Titre du laboratoire

Auteurs: Camille Koestli, Victor Nicolet

Description des fonctionnalités du logiciel

Ce laboratoire a pour but de simuler le fonctionnement d'un réseau ferroviaire à travers la gestion de ressources partagées. Ces ressources comprennent le tronçon commun, qui est emprunté par une seule locomotive à la fois et une gare où les locomotives doivent s'attendre à chaque tour.

La gestion d'entrée dans le tronçon commun est accordé à la locomotive qui a une priorité supérieur à l'autre. Si les deux locomotives ont la même priorité, alors elle sera accordée à la première arrivée sur le tronçon commun.

 parcours-choisi

Choix d'implémentation

Gestion de l'attente en gare

La classe `SharedStation` gère l'attente des locomotives dans une gare. Elle régule l'entrée et sortie en fonction du nombre maximum de locomotives autorisées à attendre.

Le sémaphore `semaphoreEntreeStation` contrôle le nombre de locomotives pouvant entrer dans la station. La sémaphore `semaphoreAttenteStation` gère le nombre maximum de locomotives pouvant attendre dans la gare. Lorsqu'une locomotive atteint le seuil des tours prédéfinis, elle appelle `waitingAtStation()`, ce qui met en pause son thread jusqu'à ce qu'elle soit autorisée à repartir. La variable `locosEnAttente` indique si des locomotives sont actuellement en attente. Cela pourrait être utile pour une gestion avancée des priorités.

Cela illustre un usage efficace des sémaphores pour synchroniser plusieurs threads sur une ressource partagée.

Gestion de l'accès au tronçon critique

La classe `SharedSection` représente un tronçon partagé du réseau ferroviaire. Ce tronçon est une ressource critique qui ne peut être utilisée que par une locomotive à la fois. L'objectif principal de cette classe est de garantir l'exclusion mutuelle dans l'accès à cette section.

- Accès à la section : Lorsque la méthode `access` est appelée par une locomotive, elle vérifie si la section est occupée. Si oui, la locomotive s'arrête, et son thread est mis en attente jusqu'à ce que la section soit libérée. Cette attente est gérée par un sémaphore (`semaphoreSection`) qui permet de bloquer les threads en attente.
- Sortie de la section : Une fois que la locomotive quitte la section, elle appelle `leave`, qui libère le sémaphore et réveille les threads en attente. Un verrou (`mutexSection`) protège l'accès aux indicateurs d'état internes (`estOccupee`).

Ce mécanisme évite les conditions de course (data races) et garantit une synchronisation correcte.

Gestion de la priorité

Les priorités sont attribuées aléatoirement grâce à la méthode `randPriority()`. Cela permet de simuler une variation dynamique des priorités entre les locomotives. La classe `SharedSection` implémente également une méthode `togglePriorityMode()` pour basculer entre deux modes de gestion :

- HIGH_PRIORITY : Les locomotives avec les priorités les plus élevées sont servies en premier.
- LOW_PRIORITY : Les locomotives avec les priorités les plus basses sont servies en premier.

Le mode de priorité est basculé à intervalles réguliers après un nombre défini de tours.

Comportement des locomotives

La classe `LocomotiveBehavior` définit les actions de chaque locomotive. Elle repose sur deux méthodes principales :

`moveForward()` : Déplace la locomotive dans la direction avant, en respectant les points de contact et en configurant les aiguillages. `moveBackward()` : Déplace la locomotive dans la direction arrière.

Chaque méthode suit ces étapes :

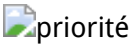
1. Attente au point de contact d'entrée.
2. Demande d'accès au tronçon critique (`request()`).
3. Accès au tronçon critique une fois les conditions remplies (`access()`).
4. Configuration des aiguillages si nécessaire.
5. Quitte le tronçon et notifie les autres locomotives (`leave()`).
6. Attente à la gare après un certain nombre de tours (`waitingAtStation()`).

Tests effectués

Nous avons exécuté notre programme en gardant l'inertie et laissé quelques tours pour vérifier que toutes les contraintes étaient fonctionnelles.

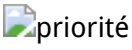
Test 1 : Priorité plus élevée pour la locomotive 42

D'abord on voit que la priorité et l'attente sont respecté, la loco 42 prend la priorité et donc la loco 7 attend bien avant l'entrée de la section critique et part dès que la loco 42 a quitté la section critique.

 priorité

Test 2 : Priorités plus élevée pour la locomotive 42 et attente de la loco 7

Nous voyons ici que la loco 42 arrivant après, est bien laissée prioritaire pour la section critique.

 priorité

Test 3 : Aiguillages dynamiques

On voit également que les aiguillages 13 et 10 ont bien été switchés pour cette loco.  aiguillage

Test 4 : Réinitialisation des aiguillages

Ils sont bien remis en place quand la loco 7 est hors de la section critique. 

Test 5 : Gestion de l'attente en gare

On voit ici que la loco 42 attend bien en gare après 2 tours.