INSA LYON || IF-3-GI&UML

TP Génie Logiciel & UML 2017/2018

TP2: Dossier de Conception et de Tests

CHRISTOPHE ETIENNE | LYNN GHANDOUR | MUSTAFA COREKCI

02/05/2018

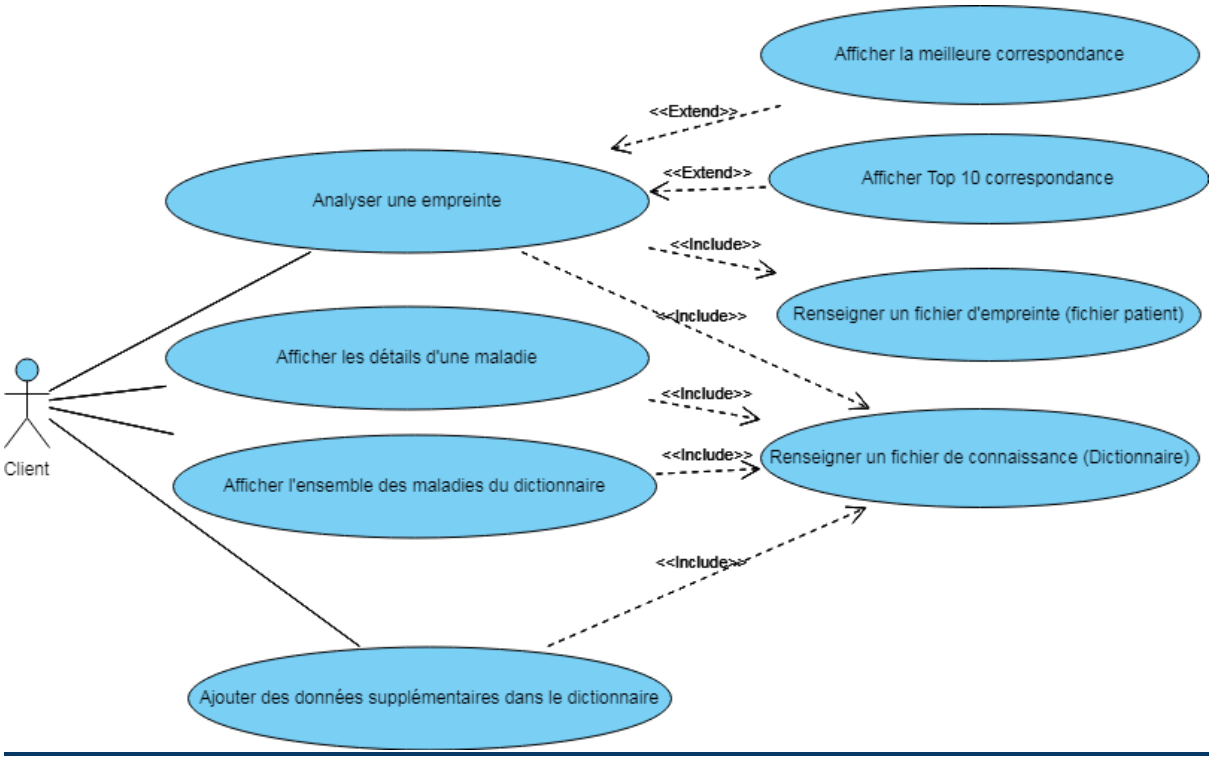
Table des matières

Conception

I. Diagramme de classe

Voir Annexe 1

II. Diagramme de cas d'utilisation



III. Diagramme de séquence

1. Renseigner Dictionnaire

a. Cas où la maladie existe déjà dans le dictionnaire

Voir Annexe 2

b. Cas où la maladie n'existe pas dans le dictionnaire

Voir Annexe 3

2. Renseigner FichierPatient

Voir Annexe 4

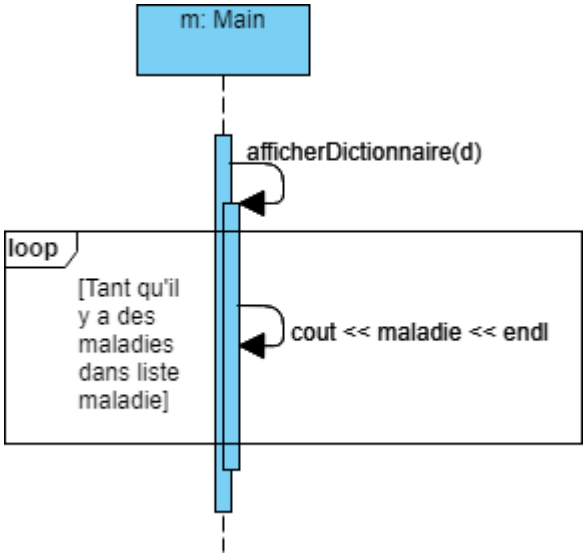
3. Afficher Top10

Voir Annexe 5

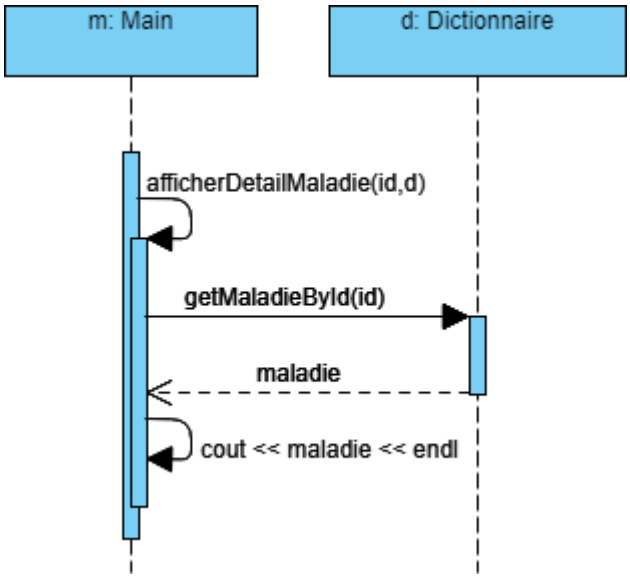
4. Afficher Meilleure Correspondance

Voir Annexe 6

5. **Afficher Dictionnaire**

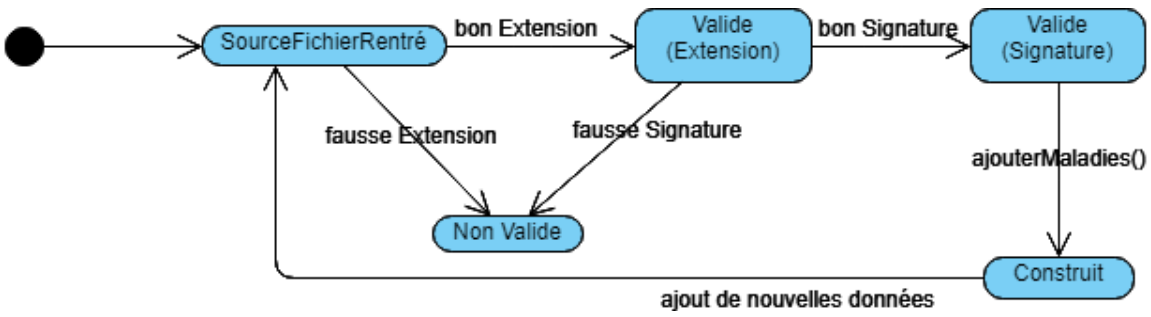


6. **Afficher détail de la Maladie**

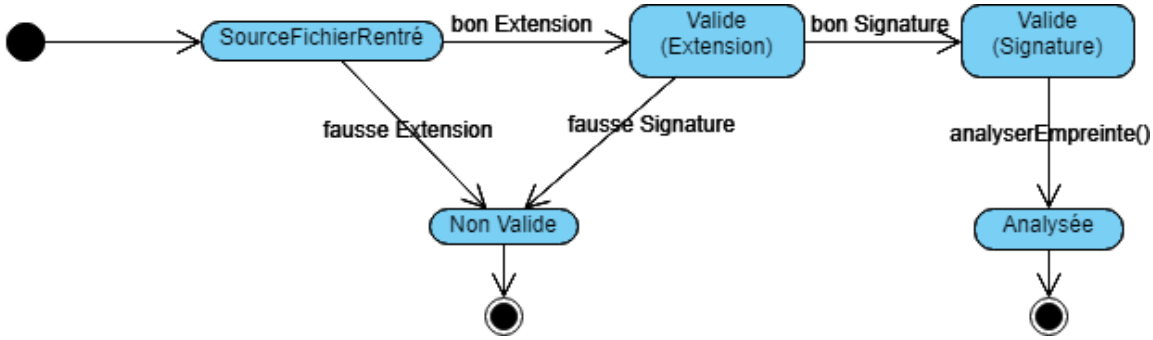


IV. Diagramme d'états-transitions

1. **Dictionnaire**



2. FichierPatient



V. Description des principaux algorithmes

1. Analyse

1.1. genererClassement(e:Empreinte,d:dictionnaire): Correspondance[]

On calcule les correspondances entre l'empreinte donnée en paramètre et toutes les maladies du dictionnaire pour obtenir un tableau de paire <idMaladie, probabilité>. On ne compte pas les cas où il n'y a pas de risque (probabilité < 20%)

1.2. calculerProbabilité(e: Empreinte, m:Maladie)

On compare les attribue de e avec les empreintes de la maladie m un à un. Soit a le nombre d'attributs identiques et n le nombre d'attribut total. La probabilité calculé est égale à : (a/n)*100, on ne garde dans le tableau de paire correspondance que la plus haute probabilité avec la maladie comparée. Si un des attribut a une valeur null a n'est pas incrémenté. Si le résultat est inférieur à 20%, la maladie n'est pas stocké dans le tableau correspondance.

2. Empreinte

2.1. ajouterAttribut(a:Attribut)

On ajoute un attribut à la liste d'attribut des empreintes

2.2. lancerAnalyse(): Analyse

Appelle la méthode générerClassement de la classe Analyse pour l'empreinte. Cette analyse est ensuite retourner

3. Maladie

3.1. ajouterEmpreinet(e:Empreinte): boolean

On ajoute un empreinte à la liste d'empreinte de la maladie. Retourne true si l'empreinte a été ajouté sinon false (si cette empreinte exister déjà dans le dictionnaire pour cette maladie par exemple).

3.2. getEmpreinteById (id: Long) : Empreinte

Retourne l'empreinte ayant l'identifiant id. Si l'empreinte n'existe pas, retourne null.

4. Dictionnaire

4.1. ajouterMaladie(emp: Empreinte): boolean

Vérifie que la maladie n'existe pas dans le dictionnaire avant de l'ajouter avec son empreinte. Si elle est déjà présente, on regarde si l'empreinte donnée pour cette maladie est déjà répertoriée dans le dictionnaire. Si ce n'est pas le cas on ajoute cette empreinte à la maladie dans le dictionnaire.

4.2. getMaladieById(id: Long): Maladie

On retourne la maladie qui possède l'identifiant id. Si la maladie n'existe dans le dictionnaire, on retourne null.

5. FichierPatient

5.1. analyserEmpreinte(): Analyse []

Pour chaque empreinte de cette classe, on appelle la méthode lancerAnalyse. Chacune des analyse retourner est regroupé dans un tableau que l'on retourne

5.2. ajouterEmpreinte(e:Empreinte): boolean

On vérifie que l'empreinte n'existe pas déjà dans la liste d'empreinte de la classe avant de l'ajouter dans la liste.

6. FichierEmpStream

6.1. lireFichierPatient(sourceFichier: String): FichierPatient

On ouvre le fichier sourceFichier.txt et on le lit grâce à un objet ifstream. On appelle la classe verifierExtension et verifierSignature. Si les deux retourne True, pour chaque ligne du fichier on crée une Empreinte correspondante que l'on ajoute à un FichierPatient. On retourne ce dernier.

6.2. lireDictionnaire(sourceFichier: String): Dictionnaire

On ouvre le fichier sourceFichier.txt et on le lit grâce à un objet ifstream. On appelle la classe verifierExtension et verifierSignature. Si les deux retourne True, pour chaque ligne du fichier on crée une Maladie correspondante que l'on ajoute au Dictionnaire. On retourne ce dernier.

6.3. verifierExtension(sourceFichier: String): boolean

On récupère l'extension en manipulant le nom du fichier.Retourne vrai si il s'agit d'un extension .txt (celle choisie dans notre cas) sinon false.

6.4. verifierSignature(): boolean

On lit et récupère la signature du fichier et on l'a compare avec la signature renseigner lors du premier chargement de fichier. Retourne true si elles sont égale, sinon false. Dans le cas où il s'agit du premier chargement (signature = null) on retourne forcément true.

7. main

7.1. renseignerDictionnaire(sourceFichier: String): Dictionnaire
On crée un FichierEmpStream à partir duquel on appelle la méthode lireDictionnaire avec comme paramètre le fichier sourceFichier.

7.2. renseignerFichierPatient(sourceFichier: String): FichierPatient
On crée un FichierEmpStream à partir du quel on appelle la méthode lireFichierPatient avec comme paramètre le fichier sourceFichier.

7.3. afficherTop10(d: Dictionnaire,fp: FichierPatient)
On appelle ma méthode analyserEmpreinte pour le FichierPatient en paramètre fp avec en paramètre le Dictionnaire en paramètre d. Pour chacune des analyse du tableau d'analyse retournée par cette méthode on affiche les maladies des 10 premières paires du tableau correspondance. (Plus ou moins de 10 maladie peuvent être affichées : voir les besoins fonctionnels du TP1)

7.4. afficherMeilleurCorresp(d: Dictionnaire,fp: FichierPatient)
On appelle ma méthode analyserEmpreinte pour le FichierPatient en paramètre fp avec en paramètre le Dictionnaire en paramètre d. Pour chacune des analyse du tableau d'analyse retournée par cette méthode on affiche la maladie de la première paire du tableau correspondance. (plus ou moins d'une maladie peut être affichée : voir les besoins fonctionnels du TP1)

7.5. afficherDictionnaire(d: Dictionnaire)
On affiche toutes les maladies contenues dans la liste de maladie du dictionnaire d donné en paramètre

7.6. afficherDetailMaladie(id: Long, d: Dictionnaire)
On affiche la maladie dont l'identifiant dans le dictionnaire d est id.

VI. Structuration des données

Dans notre application nous retrouvons 3 structures de données.

- Le Dictionnaire, pour stocker les maladie, possède une map

- Le FichierPatient, pour stocker ses empreintes, possède une multimap (car il peut y avoir plusieurs fois la même empreinte à analyser
- L'analyse, pour stocker les résultats, possède un tableau de pair <Long, double> pour avoir une paire de maladie avec la probabilité qui lui correspond.

Plan de tests d'intégration

Pour chaque service que nous proposons dans notre application nous mettons en place un test d'intégration mis en place dans la classe main.

1. Renseigner un Dictionnaire

1.1. Lancer la méthode renseignerDictionnaire avec un fichier valide (bonne extension et bonne signature) non vide, au début de l'exécution de l'application (premier dictionnaire renseigner).
→ Les maladies sont chargées et stockées dans la classe Dictionnaire

1.2. Lancer la méthode renseignerDictionnaire avec un fichier valide (bonne extension et bonne signature) non vide, au milieu de l'exécution de l'application.
→ Les maladies sont chargées et stockées dans la classe Dictionnaire en plus des maladie déjà presnetes.

1.3. Lancer la méthode renseignerDictionnaire avec la bonne extension, au milieu de l'exécution de l'application, mais qui ne possède pas la signature demandée
→ Un message d'erreur est affiché

1.4. Lancer la méthode renseignerDictionnaire avec un fichier possédant la mauvaise extension (dans notre cas cela signifie différente de .txt)
→ Un message d'erreur est affiché

1.5. Lancer la méthode renseignerDictionnaire avec un fichier valide (bonne extension et bonne signature) vide.
→ Un message d'erreur est affiché

2. Renseigner un Fichier Patient

2.1. Lancer la méthode renseignerFichierPatient avec un fichier valide (bonne extension et bonne signature) non vide, au début de l'exécution de l'application (premier fichier renseigner).
→ Un message d'avertissement est afficher pour signaler qu'il n'y a pas encore de dictionnaire.

2.2. Lancer la méthode renseignerFichierPatient avec un fichier valide (bonne extension et bonne signature) non vide, au milieu de l'exécution de l'application.
→ Les empreintes sont chargées et stockées dans une classe FichierPatiente. Les précédentes empreintes sont supprimées.

2.3. Lancer la méthode renseignerFichierPatient avec la bonne extension, au milieu de l'exécution de l'application, mais qui ne possède pas la signature demandée
→ Un message d'erreur est affiché

2.4. Lancer la méthode renseignerFichierPatient avec un fichier possédant la mauvaise extension (dans notre cas cela signifie différente de .txt)
→ Un message d'erreur est affiché

2.5. Lancer la méthode renseignerFichierpPatient avec un fichier valide (bonne extension et bonne signature) vide.
→ Un message d'erreur est affiché

3. Afficher le Top 10 des maladie

3.1. Lancer la méthode afficherTop10 avant que le Dictionnaire et/ou le FichierPatient ne soit renseigné.
→ Un message d'avertissement est affiché

3.2. Lancer la méthode afficherTop10 avec un FichierPatient qui possède une empreinte et un dictionnaire qui présente plus de 20 maladies
→Les 10 maladie les plus probables pour cette empreinte sont affichées. Si des maladie présente une probailité superieur à 70%, un message « Une analyse supplémentaire pour vérification est conseillée. » est rajouté.

3.3. Lancer la méthode afficherTop10 avec un FichierPatient qui possède plusieurs empreintes et un dictionnaire qui présente plus de 20 maladies
→Les 10 maladie les plus probables pour chacune des empreintes sont affichées

3.4. Lancer la méthode affciherTop10 avec un FichierPatient qui possède une ou plusieurs empreintes et un dictionnaire qui présente moins de 10 maladies.

→ Les maladies les plus probables (moins de 10) sont affichées pour chacune des empreintes.

3.5. Lancer la méthode afficherTop10 avec un FichierPatient présantant une empreinte et un dictionnaire tel qu'aucune maladie du dictionnaire ne présente de risque pour l'empreinte du patient (probabilité < 20%).
→ Un message d'informations est affichées.

4. Afficher la meilleure correspondance de maladie

4.1. Lancer la méthode afficherMeilleurCorresp avant que le Dictionnaire et/ou le FichierPatient ne soit renseigné.
→ Un message d'avertissement est affiché

4.2. Lancer la méthode afficherMeilleurCorresp avec un FichierPatient qui possède une empreinte et un dictionnaire qui présente plus de 20 maladies
→Les 10 maladie les plus probables pour cette empreinte sont affichées. Si des maladie présente une probailité superieur à 70%, un message « Une analyse supplémentaire pour vérification est conseillée. » est rajouté.

4.3. Lancer la méthode afficherMeilleurCorrespavec un FichierPatient qui possède plusieurs empreintes et un dictionnaire qui présente plus de 20 maladies
→La maladie la plus probable pour chacune des empreintes est affichée.

4.4. Lancer la méthode afficherMeilleurCorrespavec un FichierPatient présantant une empreinte et un dictionnaire tel qu'aucune maladie du dictionnaire ne présente de risque pour l'empreinte du patient (probabilité < 20%).
→ Un message d'informations est affichées.

5. Afficher le dictionnaire

5.1. Lancer la méthode afficherDictionnaire avec un dictionnaire possédant des maladies.
→ Toutes les maladies contenue dans le dictionnaire sont affichées

5.2. Lancer la méthode afficherDictionnaire avec un dictionnaire vide
→ Un message d'avertissement prevenant que le dictionnaire est vide, est affiché.

6. **Afficher le détail d'une maladie**
   6.1. Lancer la méthode afficherDetailMaladie en donnant l'id d'une maladie qui existe dans le dictionnaire qui possède une empreinte.
   → La maladie est affiché (id, nom, empreinte)
   6.2. Lancer la méthode afficherDetailMaladie en donnant l'id d'une maladie qui existe dans le dictionnaire qui possède plusieurs empreintes.
   → La maladie est affiché (id, nom, lise d'empreintes)

   6.3. Lancer la méthode afficherDetailMaladie en donnant l'id d'une maladie qui n'existe pas dans le dictionnaire.
   → Un message d'avertissement est affiché.

Plan des Tests Unitaires

1. **Empreinte::ajouterAtribut(Attribut A)**
   1.1. Ajouter un attribut A de type string
   1.2. Ajouter un attribut A de type boolean
   1.3. Ajouter un attribut A de type int→ L'attribut est bien ajouter à la liste d'attribut de la classe empreinte. Pour chaque attribut, à partir de la classe empreinte on retrouve bien son nom, sa valeur et son type
2. **Maladie::getEmpreinteById(Long id)**
   2.1. Rechercher une empreinte dont l'id est répertorié dans la maladie
   → L'empreinte correspondant est retournée
   2.2. Rechercher une empreinte dont l'id n'est pas répertorié dans la maladie
   → On retourne null
3. **Dictionnaire::getMaladieById(Long id)**
   3.1. Rechercher une maladie dont l'id est répertorié dans le dictionnaire
   → La maladie correspondante est retournée
   3.2. Rechercher une maladie dont l'id n'est pas répertorié dans le dictionnaire
   → On retourne null
4. **Dictionnaire::ajouterMaladie(String ligne)**
   4.1. Ajouter une maladie qui n'existe pas dans le dictionnaire à partir d'une ligne présentant des attributs et un nom de maladie
   → La maladie est ajoutée au dictionnaire avec l'empreinte correspondante
   4.2. Ajouter une maladie qui existe déjà dans le dictionnaire mais pas son empreinte à partir d'une ligne présentant des attributs et un nom de maladie

→ L'empreinte de cette maladie est rajoutée à la liste d'empreinte de la maladie dans le dictionnaire.

4.3. Ajouter une maladie qui existe déjà dans le dictionnaire ainsi que son empreinte à partir d'une ligne présentant des attributs et un nom de maladie

→ Le dictionnaire n'est pas modifié

4.4. Ajouter une maladie à partir d'une ligne ne présentant pas de nom de maladie

→ Si l'empreinte n'existe pas déjà dans le dictionnaire sans idMaladie, la maladie est ajoutée avec un id et un nom null. Sinon le dictionnaire n'est pas modifié

4.5. Ajouter une maladie à partir d'une ligne où la valeur d'un des attributs est vide

→ La maladie est ajoutée au dictionnaire avec un de ses attributs avec une valeur null

5. FichierPatient::ajouterEmpreinte(Empreinte e)

5.1. Ajouter une empreinte qui n'existe pas dans le fichierPatient à partir d'une ligne présentant des attributs

→ L'empreinte est ajoutée au fichier patient

5.2. Ajouter une empreinte qui existe déjà dans le fichier patient à partir d'une ligne présentant des attributs

→ L'empreinte ajoutée au fichier patient

5.3. Ajouter une empreinte à partir d'une ligne où la valeur d'un des attributs est vide

→ L'empreinte est ajoutée au fichier patient avec un de ses attributs avec une valeur null

6. Maladie::ajouterEmpreinte(Empreinte e)

6.1. Ajouter une empreinte qui n'existe pas pour la maladie à partir d'une ligne présentant des attributs

→ L'empreinte est ajoutée à la maladie

6.2. Ajouter une empreinte qui existe déjà pour la maladie à partir d'une ligne présentant des attributs

→ L'empreinte ajoutée à la maladie

6.3. Ajouter une empreinte à partir d'une ligne où la valeur d'un des attributs est vide

→ L'empreinte est ajoutée à la maladie avec un de ses attributs avec une valeur null

7. Analyse::calculerProbabilite(Empreinte e, Maladie m)

7.1. Comparer une empreinte avec une maladie possédant qu'une empreinte :

e(01,12,True,rouge,39,ventre) et m(03,grippe[12,14,False,rouge,39,ventre]) on stocke dans le tableau correspondance la paire <03,60>

7.2. Comparer une empreinte avec une maladie possédant plusieurs empreintes
e(01,12,True,rouge,39,ventre) et m(03,grippe,[12,14,False,rouge,39,ventre],[13,12,True,rouge,39,tête]) on stocke dans le tableau correspondance la paire <03,80>

7.3. Comparer une empreinte qui a des attribut à valeur null avec une maladie ayant plusieurs empreintes
e(01,null,True,rouge,39,ventre) et m(03,grippe,[12,14,False,rouge,39,ventre],[13,12,True,rouge,39,tête]) on stocke dans le tableau correspondance la paire <03,60>

8. Analyse::genererClassement(Dictionnaire d, Empreinte e)

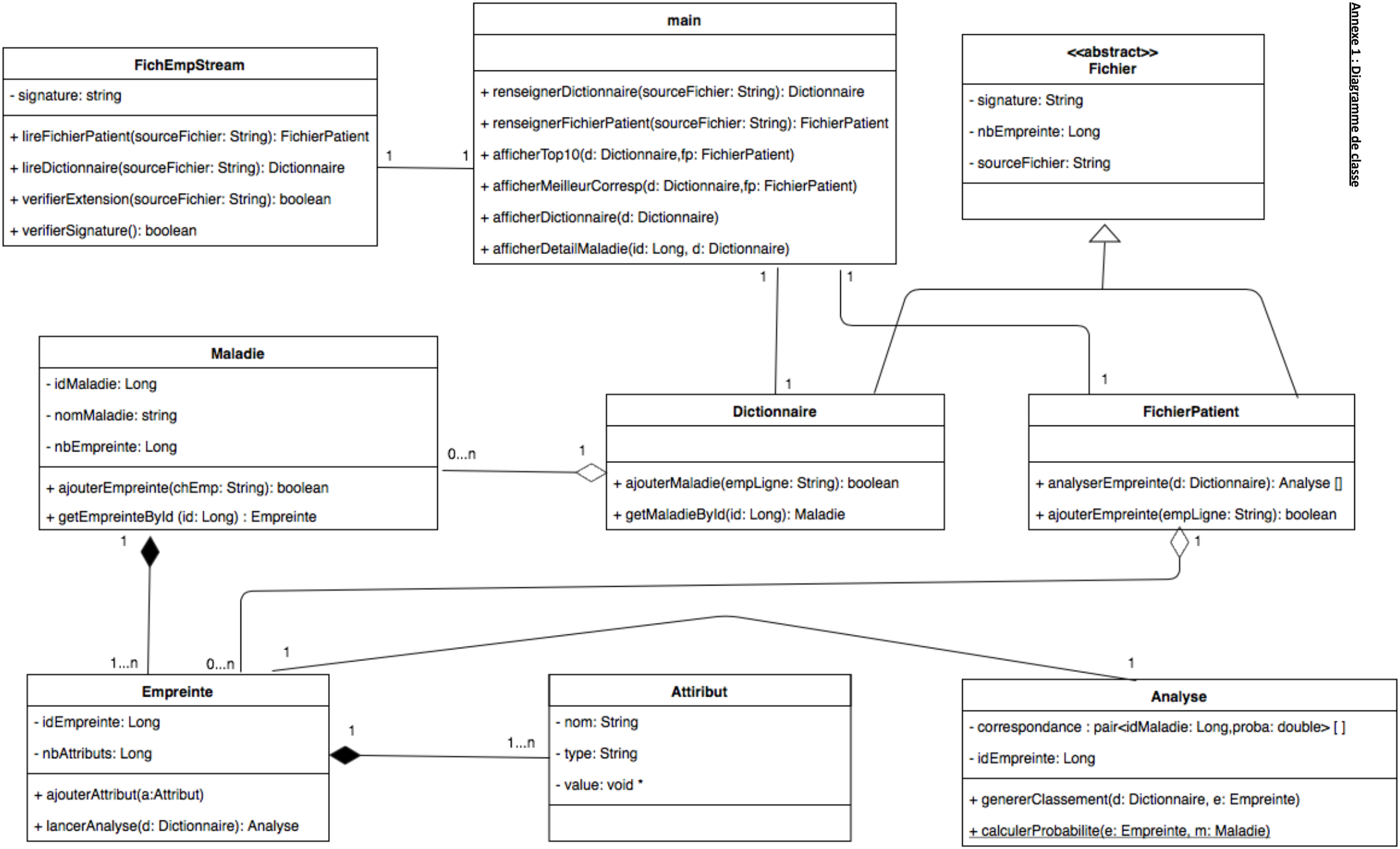
8.1. Générer le classement entre une empreinte et un dictionnaire possédant plusieurs maladie→ Le tableau correspondance possède autant de paires que de maladies dans le dictionnaire

Planning (ébauche)

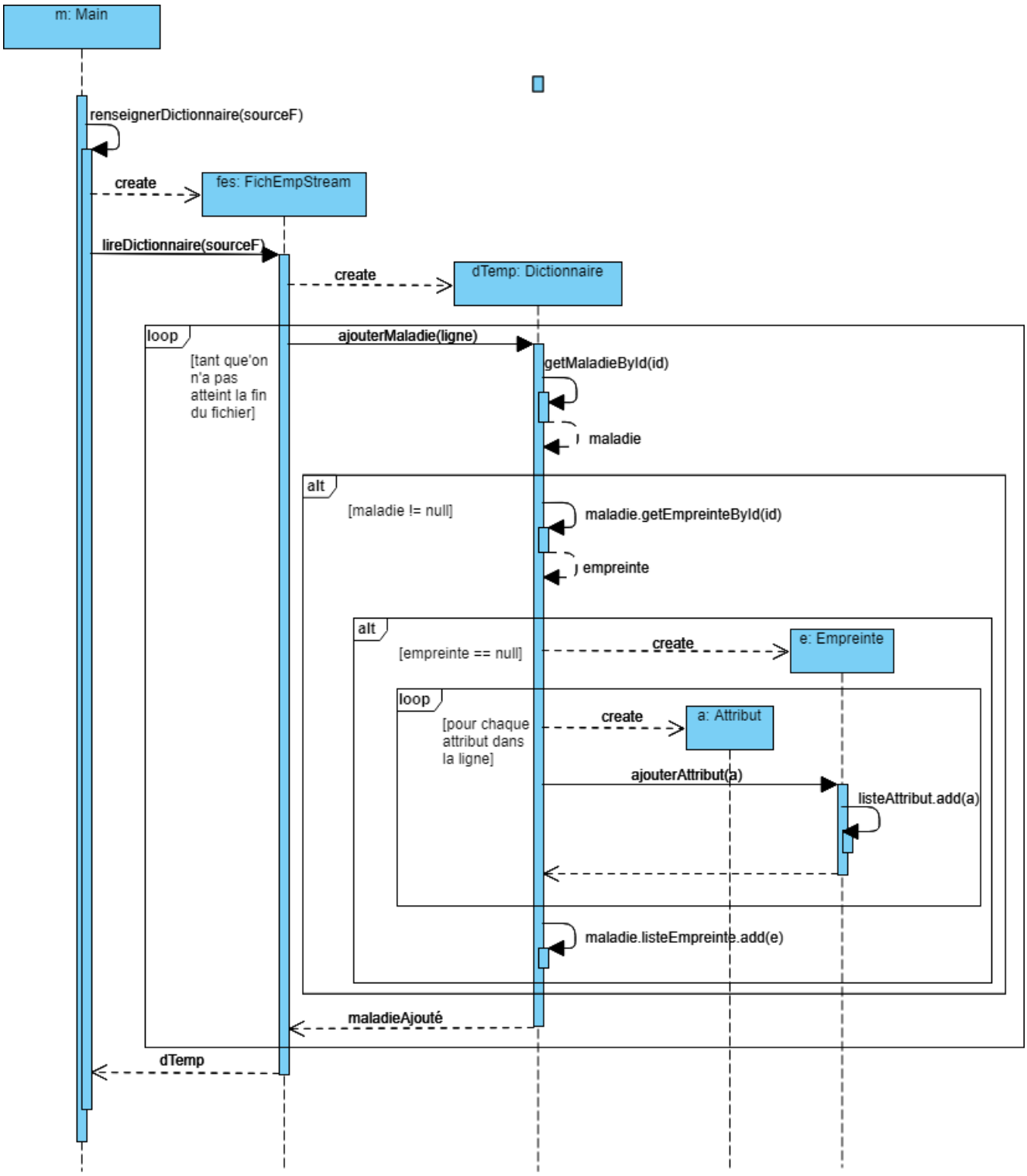
Date	Objectifs
2 mai 2018	Séance de TP numéro 3 Livrable numéro 2 terminé : - Dossier de conception complété - Plan de test d'intégration rédigé - Première version des tests unitaires rédigée - Planning ajusté Découverte de la phase numéro 3 du TP - Début de développement de l'application : programmation des différentes classes + test pour chaque méthode codée. - Ajustement du planning en conséquence des avancées.

2 mai 2018 - 23 mai 2018	Avancement de la phase de développement et réalisation des tests.
23 mai 2018	Séance de TP numéro 4 Dernière modification du code concernant les détails - Réalisation de la totalité des tests - Début de la rédaction du livrable final.
23 mai 2018 - 30 mai 2018	Finition du livrable final.
30 mai 2018	Séance de validation.

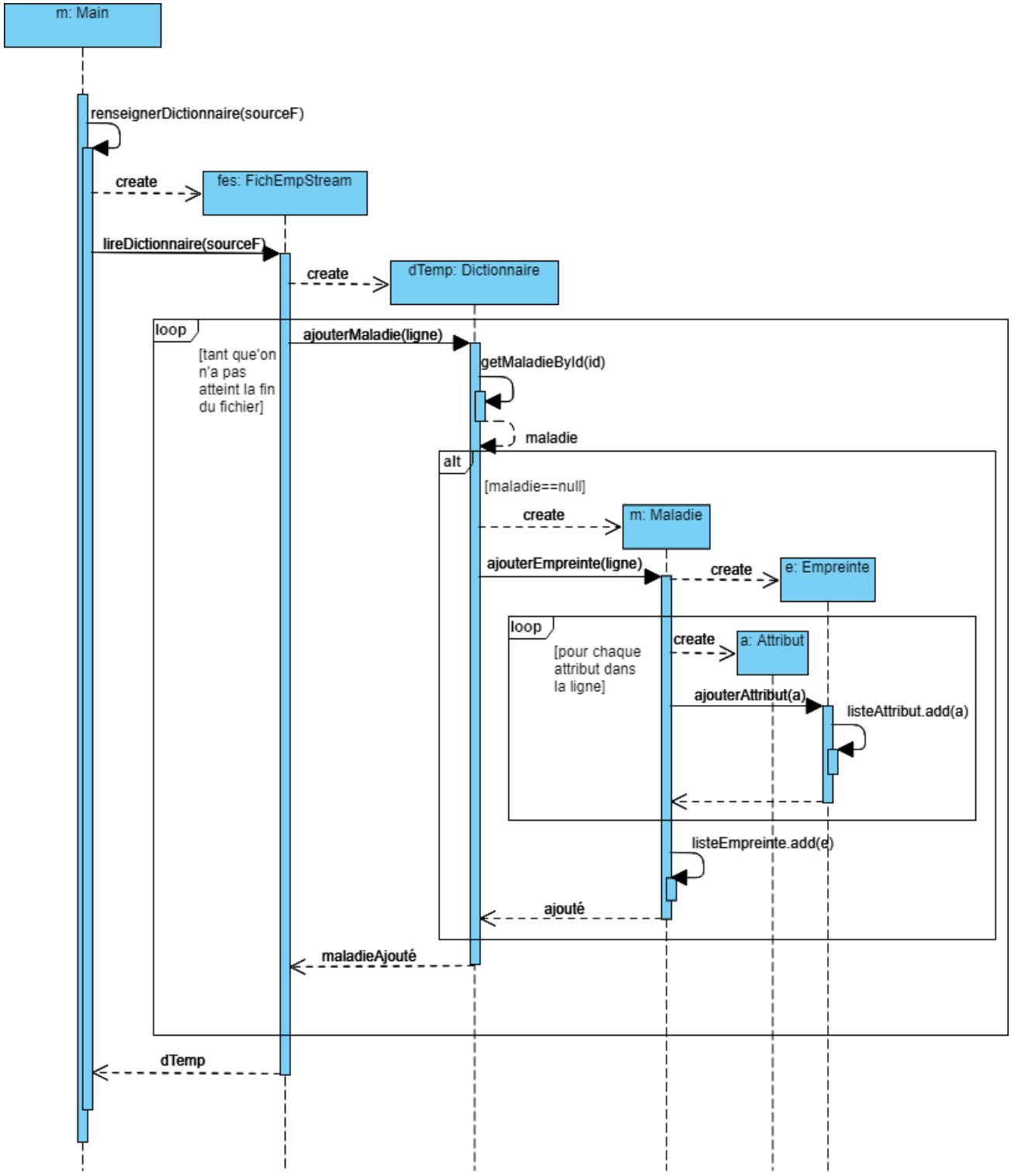
Annexe 1 : Diagramme de classe



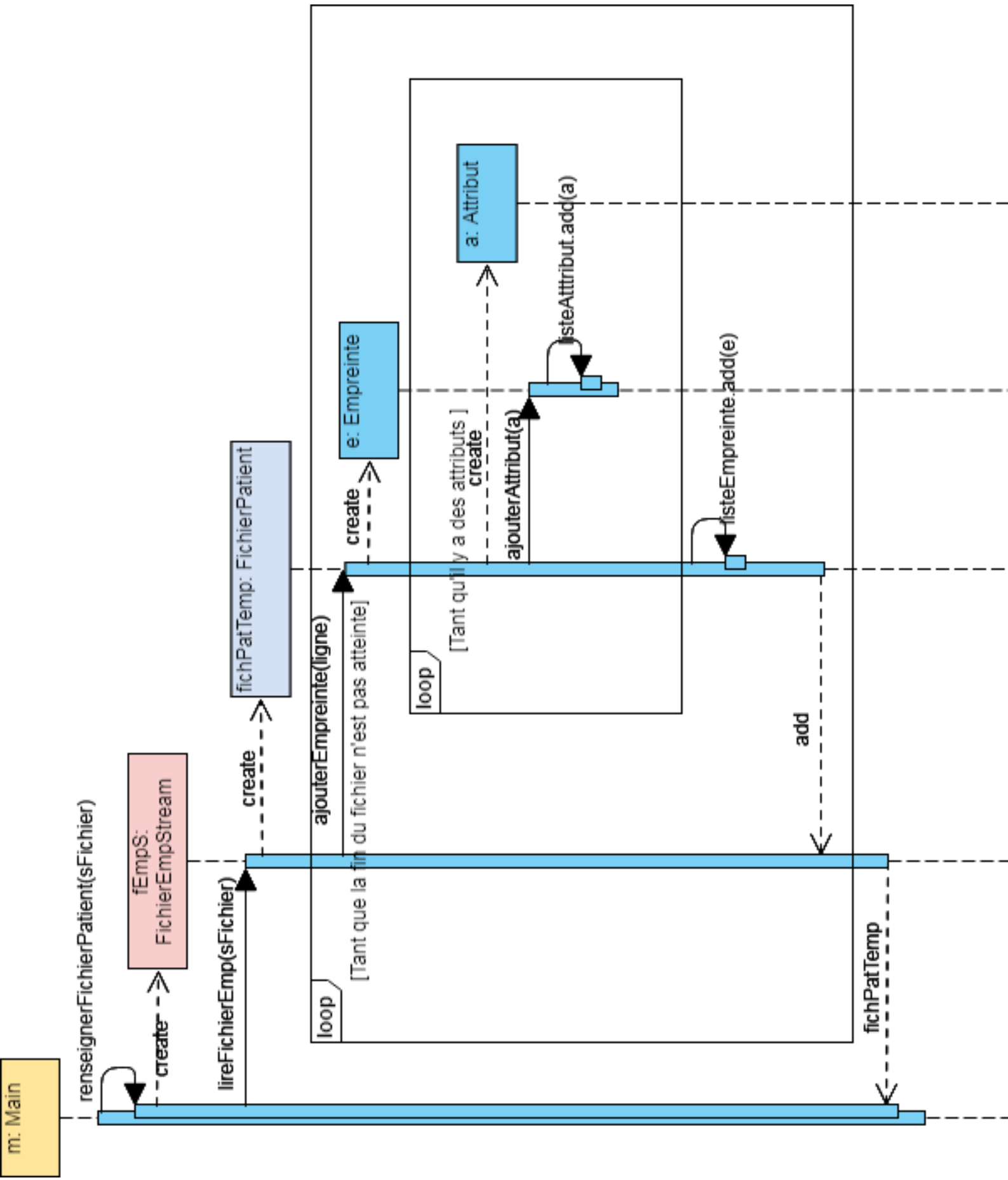
Annexe 2 : Renseigner Dictionnaire (la maladie existe dans le dictionnaire)



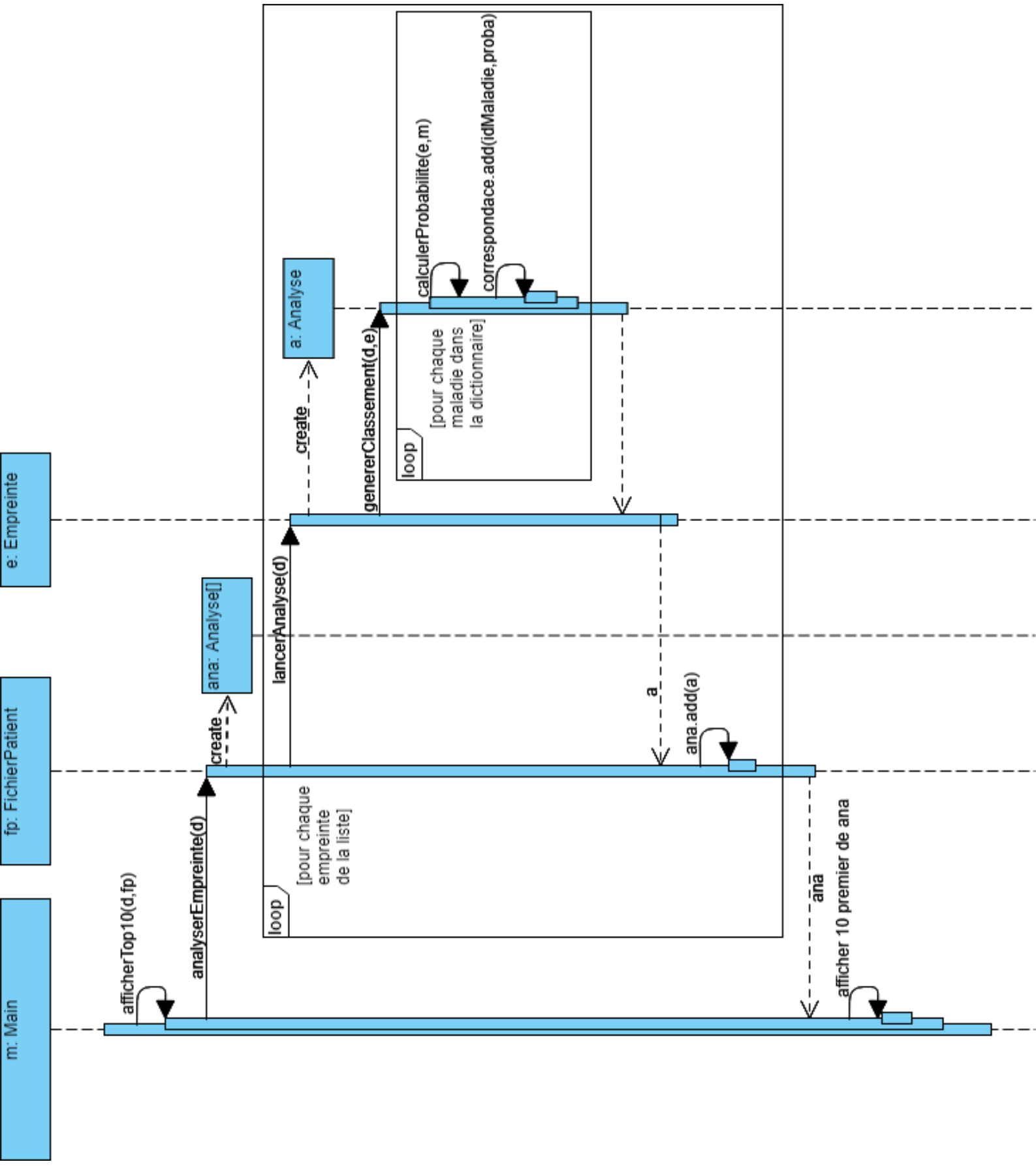
Annexe 3 : Renseigner Dictionnaire (la maladie n'existe pas dans le dictionnaire)



Annexe 4 : Renseigner FichierPatient



Annexe 5 : Afficher Top10



Annexe 6 : Afficher Meilleur Coreespondance