruntables
8: afficher("Le(s) exemplaire(s) ont été créés")


Visual Paradigm Standard(albanest(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: consulterLecteur()
2: afficher("numéro du lecteur")
3: saisir(numLecteur)
4: Vérifier l'existence du lecteur
5: Récupérer les informations du lecteur
6: afficher(numLecteur, nomLecteur, prenomLecteur, dateNaissanceLecteur, mailLecteur, collecEmprunts)


Visual Paradigm Standard(albanest(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: consulterOuvrage()
2: afficher("numéro ISBN de l'ouvrage")
3: saisir(numISBN)
4: Vérifier l'existence de l'ouvrage
5: Récupérer les informations de l'ouvrage
6: afficher(titre, numISBN, nomsAuteurs, nomEditeur, dateParution)


Visual Paradigm Standard(Banest(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: consulterExemplairesOuvrage()
2: afficher("numéro ISBN de l'ouvrage")
3: saisir(numISBN)
4: Vérifier l'existence de l'ouvrage
5: afficher(titre, numISBN)
6: Récupérer la liste des exemplaires
loop
[pour chaque exemplaire]
7: Récupérer le numéro de l'exemplaire
8: afficher(numExemplaire)


Visual Paradigm Standard(albanesr(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: emprunterExemplaire()
2: afficher("numLecteur, numISBN, numExemplaire")
3: saisir(numLecteur, numISBN, numExemplaire)
4: Vérifier l'existence du lecteur
5: Vérifier si le lecteur a moins de 5 emprunts en cours
6: Vérifier l'existence de l'ouvrage
7: Vérifier l'existence de l'exemplaire
8: Vérifier si l'exemplaire est disponible
9: Vérifier public visé
10: créer l'emprunt et le lier au lecteur et à l'exemplaire
11: afficher("L'exemplaire a été emprunté.")


Visual Paradigm Standard(albanesr(IUT2 - UPMF))
Bibliothécaire
: Bibliothèque
1: rendreExemplaire()
2: afficher("numISBN, numExemplaire")
3: saisir(numISBN, numExemplaire)
4: Vérifier l'existence de l'ouvrage
5: Vérifier l'existence de l'exemplaire
6: Vérifier si l'exemplaire est emprunté
7: Libérer l'exemplaire
8: afficher("L'exemplaire a été retourné")

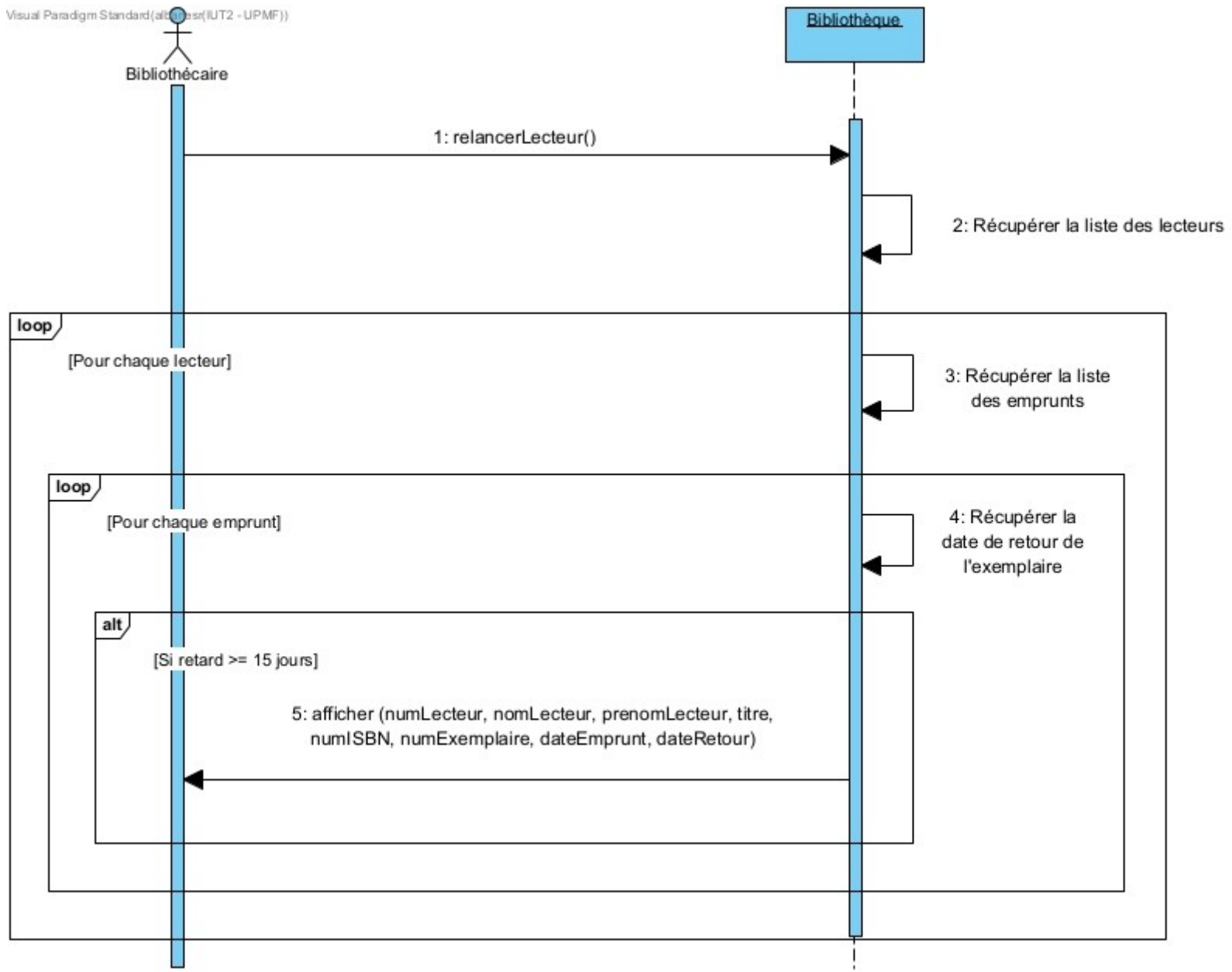
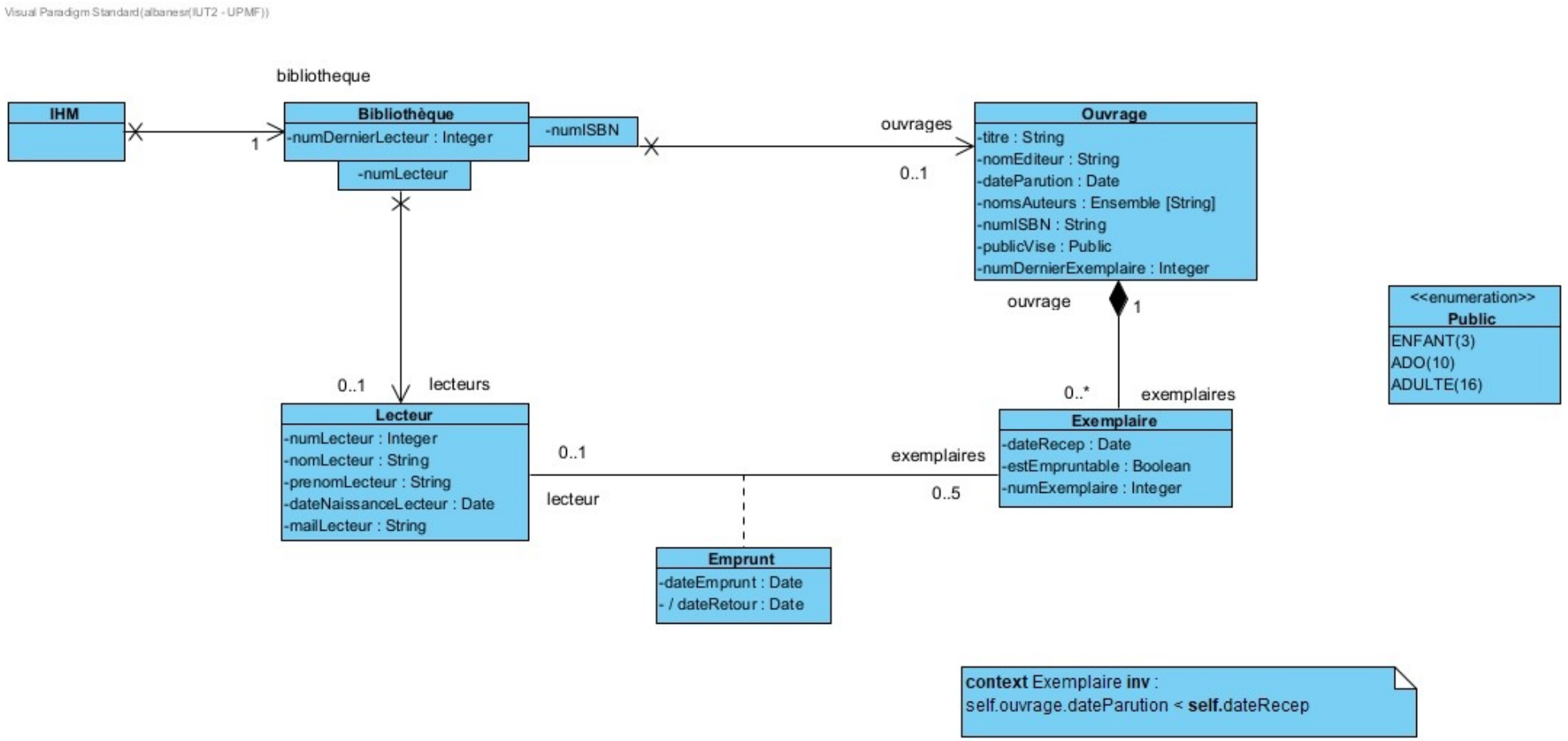
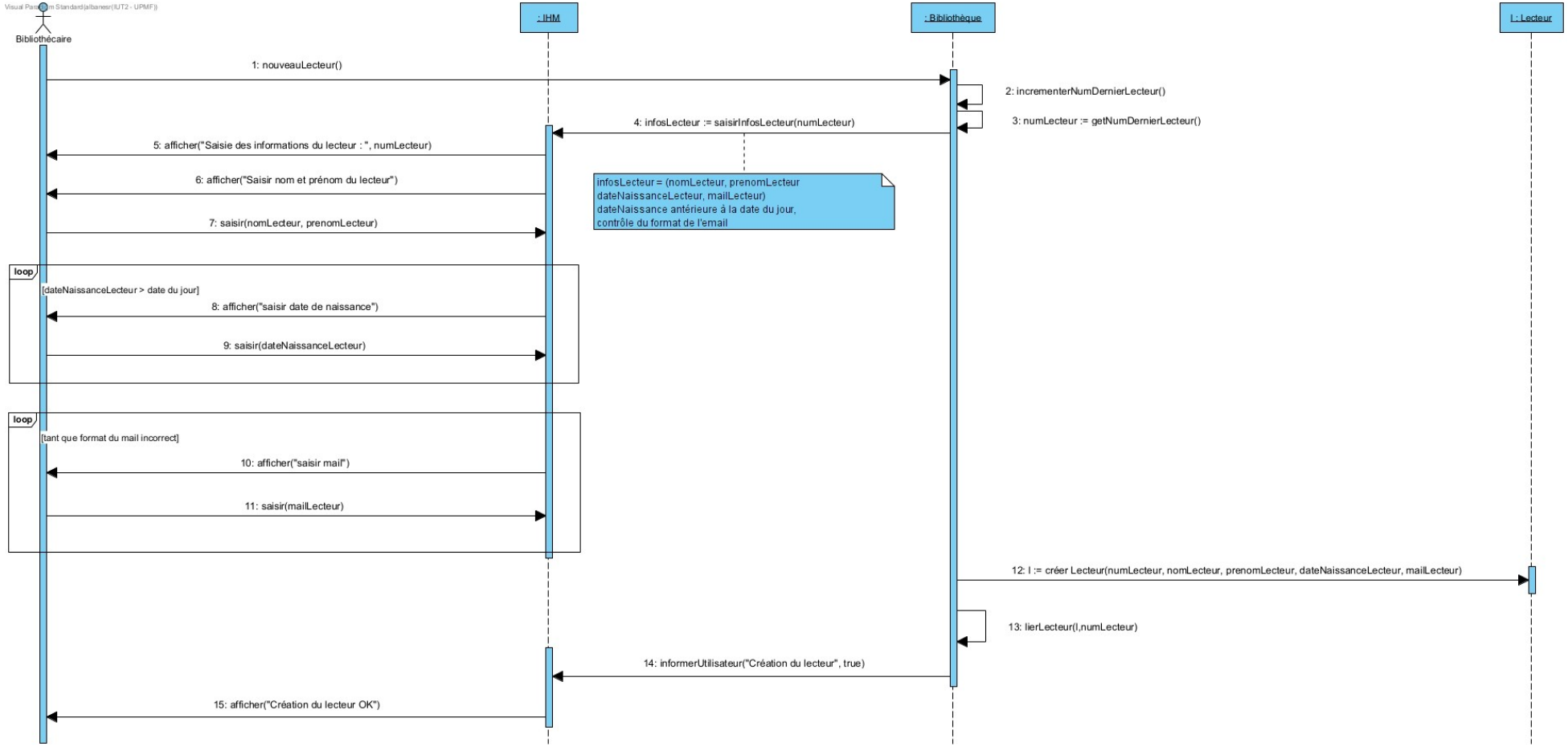

Visual Paradigm Standard(albertes(IUT2 - UPMF))
Bibliothécaire
Bibliothèque
1: relancerLecteur()
2: Récupérer la liste des lecteurs
loop
[Pour chaque lecteur]
3: Récupérer la liste des emprunts
loop
[Pour chaque emprunt]
4: Récupérer la date de retour de l'exemplaire
alt
[Si retard >= 15 jours]
5: afficher (numLecteur, nomLecteur, prenomLecteur, titre, numISBN, numExemplaire, dateEmprunt, dateRetour)

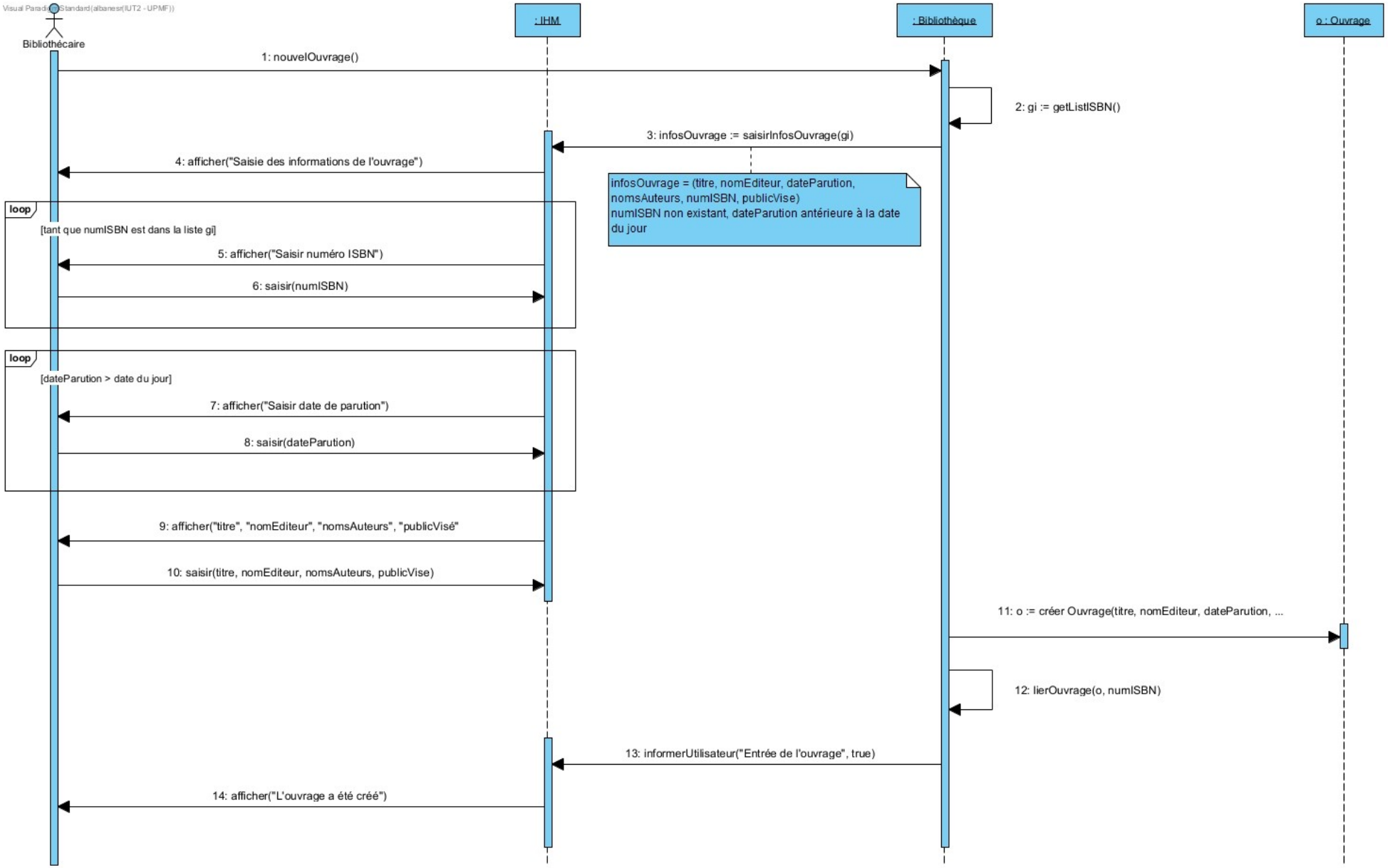
Diagramme de classe de niveau analyse

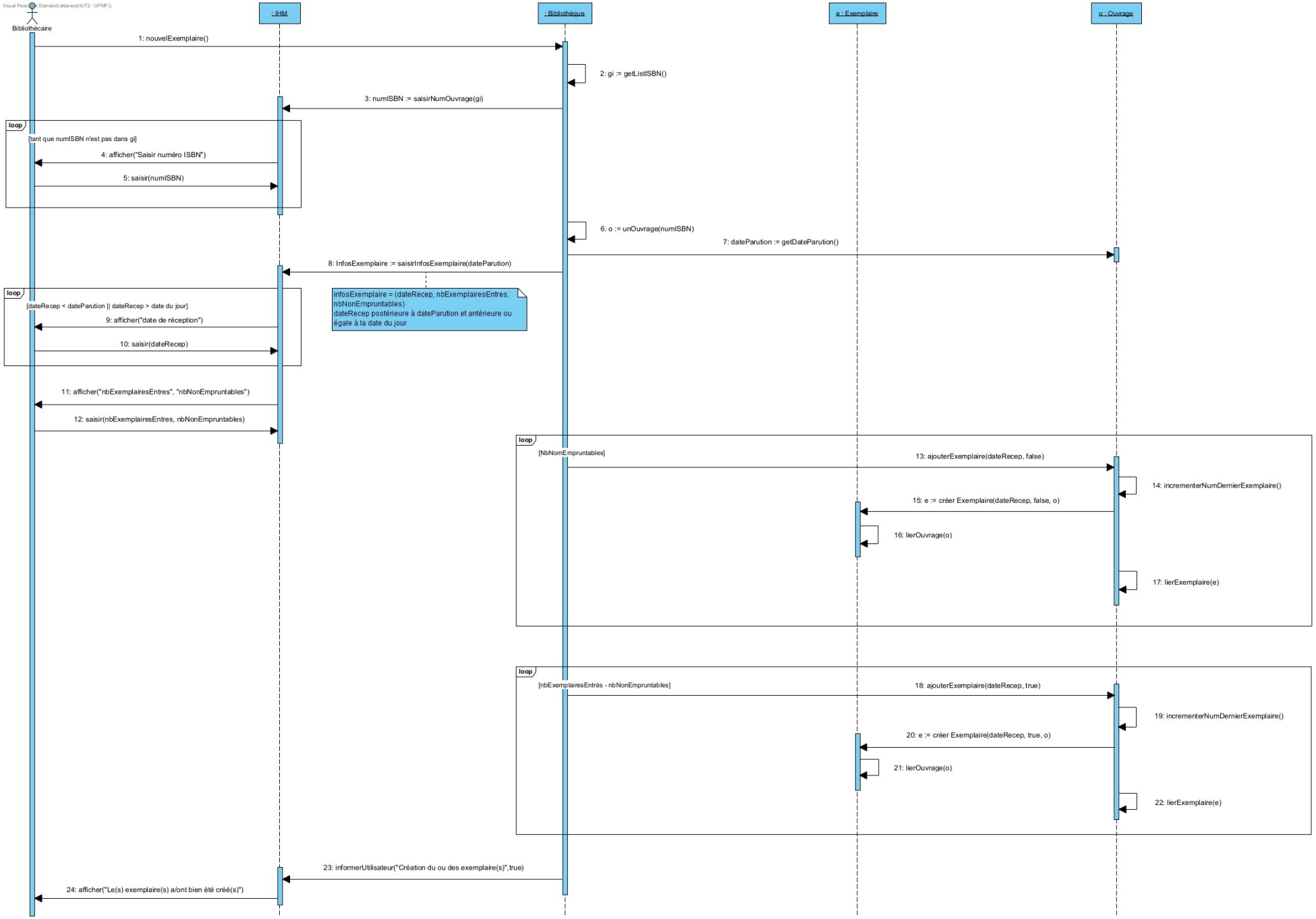


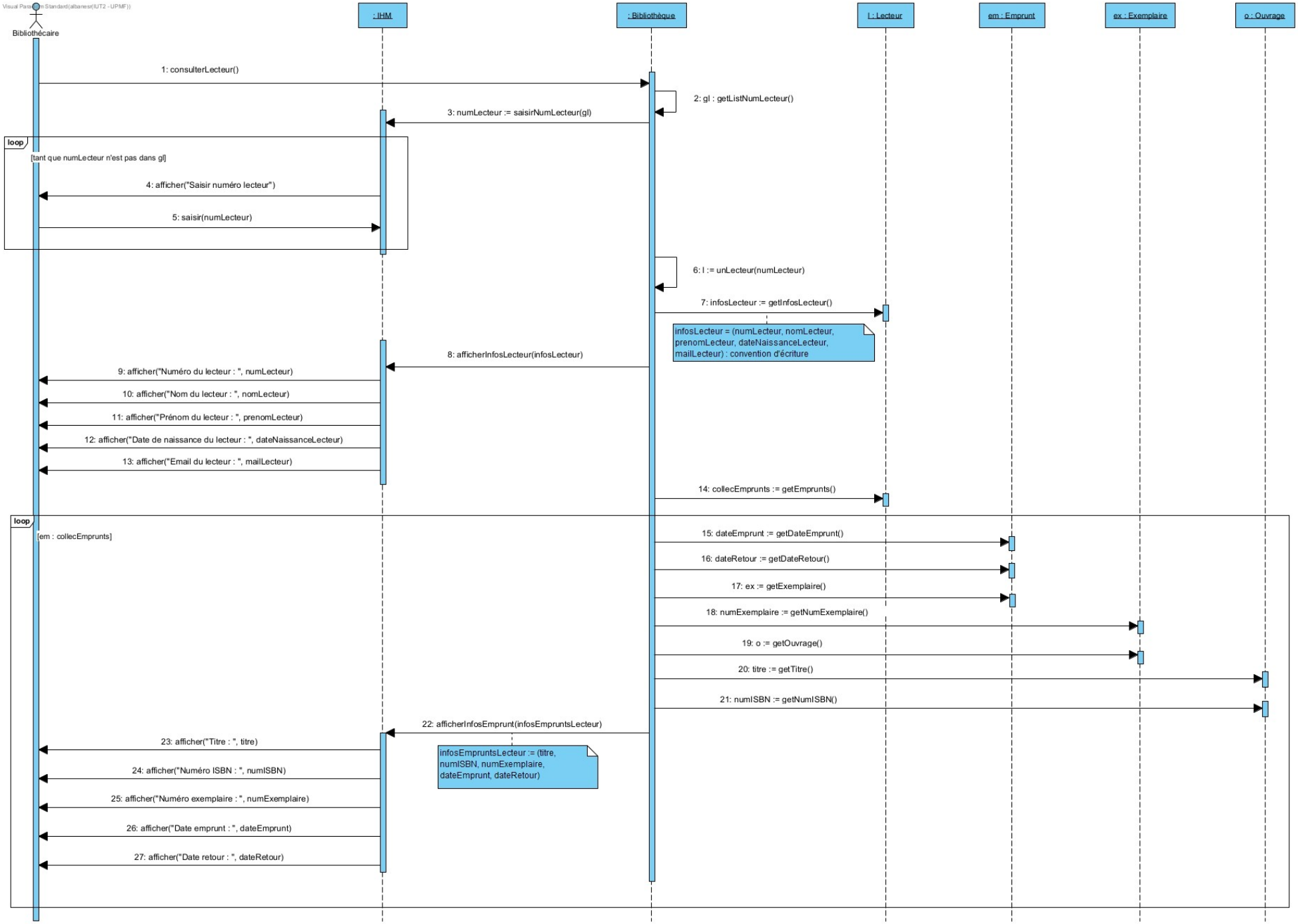
III/ Conception

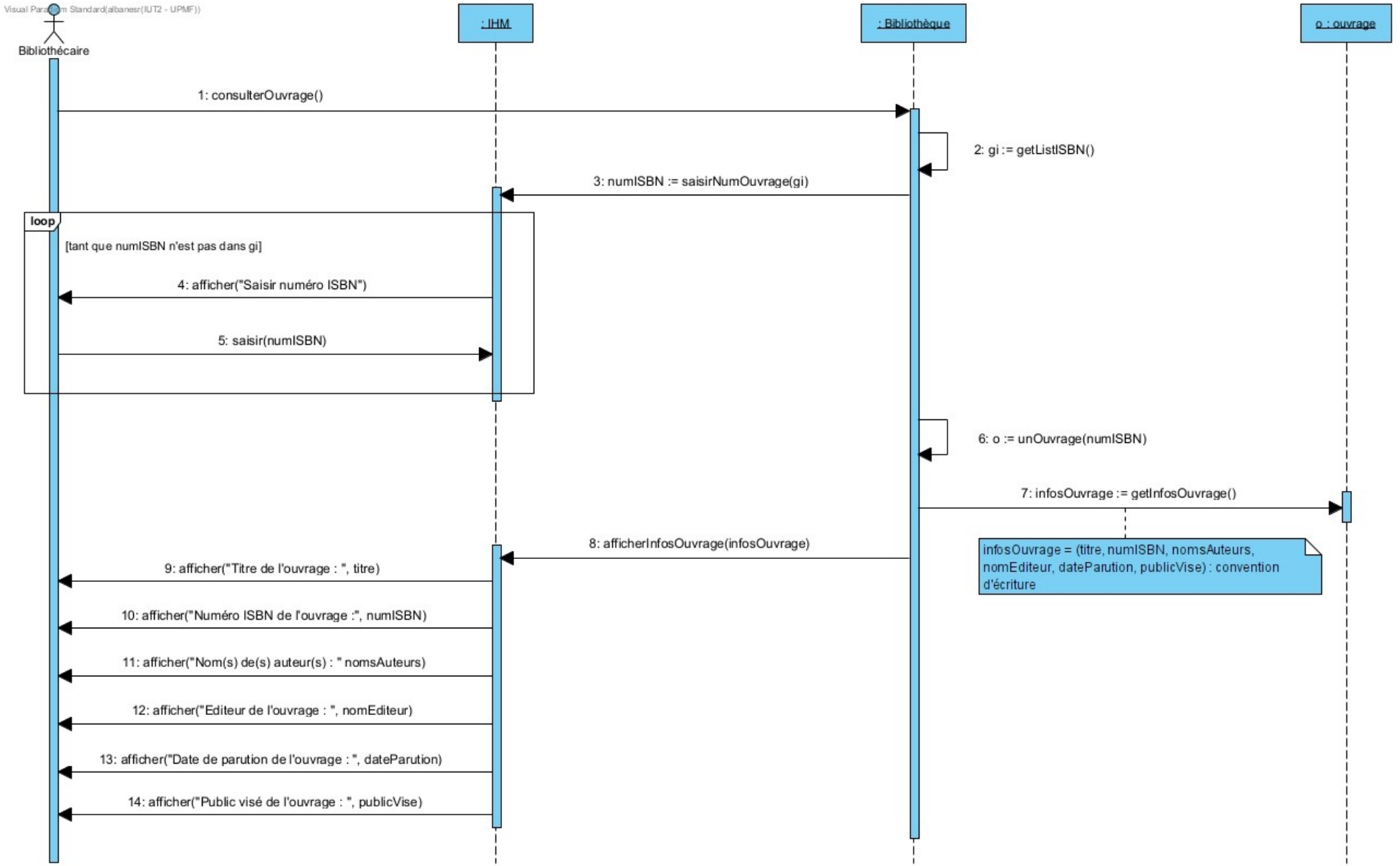
Diagrammes de séquence détaillés

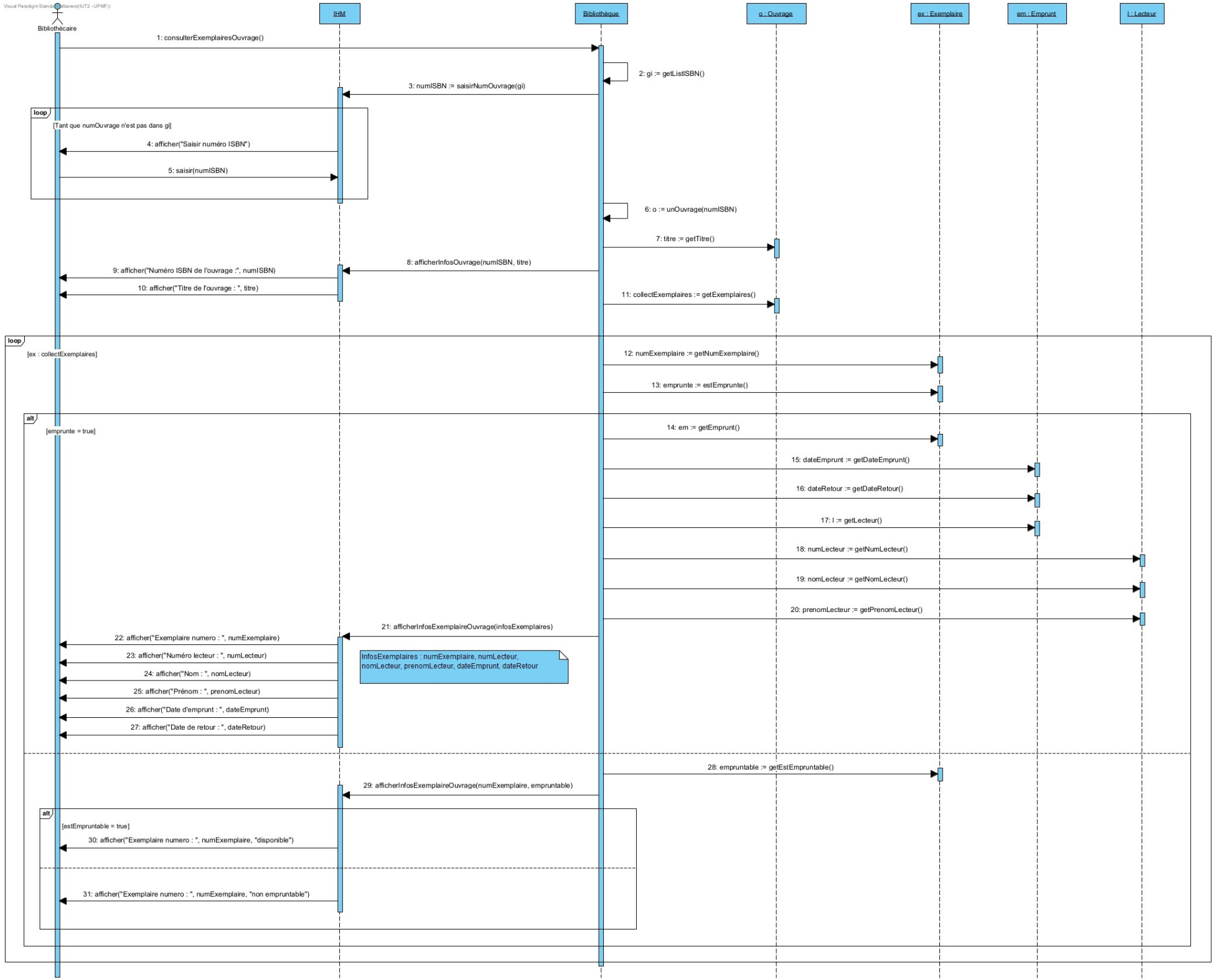


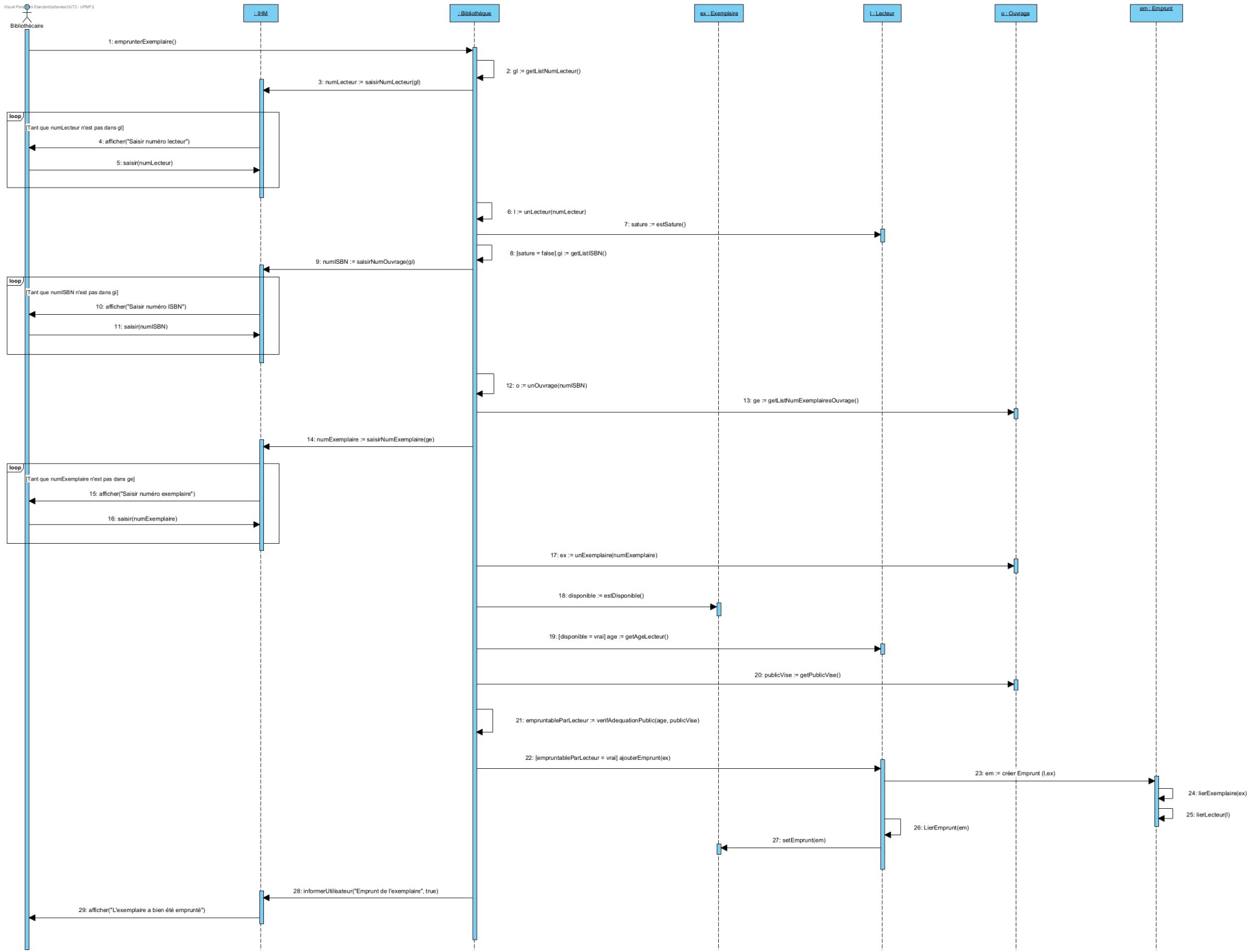

Visual Paradigm Standard(albanesr(IUT2 - UPMF))
Bibliothécaire
: IHM
: Bibliothèque
o : Ouvrage
1: nouvelOuvrage()
2: gi := getListISBN()
3: infosOuvrage := saisirInfosOuvrage(gi)
4: afficher("Saisie des informations de l'ouvrage")
infosOuvrage = (titre, nomEditeur, dateParution, nomsAuteurs, numISBN, publicVise)
numISBN non existant, dateParution antérieure à la date du jour
loop
[tant que numISBN est dans la liste gi]
5: afficher("Saisir numéro ISBN")
6: saisir(numISBN)
loop
[dateParution > date du jour]
7: afficher("Saisir date de parution")
8: saisir(dateParution)
9: afficher("titre", "nomEditeur", "nomsAuteurs", "publicVisé"
10: saisir(titre, nomEditeur, nomsAuteurs, publicVise)
11: o := créer Ouvrage(titre, nomEditeur, dateParution, ...
12: lierOuvrage(o, numISBN)
13: informerUtilisateur("Entrée de l'ouvrage", true)
14: afficher("L'ouvrage a été créé")

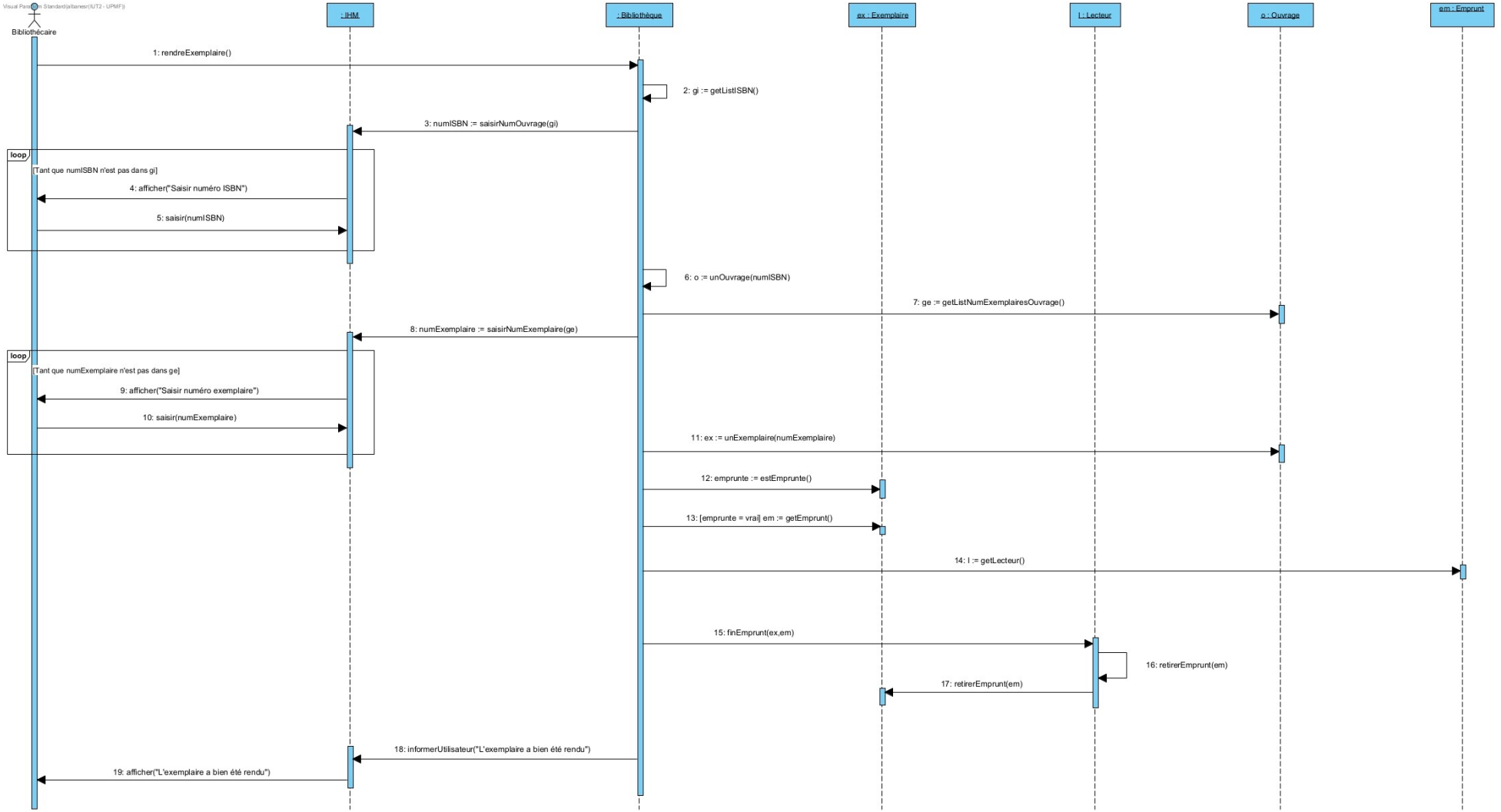

Visual Paradigm Standard(albanest(IUT2 - UPMF))
Bibliothécaire
: IHM
: Bibliothèque
e : Exemplaire
o : Ouvrage
1: nouvelExemplaire()
2: gi := getListISBN()
3: numISBN := saisirNumOuvrage(gi)
loop
[tant que numISBN n'est pas dans gi]
4: afficher("Saisir numéro ISBN")
5: saisir(numISBN)
6: o := unOuvrage(numISBN)
7: dateParution := getDateParution()
8: InfosExemplaire := saisirInfosExemplaire(dateParution)
infosExemplaire = (dateRecep, nbExemplairesEntres, nbNonEmpruntables)
dateRecep postérieure à dateParution et antérieure ou égale à la date du jour
loop
[dateRecep < dateParution || dateRecep > date du jour]
9: afficher("date de réception")
10: saisir(dateRecep)
11: afficher("nbExemplairesEntres", "nbNonEmpruntables")
12: saisir(nbExemplairesEntres, nbNonEmpruntables)
loop
[NbNomEmpruntables]
13: ajouterExemplaire(dateRecep, false)
14: incrementerNumDernierExemplaire()
15: e := créer Exemplaire(dateRecep, false, o)
16: lierOuvrage(o)
17: lierExemplaire(e)
loop
[nbExemplairesEntrés - nbNonEmpruntables]
18: ajouterExemplaire(dateRecep, true)
19: incrementerNumDernierExemplaire()
20: e := créer Exemplaire(dateRecep, true, o)
21: lierOuvrage(o)
22: lierExemplaire(e)
23: informerUtilisateur("Création du ou des exemplaire(s)",true)
24: afficher("Le(s) exemplaire(s) a/ont bien été créé(s)")


Visual Paradigm Standard(albanest(IUT2 - UPMF))
Bibliothécaire
: IHM
: Bibliothèque
l: Lecteur
em : Emprunt
ex : Exemplaire
o : Ouvrage
1: consulterLecteur()
2: gl : getListNumLecteur()
3: numLecteur := saisirNumLecteur(gl)
loop
[tant que numLecteur n'est pas dans gl]
4: afficher("Saisir numéro lecteur")
5: saisir(numLecteur)
6: l := unLecteur(numLecteur)
7: infosLecteur := getInfosLecteur()
infosLecteur = (numLecteur, nomLecteur, prenomLecteur, dateNaissanceLecteur, mailLecteur) : convention d'écriture
8: afficherInfosLecteur(infosLecteur)
9: afficher("Numéro du lecteur : ", numLecteur)
10: afficher("Nom du lecteur : ", nomLecteur)
11: afficher("Prénom du lecteur : ", prenomLecteur)
12: afficher("Date de naissance du lecteur : ", dateNaissanceLecteur)
13: afficher("Email du lecteur : ", mailLecteur)
14: collecEmprunts := getEmprunts()
loop
[em : collecEmprunts]
15: dateEmprunt := getDateEmprunt()
16: dateRetour := getDateRetour()
17: ex := getExemplaire()
18: numExemplaire := getNumExemplaire()
19: o := getOuvrage()
20: titre := getTitre()
21: numISBN := getNumISBN()
22: afficherInfosEmprunt(infosEmpruntsLecteur)
infosEmpruntsLecteur := (titre, numISBN, numExemplaire, dateEmprunt, dateRetour)
23: afficher("Titre : ", titre)
24: afficher("Numéro ISBN : ", numISBN)
25: afficher("Numéro exemplaire : ", numExemplaire)
26: afficher("Date emprunt : ", dateEmprunt)
27: afficher("Date retour : ", dateRetour)


Visual Paradigm Standard(albanesr(IUT2 - UPMF))
Bibliothécaire
: IHM
: Bibliothèque
o : ouvrage
1: consulterOuvrage()
2: gi := getListISBN()
3: numISBN := saisirNumOuvrage(gi)
loop
[tant que numISBN n'est pas dans gi]
4: afficher("Saisir numéro ISBN")
5: saisir(numISBN)
6: o := unOuvrage(numISBN)
7: infosOuvrage := getInfosOuvrage()
8: afficherInfosOuvrage(infosOuvrage)
infosOuvrage = (titre, numISBN, nomsAuteurs, nomEditeur, dateParution, publicVise) : convention d'écriture
9: afficher("Titre de l'ouvrage : ", titre)
10: afficher("Numéro ISBN de l'ouvrage :", numISBN)
11: afficher("Nom(s) de(s) auteur(s) : " nomsAuteurs)
12: afficher("Editeur de l'ouvrage : ", nomEditeur)
13: afficher("Date de parution de l'ouvrage : ", dateParution)
14: afficher("Public visé de l'ouvrage : ", publicVise)


Visual Paradigm Standard(Bibliaire(IUT2 - UPMF))
Bibliothécaire
IHM
Bibliothèque
o : Ouvrage
ex : Exemplaire
em : Emprunt
l : Lecteur
1: consulterExemplairesOuvrage()
2: gi := getListISBN()
3: numISBN := saisirNumOuvrage(gi)
loop
[Tant que numOuvrage n'est pas dans gi]
4: afficher("Saisir numéro ISBN")
5: saisir(numISBN)
6: o := unOuvrage(numISBN)
7: titre := getTitre()
8: afficherInfosOuvrage(numISBN, titre)
9: afficher("Numéro ISBN de l'ouvrage :", numISBN)
10: afficher("Titre de l'ouvrage : ", titre)
11: collectExemplaires := getExemplaires()
loop
[ex : collectExemplaires]
12: numExemplaire := getNumExemplaire()
13: emprunte := estEmprunte()
alt
[emprunte = true]
14: em := getEmprunt()
15: dateEmprunt := getDateEmprunt()
16: dateRetour := getDateRetour()
17: l := getLecteur()
18: numLecteur := getNumLecteur()
19: nomLecteur := getNomLecteur()
20: prenomLecteur := getPrenomLecteur()
21: afficherInfosExemplaireOuvrage(infosExemplaires)
22: afficher("Exemplaire numero : ", numExemplaire)
23: afficher("Numéro lecteur : ", numLecteur)
24: afficher("Nom : ", nomLecteur)
25: afficher("Prénom : ", prenomLecteur)
26: afficher("Date d'emprunt : ", dateEmprunt)
27: afficher("Date de retour : ", dateRetour)
InfosExemplaires : numExemplaire, numLecteur, nomLecteur, prenomLecteur, dateEmprunt, dateRetour
28: empruntable := getEstEmpruntable()
29: afficherInfosExemplaireOuvrage(numExemplaire, empruntable)
alt
[estEmpruntable = true]
30: afficher("Exemplaire numero : ", numExemplaire, "disponible")
31: afficher("Exemplaire numero : ", numExemplaire, "non empruntable")


Visual Paradigm Standard(albanesi(IUT2 - UPMF))
Bibliothécaire
: IHM
: Bibliothèque
ex : Exemplaire
l : Lecteur
o : Ouvrage
em : Emprunt
1: emprunterExemplaire()
2: gl := getListNumLecteur()
3: numLecteur := saisirNumLecteur(gl)
loop
[Tant que numLecteur n'est pas dans gl]
4: afficher("Saisir numéro lecteur")
5: saisir(numLecteur)
6: l := unLecteur(numLecteur)
7: sature := estSature()
8: [sature = false] gi := getListISBN()
9: numISBN := saisirNumOuvrage(gi)
loop
[Tant que numISBN n'est pas dans gi]
10: afficher("Saisir numéro ISBN")
11: saisir(numISBN)
12: o := unOuvrage(numISBN)
13: ge := getListNumExemplairesOuvrage()
14: numExemplaire := saisirNumExemplaire(ge)
loop
[Tant que numExemplaire n'est pas dans ge]
15: afficher("Saisir numéro exemplaire")
16: saisir(numExemplaire)
17: ex := unExemplaire(numExemplaire)
18: disponible := estDisponible()
19: [disponible = vrai] age := getAgeLecteur()
20: publicVise := getPublicVise()
21: empruntableParLecteur := verifAdequationPublic(age, publicVise)
22: [empruntableParLecteur = vrai] ajouterEmprunt(ex)
23: em := créer Emprunt (l,ex)
24: lierExemplaire(ex)
25: lierLecteur(l)
26: LierEmprunt(em)
27: setEmprunt(em)
28: informerUtilisateur("Emprunt de l'exemplaire", true)
29: afficher("L'exemplaire a bien été emprunté")


Visual Paradigm Standard(jalbanesr(IUT2 - UPMF))
Bibliothécaire
: IHM
: Bibliothèque
ex : Exemplaire
l : Lecteur
o : Ouvrage
em : Emprunt
1: rendreExemplaire()
2: gi := getListISBN()
3: numISBN := saisirNumOuvrage(gi)
loop
[Tant que numISBN n'est pas dans gi]
4: afficher("Saisir numéro ISBN")
5: saisir(numISBN)
6: o := unOuvrage(numISBN)
7: ge := getListNumExemplairesOuvrage()
8: numExemplaire := saisirNumExemplaire(ge)
loop
[Tant que numExemplaire n'est pas dans ge]
9: afficher("Saisir numéro exemplaire")
10: saisir(numExemplaire)
11: ex := unExemplaire(numExemplaire)
12: emprunte := estEmprunte()
13: [emprunte = vrai] em := getEmprunt()
14: l := getLecteur()
15: finEmprunt(ex,em)
16: retirerEmprunt(em)
17: retirerEmprunt(em)
18: informerUtilisateur("L'exemplaire a bien été rendu")
19: afficher("L'exemplaire a bien été rendu")

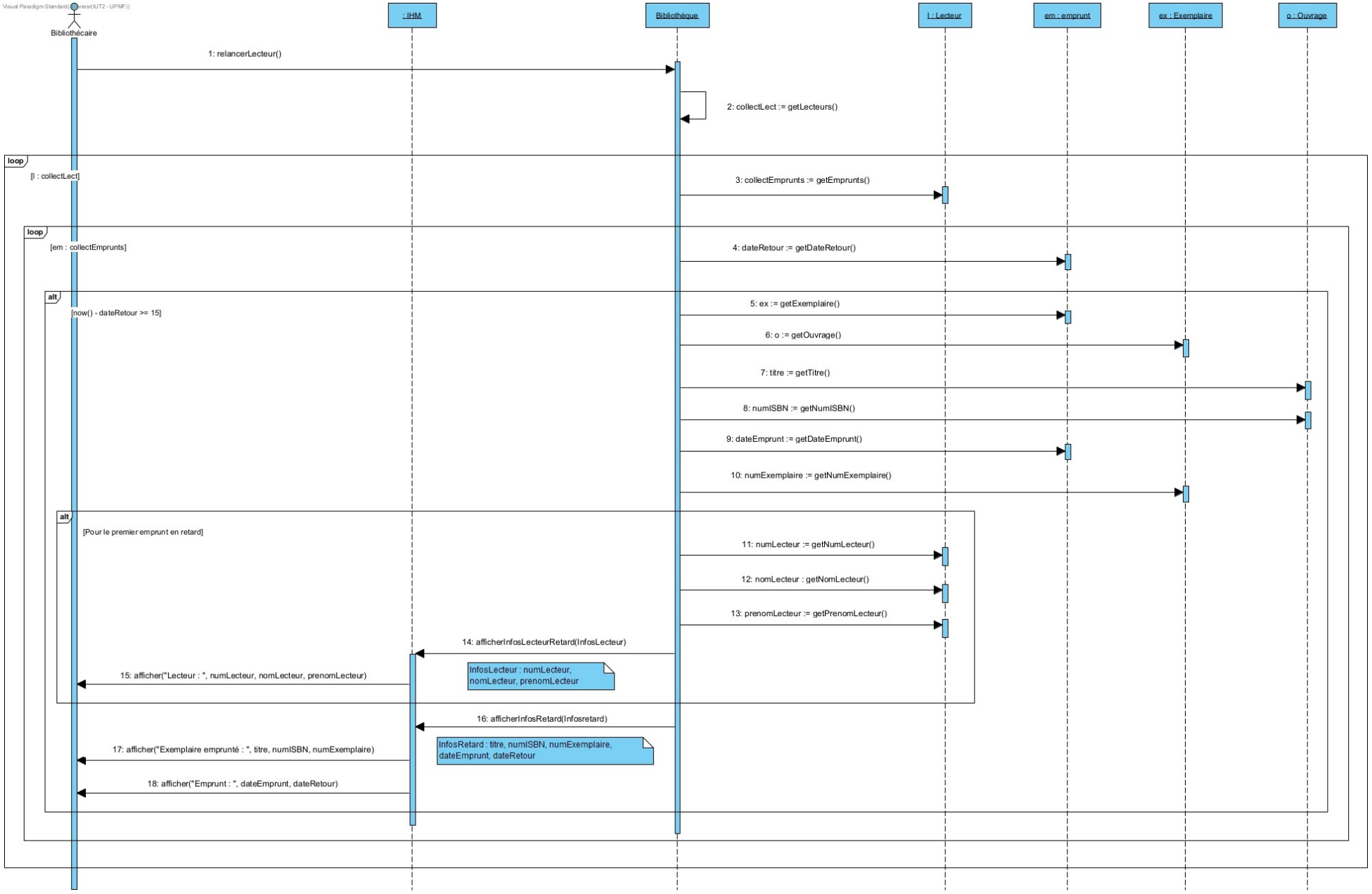

Visual Paradigm Standard(Etudiant(IUT2 - UPMF))
Bibliothécaire
: IHM
Bibliothèque
l : Lecteur
em : emprunt
ex : Exemplaire
o : Ouvrage
1: relancerLecteur()
2: collectLect := getLecteurs()
loop
[l : collectLect]
3: collectEmprunts := getEmprunts()
loop
[em : collectEmprunts]
4: dateRetour := getDateRetour()
alt
[now() - dateRetour >= 15]
5: ex := getExemplaire()
6: o := getOuvrage()
7: titre := getTitre()
8: numISBN := getNumISBN()
9: dateEmprunt := getDateEmprunt()
10: numExemplaire := getNumExemplaire()
alt
[Pour le premier emprunt en retard]
11: numLecteur := getNumLecteur()
12: nomLecteur : getNomLecteur()
13: prenomLecteur := getPrenomLecteur()
14: afficherInfosLecteurRetard(InfosLecteur)
InfosLecteur : numLecteur, nomLecteur, prenomLecteur
15: afficher("Lecteur : ", numLecteur, nomLecteur, prenomLecteur)
16: afficherInfosRetard(Infosretard)
InfosRetard : titre, numISBN, numExemplaire, dateEmprunt, dateRetour
17: afficher("Exemplaire emprunté : ", titre, numISBN, numExemplaire)
18: afficher("Emprunt : ", dateEmprunt, dateRetour)

Diagramme de classe détaillé

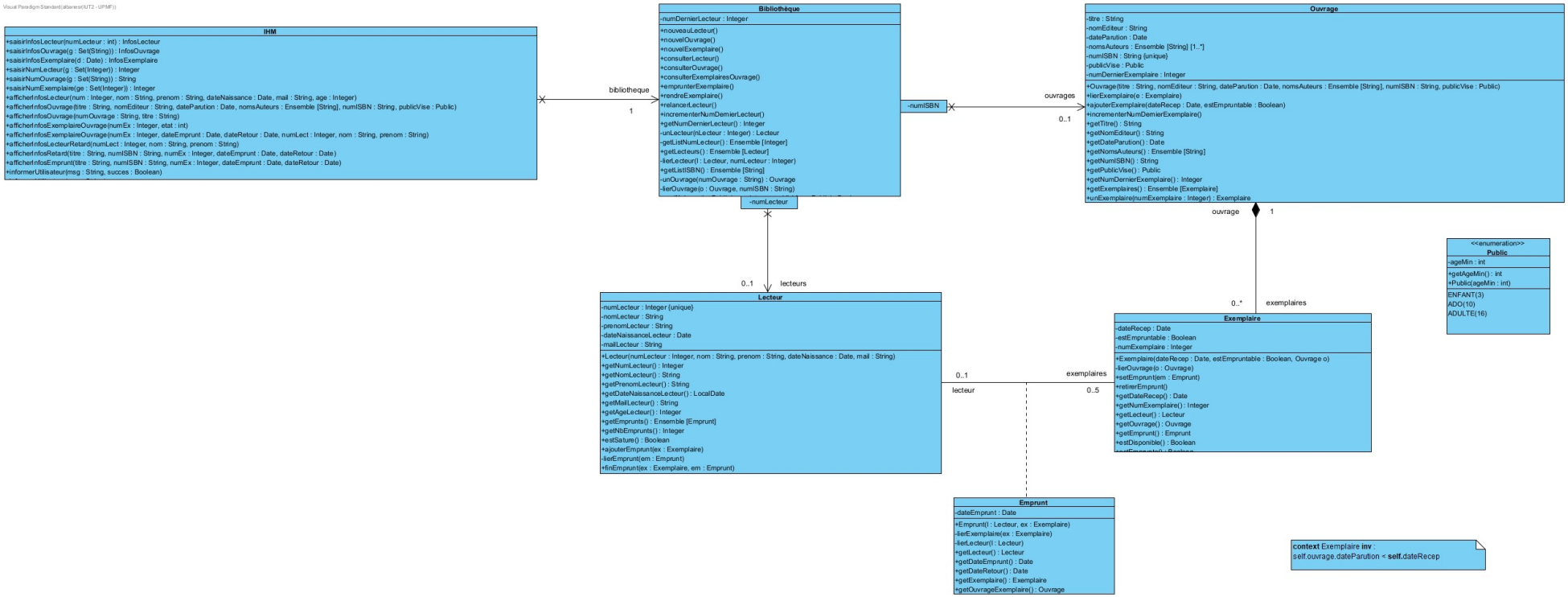
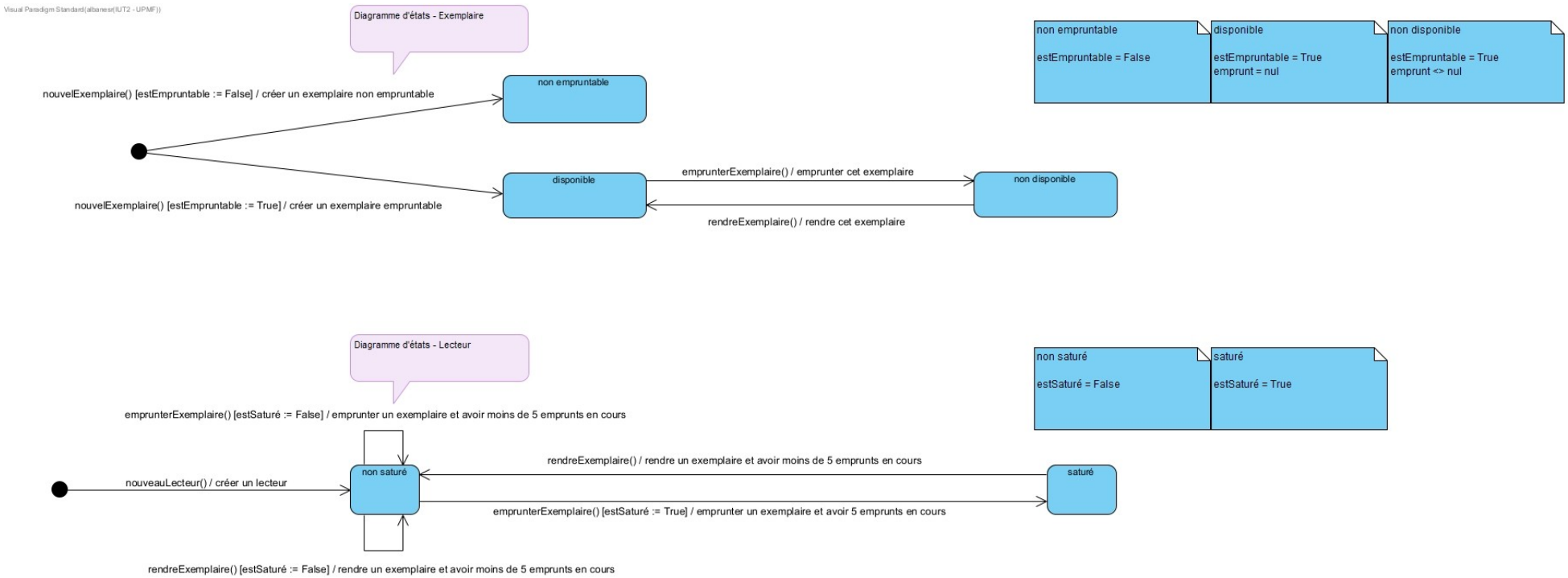


Diagramme d’états de niveau conception



IV/ Implantation

Squelette des classes

Classe Bibliothèque

```
public class Bibliotheque implements Serializable {

    private static final long serialVersionUID = 1L ;  // nécessaire pour la sérialisation
    private Integer numDernierLecteur ;
    private Map<Integer, Lecteur> lecteurs ;  // association qualifiée par le numéro d'un lecteur
    private Map<String, Ouvrage> ouvrages ;  // association qualifiée par l'ISBN d'un ouvrage
    private Map<String, Exemplaire> exemplaires;

    public Bibliotheque();

    public void nouveauLecteur (IHM ihm);

    public void nouvelOuvrage (IHM ihm);

    public void nouvelExemplaire(IHM ihm);

    public void consulterLecteur (IHM ihm);

    public void consulterOuvrage(IHM ihm);

    public void consulterExemplairesOuvrage (IHM ihm);

    public void emprunterExemplaire(IHM ihm);

    public void rendreExemplaire (IHM ihm);

    public void relancerLecteur (IHM ihm);


    public void incrementerNumDernierLecteur ();
```

```
public int getNumDernierLecteur ();

private Lecteur unLecteur (Integer nLecteur);

private Set <Integer> getListNumLecteur ();

private void lierLecteur (Lecteur l, Integer num);

public Set <String> getListISBN();

private Ouvrage unOuvrage(String numOuvrage);

private void lierOuvrage(Ouvrage o, String ISBN);

private Collection<Lecteur> getLecteurs ();
```

Classe emprunt

```
public class Emprunt implements Serializable{

    private LocalDate dateEmprunt ;
    private Lecteur lecteur ;
    private Exemplaire exemplaire ;

    public Emprunt (Lecteur lecteur, Exemplaire exemplaire);

    private void lierExemplaire (Exemplaire ex);

    private void lierLecteur (Lecteur l);

    public Lecteur getLecteur();

    public LocalDate getDateEmprunt();

    public LocalDate getDateRetour();

    public Exemplaire getExemplaire();

    public Ouvrage getOuvrageExemplaire();
```

Classe exemplaire

```
public class Exemplaire implements Serializable{

    // Attributs
    private static final long serialVersionUID = 1L;  // nécessaire pour la sérialisation
    private LocalDate dateRecep;
    private Boolean estEmpruntable;
    private final Integer numExemplaire;
    private Ouvrage ouvrage;
    private Emprunt emprunt;
}

    public Exemplaire(LocalDate dateRecep, Boolean estEmpruntable, Ouvrage ouvrage);

    private void lierOuvrage (Ouvrage ouvrage);
    public LocalDate getDateRecep();

    public Integer getNumExemplaire();

    public boolean estDisponible();

    public Lecteur getLecteur();

    public Ouvrage getOuvrage();

    public Emprunt getEmprunt();

    public void retirerEmprunt();

    public void lierEmprunt (Emprunt em);
```

Classe Lecteur

```
public class Lecteur implements Serializable {

    private static final long serialVersionUID = 1L ;
    private final Integer numLecteur ;
    private String nomLecteur ;
    private String prenomLecteur ;
    private LocalDate dateNaissanceLecteur ;
    private String mailLecteur ;
    private HashSet<Emprunt> emprunts ;
}

    public Lecteur (Integer numLecteur, String nomLecteur, String prenomLecteur, LocalDate dateNaissanceLecteur, String mailLecteur);

    public Integer getNumLecteur();

    public String getNomLecteur();

    public String getPrenomLecteur();

    public LocalDate getDateNaissanceLecteur();

    public String getMailLecteur();

    public Integer getAgeLecteur();

    public HashSet<Emprunt> getEmprunts();

    public Integer getNbEmprunts();

    public Boolean estSature ();

    public void nouvelEmprunt (Exemplaire e);
```

```
    private void lierEmprunt (Emprunt em);

    public void finEmprunt (Exemplaire e, Emprunt em);

    public void retirerEmprunt(Emprunt em);public class Ouvrage implements Serializable {

    // Attributs
    private static final long serialVersionUID = 1L;
    private String titre;
    private String nomEditeur;
    private LocalDate dateParution;
    private ArrayList<String> nomAuteurs;
    private final String numISBN;
    private Public publicVise;
    private Integer numDernierExemplaire;
    private ArrayList <Exemplaire> exemplaires;
}

    public Ouvrage(String titre, String nomEditeur, LocalDate dateParution, ArrayList<String> nomAuteurs, String numISBN, Public publicVise);

    public void ajouterExemplaire (LocalDate dateRecep, Boolean estEmpruntable);

    public void incrementerNumDernierExemplaire();

    public void lierExemplaire (Exemplaire e);

    public String getTitre();

    public String getNomEditeur();

    public LocalDate getDateParution();

    public ArrayList<String> getNomsAuteurs();
```

```
public String getNumISBN();

public Public getPublicVise();

public Integer getNumDernierExemplaire();

public ArrayList <Exemplaire> getExemplaires();

public Exemplaire getUnExemplaire (Integer numExemplaire);

public ArrayList <Integer> getListNumExemplairesOuvrage();

public boolean verifAdequationPublic(Integer age, Public publicVise);
```

Classe Ouvrage

```
public class Ouvrage implements Serializable {

    private static final long serialVersionUID = 1L;
    private String titre;
    private String nomEditeur;
    private LocalDate dateParution;
    private ArrayList<String> nomAuteurs;
    private final String numISBN;
    private Public publicVise;
    private Integer numDernierExemplaire;
    private ArrayList <Exemplaire> exemplaires;
}

    public Ouvrage(String titre, String nomEditeur, LocalDate dateParution, ArrayList<String> nomAuteurs, String numISBN, Public publicVise);

    public void ajouterExemplaire (LocalDate dateRecep, Boolean estEmpruntable);

    public void incrementerNumDernierExemplaire();

    public void lierExemplaire (Exemplaire e);

    public String getTitre();

    public String getNomEditeur();

    public LocalDate getDateParution();

    public ArrayList<String> getNomsAuteurs();

    public String getNumISBN();

    public Public getPublicVise();
```

```
public Integer getNumDernierExemplaire();

public ArrayList <Exemplaire> getExemplaires();

public Exemplaire getUnExemplaire (Integer numExemplaire);

public ArrayList <Integer> getListNumExemplairesOuvrage();

public boolean verifAdequationPublic(Integer age, Public publicVise);
```

Code java

On présente ici quatre cas d'utilisation mis en œuvre dans Bibliotheque.java : consulterLecteur, consulterExemplairesOuvrage, emprunterExemplaire et rendreExemplaire. Ils ont été choisis pour les interactions entre objets des différentes classes du modèle de notre système de gestion de bibliothèque qu'ils mettent en œuvre. Les cas d'utilisation doivent se lire avec les diagrammes de séquence détaillés dont ils découlent (cf plus haut).

Cas d'utilisation 'consulterLecteur'

```java
public void consulterLecteur (IHM ihm) {
    Set <Integer> listNumLecteur = getListNumLecteur() ;
    if (listNumLecteur.size()>0){
        ES.afficherSetInt(listNumLecteur, "Liste des lecteurs existants :");
        Integer numLecteur = ihm.saisirNumLecteur(listNumLecteur) ;
        Lecteur l = unLecteur (numLecteur) ;
        ihm.afficherInfosLecteur(l.getNumLecteur(), l.getNomLecteur(), l.getPrenomLecteur(),
         l.getDateNaissanceLecteur(), l.getMailLecteur(), l.getAgeLecteur()) ;
        HashSet<Emprunt> collecEmprunts = l.getEmprunts();
        if (collecEmprunts.size()>0){
            ES.afficherTitre("== affichage des emprunts en cours ==");
            for (Emprunt em : collecEmprunts){
                LocalDate dateEmprunt = em.getDateEmprunt();
                LocalDate dateRetour = em.getDateRetour();
                Exemplaire ex = em.getExemplaire();
                Integer numExemplaire = ex.getNumExemplaire();
                Ouvrage o = ex.getOuvrage();
                String titre = o.getTitre();
                String numISBN = o.getNumISBN();
                ihm.afficherInfosEmprunt(titre, numISBN, numExemplaire, dateEmprunt, dateRetour);
            }
        }
        else {
            ihm.informerUtilisateur("Le lecteur n'a aucun emprunt en cours.");
        }
        ihm.informerUtilisateur("Consultation d'un lecteur et de ses emprunts",true);
    }
    else{
        ihm.informerUtilisateur("Aucun lecteur dans la base.");
        ihm.informerUtilisateur("Consultation de lecteurs", false);
    }
}
```

Cas d'utilisation 'consulterExemplairesOuvrage'

```
public void consulterExemplairesOuvrage (IHM ihm) {
    Set <String> listISBN = getListISBN () ;
    if (listISBN.size()>0){
        ES.afficherSetStr(listISBN, "Liste des ouvrages existants :");
        String numISBN = ihm.saisirNumOuvrage(listISBN) ;
        Ouvrage o = unOuvrage (numISBN) ;
        ihm.afficherInfosOuvrage(o.getNumISBN(), o.getTitre()) ;
        ArrayList <Exemplaire> exemplaires = o.getExemplaires() ;
        if (exemplaires.size()>0){
            for (Exemplaire ex : exemplaires ) {
                if (ex.estEmprunte()) {
                    Emprunt em = ex.getEmprunt();
                    Lecteur l = em.getLecteur();
                    ihm.afficherInfosExemplaireOuvrage(ex.getNumExemplaire(), em.getDateEmprunt(), em.getDateRetour(),
                    l.getNumLecteur(), l.getNomLecteur(), l.getPrenomLecteur());
                }
                else if (ex.estDisponible()) {
                    ihm.afficherInfosExemplaireOuvrage(ex.getNumExemplaire(), 3);
                }
                else {
                    ihm.afficherInfosExemplaireOuvrage(ex.getNumExemplaire(), 1);
                }
            }
            ihm.informerUtilisateur("Consultation d'exemplaires ",true);
        }
        else {
            ihm.informerUtilisateur("pas d'exemplaires /");
            ihm.informerUtilisateur("Consultation d'exemplaires ",false);
        }
    }
    else{
        ihm.informerUtilisateur("Il n'existe pas encore d'ouvrages, il n'existe donc pas encore d'exemplaires.");
        ihm.informerUtilisateur("Consultation d'exemplaires ",false);
```

Cas d'utilisation 'emprunterExemplaire'

```
public void emprunterExemplaire(IHM ihm) {
    Set <Integer> listNumLecteur = getListNumLecteur() ;
    if (listNumLecteur.size()>0){
        ES.afficherSetInt(listNumLecteur,"Liste des lecteurs existants : ");
        Integer numLecteur = ihm.saisirNumLecteur(listNumLecteur);
        Lecteur l = unLecteur(numLecteur);
        boolean sature = l.estSature();
        if (sature == false){
            Set<String> listISBN = getListISBN();
            if (listISBN.size()>0){
                ES.afficherSetStr(listISBN, "Liste des ouvrages existants : ");
                String numISBN = ihm.saisirNumOuvrage(listISBN);
                Ouvrage o = unOuvrage(numISBN);
                ArrayList <Integer> listNumExemplaire = o.getListNumExemplairesOuvrage() ;
                if(listNumExemplaire.size()>0){
                    ES.afficherArrayInt(listNumExemplaire, "Liste des exemplaires existants");
                    Integer numExemplaire = ihm.saisirNumExemplaire(listNumExemplaire);
                    Exemplaire ex = o.unExemplaire(numExemplaire);
                    if(ex.estDisponible()){
                        Integer age = l.getAgeLecteur();
                        Public publicVise = o.getPublicVise();
                        if(verifAdequationPublic(age, publicVise)){
                            l.ajouterEmprunt(ex);
                            ihm.informerUtilisateur("L'exemplaire a bien été emprunté.");
                            ihm.informerUtilisateur("Emprunt de l'exemplaire",true);
                        }
```

Cas d'utilisation 'rendreExemplaire'

```
public void rendreExemplaire (IHM ihm) {
    Set <String> listISBN = getListISBN() ;
    if (listISBN.size() > 0) {
        ES.afficherSetStr(listISBN, "Liste des ouvrages existants :") ;
        String numISBN = ihm.saisirNumOuvrage(listISBN) ;
        Ouvrage o = unOuvrage(numISBN) ;
        ArrayList <Integer> listNumExemplaires = o.getListNumExemplairesOuvrage() ;
        if (listNumExemplaires.size() > 0) {
            ES.afficherLibelle("Liste des exemplaires existants : " + listNumExemplaires) ;
            Integer numExemplaire = ihm.saisirNumExemplaire(listNumExemplaires) ;
            Exemplaire ex = o.unExemplaire(numExemplaire) ;
            if (ex.estEmprunte()) {
                Emprunt em = ex.getEmprunt() ;
                Lecteur l = em.getLecteur() ;
                l.finEmprunt(ex, em) ;
                ihm.informerUtilisateur("L'exemplaire a bien été rendu.");
                ihm.informerUtilisateur("Retour de l'exemplaire", true) ;
            }
            else {
                ihm.informerUtilisateur("Cet exemplaire n'est pas emprunté, il ne peut être rendu.\nRetour au menu");
            }
        }
        else {
            ihm.informerUtilisateur("Cet ouvrage n'a pas d'exemplaires.\nRetour au menu.") ;
        }
    }
    else {
        ihm.informerUtilisateur("Il n'existe pas encore d'ouvrages.\nRetour au menu.") ;
    }
}
```

Conclusion

Au terme de ce projet, nous avons réussi à répondre à la demande de notre client en implémentant chacun des cas d'utilisation. Notre travail nous a amené à penser les possibles améliorations de ce système, comme par exemple la réservation d'un exemplaire quand aucun exemplaire n'est disponible actuellement ou l'envoi d'un message automatique pour chaque emprunt en retard de 15 jours. Une autre amélioration possible serait la création d'une version graphique, peut être envisageable plus tard grâce à nos futurs cours d'interfaces homme- machine.

Nous avons donc été efficaces au sens de la gestion de ce projet. Pour ce qu'il en est de l'efficience, ce travail de groupe nous a permis d'apprendre à gérer certaines éventualités et à repérer plusieurs points à améliorer. L'emploi de la méthode UML nous a d'abord paru un peu lourde au regard du nombre de cas d'utilisation à traiter. Cependant, nous avons concédé une grande partie du délai de notre projet à la réalisation des différents diagrammes UML, ce qui nous a finalement aidé à mieux structurer notre réflexion et à ne pas oublier d'informations. Les diagrammes de séquence détaillés nous ont particulièrement été utiles pour la phase d'implémentation. En effet, les efforts appliqués à leur réalisation nous ont permis de coder assez rapidement, il nous suffisait de lire nos diagrammes et le code en découlait naturellement. L'aspect strict et très cadré de la démarche UML, nous a donc fait économiser du temps en réduisant les incertitudes mais aussi de l'énergie. En effet, au terme de la première itération, notre code s'est exécuté sans erreur du premier coup, ce qui n'est pas le cas de nos programmes dans les autres matières, où bien souvent nous codons sans plan préalablement défini et à la compilation nous devons déchiffrer plusieurs lignes de messages d'erreurs. Ce travail de groupe nous a aidé à mieux appréhender les étapes du développement d'un projet orienté objet. La lecture de la partie sept du cours pouvait laisser supposer que c'était un processus linéaire, où chaque étape découlait logiquement de la précédente, or la pratique nous a fait comprendre que les allers-retours entre les modèles et le code étaient indispensables. A propos des conventions de nommage, nous nous sommes rendus compte qu'uniformiser des noms de variables et méthodes été un travail complexe, en particulier lorsque nous n'avons pas encore une vision d'ensemble claire du projet. Nous avons voulu remettre ça à plus tard, ce qui nous a demandé du temps supplémentaire par la suite, surtout que le nombre de méthodes avait considérablement augmenté. Une des étapes qui nous a donné le plus de fil à retordre est sûrement le passage des diagrammes de séquence de haut niveau aux diagrammes de séquence détaillés. Ceux de haut niveau pouvaient laisser supposer un déroulement d'interactions assez simple, mais finalement se sont traduit en des interactions beaucoup plus complexes et développées dans les diagrammes détaillés. Dans l'ensemble c'est donc le souci de la cohérence, que ce soit entre différents diagrammes ou entre diagramme et code, qui a été le plus chronophage. Le caractère itératif de ce projet nous a permis de pouvoir apprendre de nos erreurs de la première itération et donc de gagner en efficacité. Les notions liées à la démarche UML étaient mieux maîtrisées et les outils également. En effet, l'utilisation plus fluide de Visual Paradigm et Netbeans, notamment avec Git, a été déterminante pour conclure ce projet