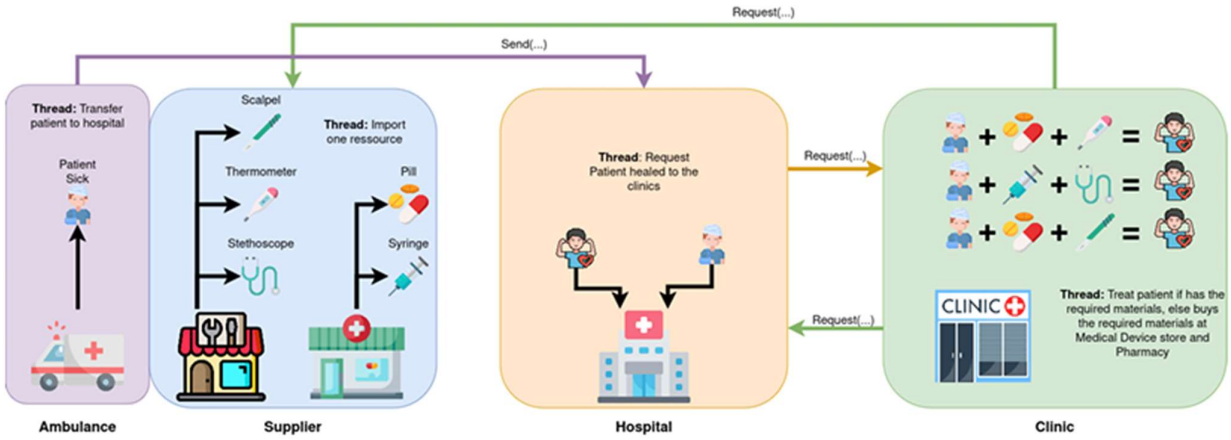
Laboratoire 3

Gestion d'accès concurrents



Introduction au problème

Nous cherchons à représenter un système de santé composé de patients, d'ambulances, d'hôpitaux, de cliniques et de fournisseurs. De la sorte, nous devons faire de la gestion de stock, patients, et argent dans un système qui évolue via plusieurs threads et pose donc des questions de concurrence.

Choix d'implémentation

Pour ce laboratoire, étant donné la donné que laissait de la place à l'interprétation, nous avons fait plusieurs choix importants.

Le plus important est que nous avons modifié la manière dont la libération des patients se produit. Initialement, les patients se font libérer d'un hôpital après 5 jours, et c'est tout. Si on suit cette logique, nous nous retrouvons avec un système qui s'échoue par manque d'argent. Les hôpitaux et les cliniques sont perdants, et donc n'ont pas d'intérêt à participer au système.

De la sorte, nous avons fait le choix de remettre aux hôpitaux de l'argent quand ils libèrent un patient. Cela fait en sorte que les hôpitaux soient gagnants à opérer dans le système.

Nous avons aussi augmenté le « coût » d'un patient guéri, ce qui permet aux cliniques d'avoir un retour positif sur leurs activités, et de ne pas arriver à court d'argent et de bloquer le système.

Cela a l'impact important qu'on ajoute de l'argent dans le système, proportionnel au nombre de gens libérés. Cela sera très visible lorsque les tests seront lancés, mais vous remarquerez qu'on a mis à jour la condition de fin pour la comparaison de l'argent pour représenter notre changement.

Nous avons aussi choisi de centraliser la gestion de l'interface et des mutex dans des classes spécialisées, enfants de Seller.

SellerInterface est un enfant direct de Seller, et implémente juste de quoi effectuer les interactions avec l'interface. Cela permet aussi de ne pas avoir à setInterface pour tous les éléments de notre système un à un.

SellerMutex est un enfant direct de SellerInterface, et permet l'utilisation de mutex. Nous avons un mutex pour l'interface et un mutex pour les ressources. Le seul qu'on ouvre et ferme manuellement dans le code des sous-classes est le mutex des ressources, étant donné qu'on transforme la gestion de l'interface dans cette classe pour prendre directement en compte le mutex de l'interface.

Toutes les classes qui peuvent voir leurs ressources être changées par l'appel de plusieurs threads utilisent SellerMutex, tandis que tous ceux qui ne peuvent être modifiés que par un thread utilisent SellerInterface. La seule classe dans cette dernière catégorie est la classe « Ambulance ».

Nous gérons la création des ressources dans les fournisseurs pour qu'ils essaient de garder un équilibre dans leur stock. Cela est dû au fait qu'on a remarqué que les cliniques ne demandent pas les ressources de manière uniforme, donc les créer de manière aléatoire ou uniforme ne serait pas optimisé pour le fournisseur.

Tests effectués

Pour ce laboratoire, nous nous sommes constamment servi des output en interface ou en terminal pour suivre et corriger les éventuelles erreurs que nous rencontrions. De la sorte, nous avons généralement été capables d'isoler les méthodes problématiques en cas de problème et de nous concentrer uniquement sur elles jusqu'à régler les erreurs.

Nous avons aussi procédé à des changements des constantes pour s'assurer du bon fonctionnement dans des cas plus limites, où notre programme allait se terminer pour cause de manque d'argent dans un des vendeurs. Cela nous a permis de nous rendre compte des limitations du système et de comment régler les constantes pour faire en sorte d'arriver à bout des patients de l'ambulance. Pour ainsi dire, nous avons fait un contrôle des prix dans un système « capitaliste » où tous trouvaient donc un bénéfice à exercer leurs activités.

Nous nous sommes aussi plongé dans des discussions intéressantes sur la gestion des ressources et la position des mutex dans notre code entre nous et avec l'assistant, nous permettant de la sorte de corriger notre structure et d'arriver à un résultat qui nous satisfait, et qui, nous espérons, vous satisfera également.

Merci de votre lecture !