Rapport pour le Projet 1 d'Algorithmique

OOMS Aurélien BA1 INFO

28 février 2011

Table des matières

1 Introduction

Dans le cadre du cours d'algorithmique, il nous a été demandé d'implémenter un type de donnée abstrait permettant de gérer la file d'attente des urgence d'un hôpital. Les sections qui vont suivre vont décrire le fonctionnement des différentes classes et fonctions misent en place à cet effet.

2 Patient

Classe ayant 3 attributs :

1. ticket : ordre du patient dans la file
2. name : nom du patient
3. language : langue(s) parlée(s) par le patient

Name est juste là à titre indicatif alors que ticket et language permettent de choisir quel est le patient suivant en fonction du ou des médecins disponibles. Des méthodes get et set sont implémentées en conséquence.

3 Docteur

Presque la même classe que Patient. On a rajouté un attribut *disposable* qui permet de déterminer si le médecin est en consultation ou non. Des méthodes get et set sont implémentées en conséquence.

4 FileLangues

Classe qui fait office de file FIFO. Elle est composée d'un dictionnaire comprenant tous les objets et de 3 listes d'indices qui rendent la recherche dans le dictionnaire directe. L'attribut nbrPatients permet un affichage immédiat de la taille du dictionnaire. Les sous-sections suivantes décrivent les méthodes propres à la classe.

4.1 getFile

Renvoie le dictionnaire.

4.2 isEmpty

Renvoie True si le dictionnaire est vide.

4.3 insert

Insère le patient dans le dictionnaire et place son indice dans la liste correspondant à sa langue.

4.4 size

Renvoie la taille du dictionnaire.

4.5 removeA, removeB, remove

Renvoie le patient correspondant à la langue le plus anciennement rentré dans la file.

4.6 repr

Affiche uniquement le dictionnaire.

5 Display

Fonction utilisée dans l'interface qui affiche la file d'attente, les médecins disponibles et les patients en consultation.

6 Interface

L'interface génère les FileLangues pour les patients et les médecin ainsi qu'une liste cabinets qui retient les consultations en cours. Elle utilise également deux variables permettant d'attribuer un ordre (tickets) aux patients et aux médecins. Elle propose 4 choix différents sous forme de boucle.

6.1 1, Ajouter un patient dans la file d'attente

Demande le nom et la langue, attribue un ticket et insère le patient.

6.2 2, Indiquer qu'un médecin est arrivé à l'hôpital

Demande le nom et la langue, attribue un ticket et insère le médecin.

6.3 3, Indiquer qu'un médecin est disponible

Termine la consultation en cours d'un médecin, avec pour conséquences mise à jour de son ticket, modification de son attribut de disponibilité et suppression de la consultation dans la liste cabinets.

6.4 4, Fermeture du programme

Choix qui a pour conséquence l'arrêt du programme. Toutes les données sont alors perdues car non enregistrées.

6.5 Attribution des consultations

A la fin des choix 1, 2 et 3 on vérifie si il n'y a pas moyen d'attribuer un médecin à un patient. On fait le teste pour chaque médecin disponible. On recommence ensuite la boucle des choix.

7 Conclusion

Le choix du dictionnaire comme file d'attente permet la non recherche pour les différentes méthodes implémentées car il permet de supprimer des éléments sans pour autant altérer les indices d'accès aux valeurs (clefs) et donc évite tout parcours de la file.

Une autre option aurait été d'utiliser une classe file doublement chaînée ayant pour attributs :

1. head
2. topA
3. topB

Et dont les noeuds auraient la structure suivante :

1. name
2. language
3. next
4. previous
5. nextA
6. nextB
7. previousA
8. previousB

Dans ce travail, l'utilisation des dictionnaires a été privilégiée car déjà maîtrisée et l'utilisation de tickets dans une file d'attente paraît plus intuitif.