P2 of SE **Master 1 Informatique**

Simulating the Traffic in Train Stations

Introduction

We are going to make the simulation of the main activities in a train station, with its flux of passengers and trains. In order to make our job easier, we are going to make, especially in our initial steps, some assumptions which simplify our model. Then, step by step, we'll make our traffic simulations more and more realistic.

The train station can be modeled by using the following Java objects.

The platform

It's the area where trains can stop and passengers can step out and in. We will suppose that the platform can physically accept no more than `NB_TRAINS`.

The trains

They can stop at the station if there is a free place at the platform. Once stopped, the train is allowed to wait a predetermined time, so that all passengers can step out, and all new passengers (with ticket) can step in. The number of passengers that every train is able to transport is bounded by `NB_PASSENGERS`.

The passengers

Every passenger makes the following three actions: the passenger buys a ticket, he waits for his train to arrive, he steps in the train.

Timing

Time is a very important factor in a train station. We will suppose that a predefined time is associated to every action. For example, printing a ticket may take 1 second; a passenger may need in average 2 minutes to find his train; a train may take a few hours to reach the train station. Of course, in order to accelerate the simulation of our train station, we can consider scaling the time from seconds to milliseconds.

Model 1

In this initial model, we make the following assumptions:

- All trains are headed towards the same destination; and all passengers want to go there.
- There is only one ticket office, and the number of tickets per train that can be sold is unlimited.

Model 2

In order to make our simulation more realistic, let's include the following feature:

- When the train arrives to the train station, a subset of passengers leaves the train. As soon as the last passenger leaves, the train can send to the ticket office the number of seats that are currently free onboard. There can be different scenarios, depending on the number of tickets that were already sold before the train arrival. If the number of sold tickets is smaller than the number of free seats, then sale can continue until the limit of free seats is reached. If the number of sold tickets is already equal or greater than the number of free seats, than sale is closed, and only the first passengers finding a place on the train will have the chance to travel; all other passengers will have to wait for the next train.

Model 3

Let's suppose now that our simulation includes 3 stations, say stations A, B and C. There are two railways connecting A and B, B and C, and C and A. As a consequence, our trains can travel from a given station to any of the other two. Passengers can therefore express a wish of destination when buying the ticket. We suppose that only one ticket office is placed in the station A, while the train stations B and C both have two ticket offices. The number of trains that the platforms can take is different for the three stations (you can choose the values, in a reasonable way). All trains have the same bound `NB_PASSENGERS`.

In the simulation, it is very important now to pay attention to two main facts:

- A train can go over a railway to travel from A to B, or from B to A. If a railway is already busy, a departing train can only take the same railway in the same direction of the other trains. When a new train enters in the railway it is therefore constrained to adapt its speed to the other trains on the same railway.
- When tickets are bought, seats on different trains may be booked. Tickets are bought in a concurrent fashion: different ticket offices have access to a common database where the information about the bookings is stored. The access to the database should be controlled in order to avoid reservation problems!

The final report

Together with your Java files, you will have to write a short report about your experience with this project. This report will include a short description of your Java classes, as well as a detailed explanation of the synchronization issues you had to face, with the solutions you have implemented.

Acknowledgments

This is an adaptation of a previous TP that was proposed in the year 2012-13 by Cédric Tedeschi.