DAUPHINE
UNIVERSITÉ PARIS

Module : Projet Java

Rapport de projet :

Réalisation d'une application java qui permet la correction d'erreurs dans des bases de données médicales

Réalisé par :
- Bahlel Arslane
- Amrouche Karim Moumene

Année universitaire 2018-2019

Sommaire

1. Introduction

Le Projet consiste à développer une application java, permettant la correction d'erreurs dans une base de données d'un hôpital. Nous détaillons dans ce dossier les fonctionnalités développées ainsi que l'architecture de l'application.

2. Phases d'exécution

Au lancement de l'exécution, le logiciel fera la lecture du fichier à corriger afin de récupérer toutes les informations, et va créer les objets nécessaires pour les stocker sous forme de plannings. Ainsi, chaque chirurgien aura une liste de chirurgies qu'il a effectué. C'est également le cas pour chaque bloc et pour tout couple chirurgien-bloc.

En suite, la phase de correction sera lancé et le programme va donc chercher les conflits qui existent, va créer les objets de type «Conflit» et les enregistre dans une liste de conflits qui est un attribut de la classe «Hôpital». Il y a 3 types de conflits possibles :

- Conflit de Bloc :

    Il y a conflit de bloc quand deux chirurgies qui se produisent au même moment se font dans le même bloc.

- Conflit de Chirurgien :

    Il y a conflit de chirurgien quand deux chirurgies qui se produisent au même moment sont faites par le même chirurgien.

- Conflit de chirurgien et bloc :

Il y a conflit de chirurgien et bloc, si un couple : bloc-chirurgien est mobilisé pour deux chirurgies qui se produisent au même moment.

Une fois la liste des conflits crée, si elle est vide, il y a donc aucun problème détecté dans la base de donnée, le programme va donc s'arrêter. Sinon, tant que la liste n'est pas vide, la fonction «solve» va parcourir cette liste et va traiter chaque conflit en fonction du type.

Dans le cas ou le conflit est causé par le chirurgien, la fonction «trouver» va chercher un chirurgien qui n'a pas de chirurgie planifié au même moment que les deux qui posent problème, une fois trouvé il remplacera donc le chirurgien qui a causé de conflit grâce à la méthode «replace», le conflit sera enfin supprimé de la liste. Le travail sera fait dans le cas d'un conflit sur le bloc.

Dans le cas ou le conflit est causé par le chirurgien et le bloc à la fois, la fonction de correction va essayer de trouver un créneau pour lequel le chirurgien et le bloc ne sont pas programmés et va donc modifier le créneau de la chirurgie. Si aucun créneau n'est disponible le même jour, la recherche se fera sur les jours d'après, et un remplacement de créneau sera effectué pour la chirurgie et le conflit va être supprimé de la liste.

Enfin, après que tous les conflits soient corrigés, des statistiques seront affichés, notamment le nombre de conflits détectes le temps de correction moyen.

3. Architecture

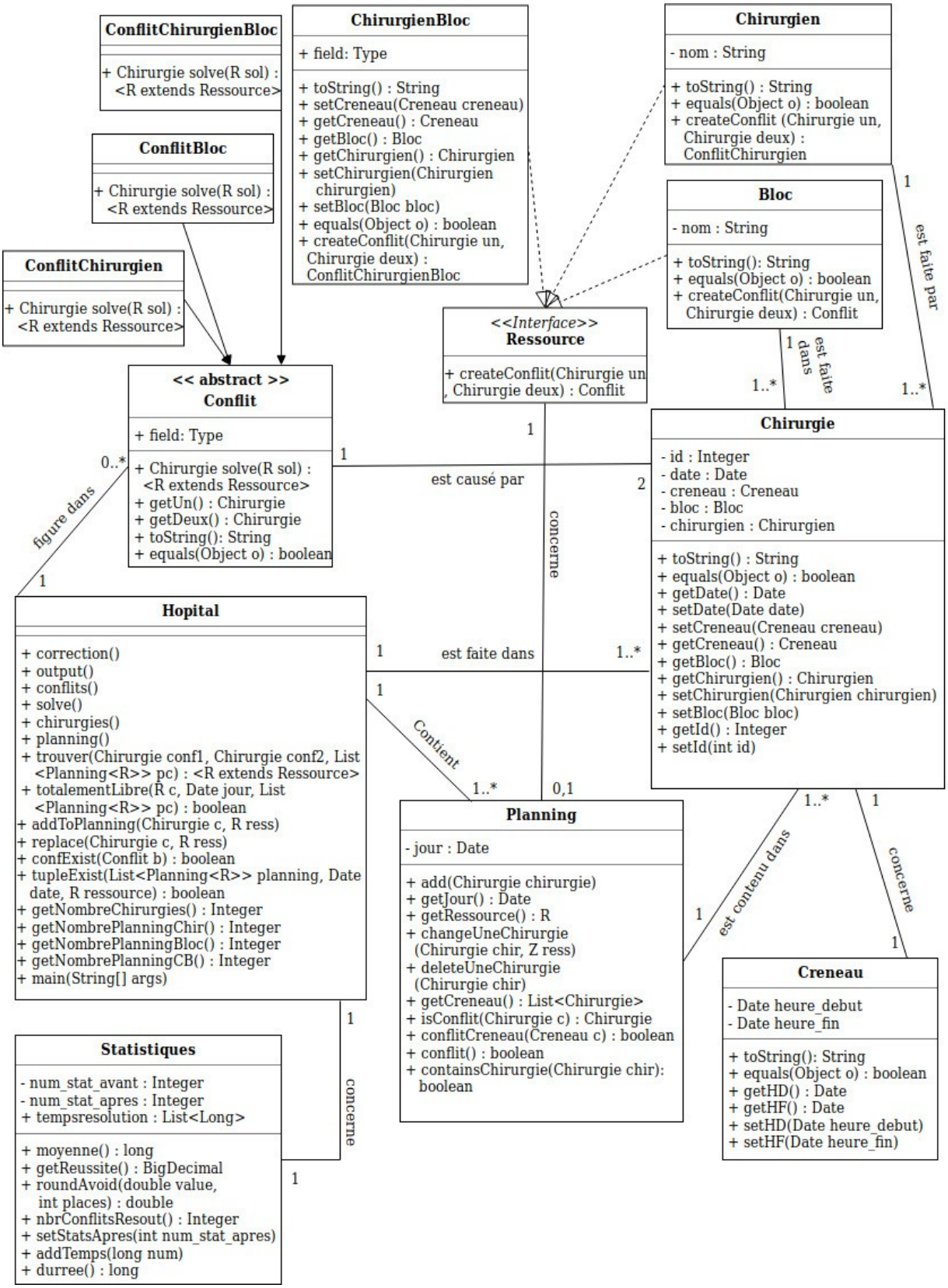

Diagramme de classes de l'application

La classe principale est «Hôpital», elle a pour attributs une liste de chirurgies, plannings de ressources et une liste de conflits. La fonction principale est «correction», elle fait appel au fonctions chirurgies et planning pour récupérer toutes les chirurgies de la base de donnes et créer les plannings pour chaque type de ressource, respectivement. La fonction «createConflit» détecte les erreurs entre chirurgies et crée un conflit pour chaque erreur en fonction du type de la ressource, cette fonction est donc redéfinie dans chaque classe qui implémente l'interface «Ressource».

La fonction «solve» est abstraite dans la classe «Conflit». Elle est donc redéfinie dans chaque sous classe et corrige les conflits de différentes manières en remplaçant la ressource qui a généré le conflit par une autre ressource que cherchera la fonction «trouver».

La classe Planning a pour attributs «jour» qui représente le jour du planning, une ressource «R», et également une liste de chirurgies qui concernent cette ressource pour ce jour là. La fonction booléenne «conflitCreeau» définie dans cette classe permet de savoir si des conflits existent pour ce créneau. La fonction «isConflit» prend en paramètre une chirurgie et cherche dans la liste des chirurgies du créneau si un conflit existe avec une d'entre elles, si c'est le cas, elle renvoie la chirurgie trouvée.

La fonction «conflitCréneau» de la classe «creneau» cherche dans ce dernier parmi toutes ses chirurgies si un conflit existe entre deux d'entre elles. Si il n'existe pas de conflits elle renvoie faux.

4. Difficultés rencontrées :

Nous avons rencontré deux difficultés lors du développement de l’application :

- La première concerne la structure à utiliser pour faciliter la détection et correction de conflits. Sachant que au début nous avions commencé a tester nos fonctions sur la petite base de donnée, nous avions crée uniquement deux sous classes de type conflit, qui sont «ConflitBloc» et «ConflitChirurgien», mais en avançant dans le projet on a crée la troisième qui est «ConflitChirurgienBloc» afin de pouvoir gérer tous les types de conflits existants.

- La deuxième concerne l’optimisation de la résolution des conflits. Nous avions au début négligé le temps que prendrait la phase de création de nouveaux plannings dans la classe «hôpital», ainsi après chaque correction de conflit on généraient de nouveaux plannings, le temps d’attente n’était pas très important sur la petite base de données. Lors du passage à la grande, nous nous sommes retrouvées avec un temps moyen d’exécution du programme de 30 minutes. Nous avons alors repensé notre algorithme et manière de travailler pour finalement changer uniquement les points important lors de la correction d’erreurs, ce qui a considérablement optimisé notre travail et temps d’exécution.