IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TP8-10 – Du modèle au code : *Roller coaster*

Fondements théoriques du développement des logiciels concurrents

Objectifs

L'objectif de ces trois TP est de produire des modèles un peu plus complexes et mieux comprendre le lien entre un modèle et le programme Java correspondant.

1 Le cadre de travail

La commune de Plouzané souhaite se munir d'un « grand huit » pour la saison touristique. Le commercial en charge de la vente des manèges propose un système de contrôle extensible en terme de nombre de wagons et de nombre de places à bord des wagons. Le logiciel assurant le contrôle est sous licence GPL et peut donc être modifié à volonté. Le commercial garantit qu'il n'y a jamais eu de problème sur la première et unique version de ce manège qui a été vendue sans facture à l'appui. La commune ne souhaite pourtant pas rester sur cette intime conviction du vendeur.

Dans son cahier des charges, elle souhaite mettre en œuvre la version du manège avec deux wagons. Le premier wagon peut accueillir deux passagers, le second trois passagers. Les deux wagons sont indépendants sur le circuit, c'est-à-dire, ils ne sont pas attachés. En revanche, ils ne peuvent pas se doubler. Les passagers sont en attente d'un wagon sur une plate-forme qui peut accueillir neuf personnes au maximum. Lorsqu'un wagon arrive sur la plate-forme, il attend qu'il y ait suffisamment de passagers présents pour les charger et partir (c'est-à-dire deux ou trois passagers suivant le wagon). Un seul des deux wagons peut être en attente sur la plate-forme d'accès à un instant donné (il existe une unique rampe d'accès pour les passagers).

2 Les exercices

Il s'agit de construire une version simplifiée du modèle du système présenté ci-dessus. Notre système sera composé de trois processus :

1. Le premier `CAR` modélise le comportement d'une voiture.
2. Le deuxième `PLATFORM` modélise le comportement de la plate-forme.

3. Le troisième `PASSENGERS` modélise le comportement des clients souhaitant accéder à la plate-forme.

Exercice 1 (*Un seul wagon à un passager*)

Dans ce premier exercice, nous commençons par travailler avec un seul wagon qui ne peut accueillir qu'un seul passager. On suppose qu'une voiture peut effectuer les actions suivantes :

- `arrive` pour modéliser l'arrivée de la voiture sur la plate-forme de chargement des passagers,
- `requestPassenger` pour indiquer au système que la voiture est prête à embarquer des passagers,
- `getPassenger` pour permettre la montée à bord du passager,
- `depart` pour indiquer le départ de la voiture de la plate-forme.

▷ **Question 1.1 :**
Proposer un modèle de la voiture `CAR` en FSP.

La plate-forme peut contenir un nombre maximal de neuf passagers. Son contrôleur peut réaliser les actions `newPassenger` lorsqu'un nouveau passager candidat se présente, `requestPassenger` et `getPassenger` pour contrôler l'accès des passagers aux voitures. Initialement, on supposera que la plate-forme est vide.

▷ **Question 1.2 :**
Proposer un modèle de la plate-forme `PLATFORM` et de `PASSENGERS` en FSP.

▷ **Question 1.3 :**
Proposer un modèle complet du système.

▷ **Question 1.4 :**
Proposer une première version simple du code Java du système.

Pour faciliter la visualisation de l'application, nous vous proposons de réutiliser un code fourni qui permet de suivre l'affichage de valeur et l'évolution de *thread* en les animant. Un code exemple est sur moodle. Son rendu graphique est en figure 1(a). La partie supérieure affiche une valeur entière (ici la valeur de l'attribut `val` de la classe `Exemple`), la partie centrale affiche la progression du *thread* sous la forme d'un secteur angulaire qui progresse (ici en violet) et en partie inférieure, un bouton pour démarrer et un pour suspendre le *thread*.

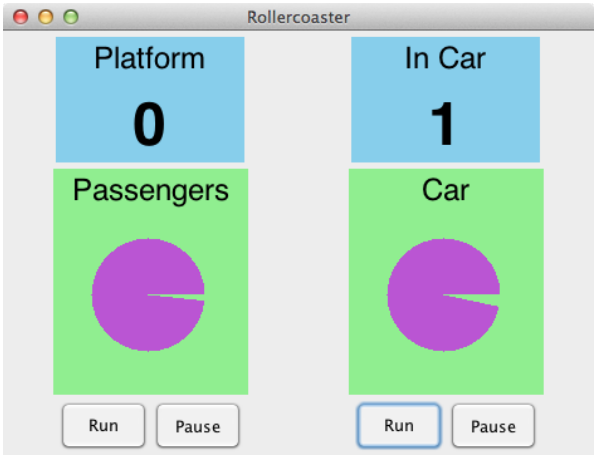
▷ **Question 1.5 :**
En vous inspirant de l'exemple fourni, proposer une version de votre code Java du système de façon à ce qu'il suive l'affichage de la figure 1(b).

Exercice 2 (*avec plusieurs voitures*)

▷ **Question 2.1 :**
Modifier le modèle de l'exercice précédent pour qu'il supporte les deux voitures en même temps.

Un de vos camarades a un modèle dans lequel il y a deux voitures `a` et `b`. Dans son modèle, la trace suivante est accessible :

(a) Un exemple d'affichage de *thread*



(b) L'affichage voulu pour l'exercice 1

`a.arrive`, `a.requestPassenger`, `newPassenger`, `a.getPassenger`, `newPassenger`, `b.arrive`, `b.requestPassenger`, `b.getPassenger`, `b.depart`, `a.depart` ...

▷ **Question 2.2 :**
Cette trace vous semble-t-elle acceptable ? Pourquoi ?

Exercice 3 (*version finale de la spécification*)

▷ **Question 3.1 :**
Généraliser votre solution précédente pour fournir un modèle (correct [1]) répondant au problème décrit dans la première section (c'est-à-dire, les voitures peuvent embarquer plus d'un passager).

▷ **Question 3.2 :**
Modifier votre code Java pour qu'il réalise votre nouveau modèle, l'exécuter et vérifier si vous obtenez des comportements non souhaitables.

Exercice 4

L'objectif de cette question est d'éliminer le problème décrit par la trace de la question 2.2.

Une solution possible consiste à obliger la trace à ne contenir que des suites de la forme `arrive` puis `depart` puis `arrive` puis `depart` ...

▷ **Modifier votre modèle et vérifier à l'aide de l'animateur que les wagons ne peuvent plus être en même temps au niveau de la plate-forme. Modifier ensuite le code pour mettre en œuvre cette solution.**

1. Sans *deadlock* par exemple.

3 Pour aller plus loin

Exercice 5 (*Optimisation du temps d'attente des passagers*)

Pour le moment, nos wagons ne partent que s'ils sont pleins. Afin d'éviter une longue attente inutile aux passagers, on permet qu'un wagon parte même s'il n'est pas plein.

▷ **Modifier le modèle et le code pour satisfaire à cette nouvelle exigence.**

Exercice 6 (*Remplissage des wagons au fur et à mesure de l'arrivée de passagers*)

On abandonne l'idée précédente et on choisit de remplir totalement les wagons en permettant aux passagers de s'installer dans le wagon pendant que celui-ci est immobilisé dans l'attente de passagers.

▷ **Modifier le modèle et le code pour satisfaire à cette nouvelle exigence.**