Simulation d'une gare

Le TP est composé de deux parties. La première partie peut fonctionner sans la seconde. Lorsque vous commencerez la deuxième partie, faites une copie de la première.

Partie 1 : Simulation multithreadée

Vous allez simuler le fonctionnement d'une gare. Les comportements à simuler ne sont pas toujours réalistes mais permettent de simplifier la modélisation afin de résoudre différents problèmes de synchronisation. Le programme principal `Gare` devra contenir les éléments suivants :

- Un `EspaceQuai`, composé d'un nombre fixe de voies, chaque voie pouvant accueillir un seul train à la fois. C'est dans cet espace que les clients chercheront à monter dans un train. La gare ne comporte qu'un seul `EspaceQuai`.
- Un `EspaceVente`, dans lequel on peut acheter un billet de train. Les billets ne sont pas pour un train en particulier, mais peuvent permettre de monter dans n'importe quel train. Le nombre de billets est limité. L'impression du ticket peut prendre un certain temps.
- Des `Trains`, qui arrivent en gare (sur une voie libre de l'`EspaceQuai`), restent un temps limité en gare pour laisser monter les voyageurs, et quittent la gare. Plus précisément, un train : 1) arrive en gare en `10000 / VITESSE_TRAIN` (ms) où `VITESSE_TRAIN` est compris entre 50 et 300 (km/h), 2) cherche à se garer sur une voie libre, 3) s'arrête un temps `ARRET_TRAIN`, permettant à des voyageurs munis de billet de monter, 4) repart. Un train qui part peut être supprimé de la simulation. En arrivant en gare, le train a un nombre de places libres qui sera fixé avant son arrivée en gare, et qui sera compris entre 0 et `CAPACITE_MAX`.
- Des `voyageurs`, qui doivent dans l'ordre : 1) acheter un billet s'il en reste et 2) monter dans un train dès que possible. Pour cela, il pourra passer les trains de l'EspaceQuai en revue à la recherche d'une place libre ou attendre un temps et réessayer s'il n'y a aucune place libre dans les trains actuellement à quai.

Partie 2 : API Rest de contrôle

Une fois que votre simulation d'une gare vous semble satisfaisante, vous devrez développer une API Rest permettant de la contrôler. Plus précisément, en utilisant la librairie Restlet, vous allez devoir concevoir, développer et tester une API ayant les fonctionnalités suivantes :

- de lister les trains (GET) et de rajouter un train à votre simulation (POST)
- de lister les voyageurs (GET) et de rajouter un voyageur à votre simulation (POST)

Notez qu'afin de mieux suivre la simulation, les trains et les voyageurs devront être listés avec leur état. Un train aura au cours de la simulation plusieurs états :

1. A - en route vers la gare
2. B - en attente d'une voie libre
3. C - en gare

4. D - parti

De même, un voyageur prendra au cours de la simulation les états suivants :

1. A - en route vers la gare
2. B - muni d'un ticket
3. C - monté dans un train

Instructions

Le TP doit être envoyé au plus tard le 9 décembre 2022 adressé par mail à à `cedric.tedeschi@irisa.fr` ou `damien.hardy@irisa.fr` suivant votre groupe. L'objet de votre mail sera [TP-CSR Gr<x>] <Nom1> <Nom2> [1]. Ce courrier devra contenir :

1. Les sources Java commentées des deux étapes, dans des répertoires différents
2. Un rapport au format PDF contenant :
   (a) la décomposition en classes de votre modélisation, avec l'identification des threads et des objets partagés ;
   (b) la liste des problèmes de synchronisation rencontrés ainsi que leur résolution ;
   (c) des traces explicites montrant vos programmes en fonctionnement ;
   (d) toutes les informations que vous jugerez nécessaires à l'exécution et à la compréhension de votre modélisation et de votre code.

1. Merci de respecter ce format, permettant de retrouver votre travail dans le flot de mails.