Laboratoire 03 : Gestion d'accès concurrents

Auteurs : Léon Surbeck, Alex Berberat

Introduction au problème

Le but de ce laboratoire est de simuler la gestion d'une crise sanitaire dans une petite ville, en mettant en avant les défis liés aux accès concurrents.
Quatre acteurs principaux doivent interagir efficacement : les ambulances, les fournisseurs de ressources médicales, les hôpitaux et les cliniques spécialisées.
Chaque acteur gère des ressources limitées, et notre tâche est de réguler les flux tout en protégeant les sections critiques à l'aide de mutex.

La plupart des méthodes suivantes comportent des sections critiques, qui ont été protégées par des mutex pour éviter les conditions de concurrence.

(Presque) Toutes les classes

- La méthode `run` a été mise à jour pour s'arrêter dès qu'un `requestStop` est appelé pour ce thread.
- La méthode `request()` gère les transactions d'achat de ressources en vérifiant la disponibilité des stocks et en mettant à jour les fonds.

Ambulances

- la méthode `sendPatient` a été implémentée pour choisir au hasard un des hôpitaux et y envoyer (si place disponible et leur argent le permet) 1 patient.

Hôpitaux

- la méthode `send` a été implémentée pour vérifier si l'hôpital possède assez de fonds pour payer les frais d'un nouveau patient et vérifier si assez de lits sont disponibles.
- la méthode `freeHealedPatient` a été implémentée pour libérer un lit et mettre à jour les fonds de l'hôpital une fois qu'ils y ont passé 5 jours (cycles).
- la méthode `transferPatientsFromClinic` a été implémentée pour transférer les patients de la clinique à l'hôpital si les conditions le permettent.

Cliniques

- la méthode `sendPatient` a été implémentée pour vérifier si l'hôpital possède assez de fonds pour payer les frais d'un nouveau patient et vérifier si assez de lits sont disponibles.
- la méthode `treatPatient` a été implémentée pour traiter un patient et mettre à jour les fonds de la clinique si les conditions le permettent.
- la méthode `orderResources` a été implémentée pour commander des ressources (matériel / patients malades) si les conditions le permettent.

Fournisseurs de ressources

- la méthode `run` achète les ressources de manière aléatoire.

Tests effectués

Nous avons testé les différentes classes en les faisant interagir entre elles pour vérifier que les transactions d'achat et de traitement sont correctement gérées grâce à des tests unitaires.
Ils ont été mis en place pour chaque acteur, en utilisant `gtest`, pour s'assurer que les transactions d'achat et de traitement sont correctement gérées.