PCO - Laboratoire 5 "Gestion de ressources"

Auteurs: Gabriel Roch, Cassandre Wojciechowski

Description des fonctionnalités du logiciel

Dans le cadre de ce laboratoire, nous avons réalisé deux programmes.
Le premier permet à deux locomotives de partir depuis leur point de départ et de parcourir un certain tracé. Nous avons nous-même défini ces tracés depuis le "main" du programme. Ces deux locomotives effectuent deux fois le parcours demandé, puis elles font demi-tour et refont le parcours à l'envers et cela d'une manière infinie. Sur le chemin, elles rencontrent des sections dites partagées : elles peuvent passer dessus toutes les deux. Nous avons donc mis en place un système de demande d'accès pour qu'une seule locomotive se trouve sur la section partagée à la fois. Quand une locomotive sort de cette section, l'autre peut y accéder à son tour. Si la section est occupée, la locomotive arrivée après doit s'arrêter et attendre.

Le second programme se comporte de manière similaire au premier mais les locomotives ont chacune un niveau de priorité différent. Cela change l'accès aux sections partagée car ce n'est plus la première arrivée qui passe d'abord mais celle qui a un niveau de priorité plus haut qui est favorisée.

Choix d'implémentation

Au niveau de notre implémentation, nous avons ajouté des classes supplémentaires à celles déjà présentes dans le projet, notamment pour gérer les parcours effectués par les locomotives.

Dans la fonction LocomotiveBehavior::run(), nous avons implémenté le fonctionnement général d'une locomotive. La direction indiquée permet de savoir de quel côté du parcours nous commençons et dans quel sens nous allons l'effectuer. Au fur et à mesure du parcours, nous allons diriger les aiguillages pour aller aux endroits demandés et la locomotive va demander l'accès à chaque section partagée, puis, si elle le peut, elle l'obtiendra.
La locomotive avec la priorité la plus haute sera évidemment la première à y accéder, l'autre devant s'arrêter et attendre que la section soit libre.
Si la section partagée est déjà occupée, le thread de la locomotive effectuant l'appel sera suspendu grâce à la fonction "getAccess" et ne sera réveillé qu'au moment où l'autre locomotive sera effectivement sortie de la section.
Nous faisons appel à la fonction "request" deux sections à l'avance et à "getAccess" une section à l'avance car la fonction "getAccess" arrête effectivement la locomotive, il ne faut pas que la locomotive demande l'accès dans le cas où elle doit s'arrêter pour la section précédente.
Après que le contact ait été fait (c'est-à-dire que la locomotive soit passée sur la section), nous faisons appel à la fonction "leave" qui va permettre au thread suspendu de la locomotive en attente de se réveiller.

Quand le parcours a été effectué deux fois, nous allons arrêter la locomotive, lui faire faire demi-tour et elle va repartir pour parcourir le même chemin mais dans le sens inverse.
Pour cela, nous avons créé une classe nommée "ParcoursIterator" afin d'éviter d'implémenter deux boucles "for" faisant presque exactement la même chose à l'exception du sens du parcours.

Tests effectués

Les tests suivants ont tous été exécutés avec succès (il n'y a jamais eu de crash ou de blocage définitif des locomotives) :

- Si une loco est déjà présente sur la section partagée (peu importe la priorité car elle est déjà engagée), l'autre doit attendre son passage -> OK **avec** et **sans** inertie
- Mettre en pause une loco, puis la redémarrer pour que les deux locos arrivent en sens opposé sur la section partagée -> OK **avec** et **sans** inertie
- Mettre en pause une loco, pour que la loco à faible priorité demande l'accès, mais se fait arrêter car l'autre (haute priorité) a déjà fait un `request()` -> OK **avec** et **sans** inertie

En augmentant la vitesse au-dessus de 14, nous notons que les locomotives ont tendance à dérailler. Nous attribuons cela au fait que la simulation aurait du se faire sur une maquette et le professeur a annoncé que la vitesse maximale de la maquette est 14.
En augmentant la vitesse au-delà de 13 et en activant l'inertie, il arrive parfois des collisions. Cela est dû à l'inertie et à la vitesse élevée, même si les requêtes sur les sections partagées sont faites deux sections à l'avance, il arrive que la distance ne soit pas suffisante pour s'arrêter.