CSR - TP5

24 novembre 2016

1 Partie A : Simulation parallèle

Lors de ce projet, vous allez simuler le fonctionnement d'un supermarché, de ses clients et de ses deux employés. La supérette dispose de quatre produits en stock, et les clients disposent d'une liste de courses indiquant la quantité souhaitée de chacun de ces quatre produits.

La figure 1 présente un schéma de la supérette. La supérette est constituée de :

1. **un entrepôt** qui contient les stocks des différents produits avant leur mise en rayon. On pourra considérer que ces stocks sont illimités ; Il ne sera pas forcément nécessaire de définir une classe pour l'entrepôt.
2. **quatre rayons** qui contiennent les exemplaires disponibles à la vente de chacun des produits (*Sucre, Farine, Beurre, Lait*). Le nombre d'exemplaires d'un produit dans le rayon est limité par la taille du rayon.
3. **un tas de chariots** dans laquelle chaque client doit prendre un chariot avant d'entrer dans le magasin.
4. **une caisse** qui dispose d'un tapis où un client dépose ses produits un par un. Il ne peut y avoir qu'un seul client à un moment donné, qui dépose des marchandises sur le tapis. Le dépôt des produits par le client et la prise des produits par le caissier seront faite dans le même ordre *(Le tapis est FIFO)*. Le tapis a une taille finie et est circulaire. On pourra le modéliser sous la forme d'une suite de cases pouvant contenir un produit ou le marqueur "client suivant".

Les intervenants à simuler sont :

1. **un chef de rayon** dont le rôle est de faire en sorte que les rayons soient le plus plein possibles. Il peut transporter jusqu'à 5 exemplaires de chaque produit à la fois, et passe son temps à aller à l'entrepôt, puis à faire le tour des rayons pour disposer le plus possible d'exemplaires de chaque produit dans le bon rayon. Le chef de rayon met les temps suivants pour ses opérations : 500 ms pour "faire le plein" à l'entrepôt, 200 ms pour se déplacer entre chaque rayon, ainsi qu'entre l'entrepôt et le premier rayon.
2. **un employé de caisse** dont le rôle est d'attendre qu'un client arrive à la caisse, et lorsqu'un client commence à déposer des éléments sur le tapis, de les prendre. Il s'arrête lorsque le tapis est vide. On simulera seulement la prise des éléments sur le tapis, jusqu'à trouver la marque "client suivant". Il faudra faire en sorte que le client et l'employé se synchronisent pour effectuer le paiement (dont on pourra considérer qu'il prend un temps négligeable). Ensuite, l'employé attend qu'un nouveau client dépose des marchandises sur le tapis ;

3. **Des clients** qui effectuent la suite d'action suivante :
   (a) Décider d'une liste de courses ; [1]
   (b) Prendre un chariot dans la file. Si celle-ci est vide, attendre qu'un chariot y soit déposé ;
   (c) Parcourir les quatre rayons et prendre les exemplaires nécessaires. On devra attendre d'avoir récupérer la quantité voulue d'un produit pour changer de rayon. Un client marche pendant 300 ms entre chaque rayon, les autres temps sont négligés.
   (d) Passer en caisse : s'il y a déjà un client qui passe ses objets en caisse, le client doit attendre que celui-ci ait fini pour commencer à placer ses marchandises. Le passage des clients en caisse peut ne pas être FIFO.
   (e) Placer ses produits sur le tapis. Le client met 20 ms à placer un élément sur le tapis. Une fois que tous ses produits ont été déposés, il place le marqueur de client suivant.
   (f) Attendre que l'employé ait fini de passer ses marchandises en caisse, afin d'effectuer le règlement.
   (g) Remettre son chariot dans la file.

   Les clients démarrent tous en même temps, et la simulation s'arrête lorsqu'il n'y a plus de clients à simuler.

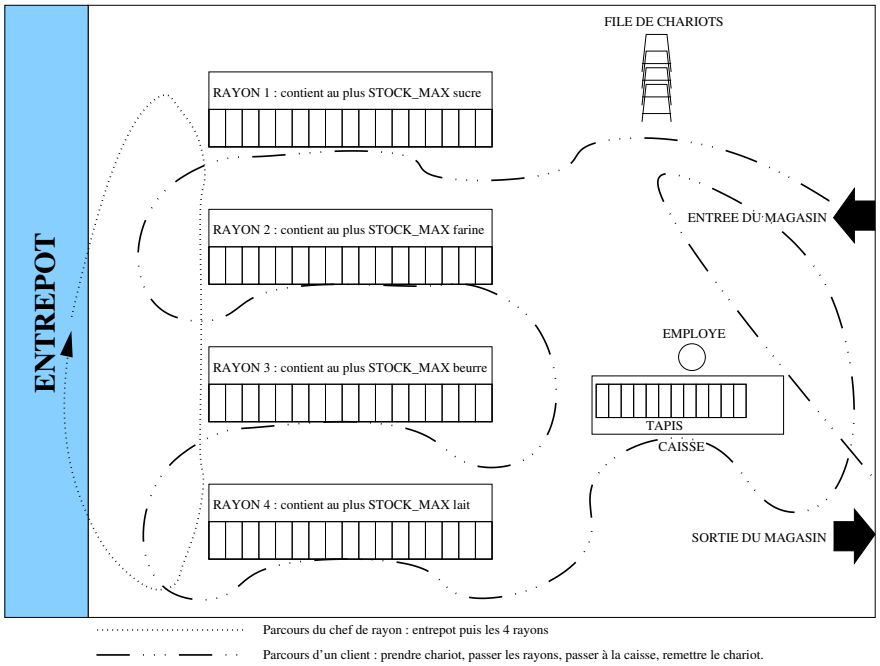

FIGURE 1 – Schéma de la supérette, avec le parcours d'un client et du chef de rayon.

La classe `Supermarche.java` contiendra la méthode `main()` qui va créer l'ensemble des ressources et des threads. Des constantes importantes y seront définies :

— `Supermarche.RAYON_STOCK_INIT` : le stock initial présent dans les rayons à l'ouverture du magasin ;

1. Vous pourrez la générer aléatoirement

— `Supermarche.RAYON_STOCK_MAX` : nombre maximum d'exemplaires d'un produit dans un rayon ;
— `Supermarche.TPS*` : les temps, en ms, des diverses opérations décrites dans le sujet pour lesquelles le temps n'est pas négligé ;
— `Supermarche.TAILLE_TAPIS` : nombre maximum d'objets présents sur le tapis de caisse ;
— `Supermarche.NB_CHARIOTS` : nombre de chariots dans la file à l'ouverture du magasin.

Les clients seront représentés par des instances d'une classe `Client.java`. Tous les objets sont créés dans le `main()` de `Supermarche.java`, et pour accéder à un élément du supermarché, ces objets seront des attibuts de classe accessibles publiquement.

2 Partie B : API REST de contrôle

Afin de faciliter le contrôle de la simulation, nous souhaiterions disposer d'une API REST de contrôle permettant les actions suivantes :

1. Rajouter plusieurs clients à la fois, fois,
2. Récupérer les informations courantes à propos des clients, en particulier leur état. Un client est à tout moment dans l'un des états suivants :
   — `ATTENTE_CHARIOT`
   — `EN_COURSE`
   — `ATTENTE_PRODUIT`
   — `ATTENTE_CAISSE`
   — `A_LA_CAISSE`

On demande donc d'implémenter l'API suivante :

URI	commande	description
/supermarche/clients	GET	renvoie la liste des clients
/supermarche/clients	POST	ajoute un certain nombre de clients
/supermarche/clients/{id}	GET	renvoie les informations du client
/supermarche/stock	GET	renvoie l'état des stocks

Voici quelques exemples de réponses attendues :

GET /supermarche/clients

```
[
  {
    "id": 0,
    "url": "/supermarche/clients/0"
  },
  {
    "id": 1,
    "url": "/supermarche/clients/1"
  }
]
```

GET /supermarche/clients/{id}

```
{
  "id": 0,
  "liste":{
    "sucre": 3,
    "farine": 0,
    "beurre": 1,
    "lait": 5
  },
  "etat": "EN_COURSE"
}
```

NB 1 : La liste représente la liste des objets qu'il lui reste à récupérer.
NB 2 : Le serveur devra renvoyer un code de retour 404 si l'id précisé n'existe pas dans la simulation.

3 Compte-rendu

Pour ce TP, vous devez rendre pour le 5 décembre 2016 23h59 CET, par mail adressé à votre chargé de TP et dont l'objet sera *[TP-SE] Groupe Nom1 Nom2* :

- les sources Java commentées (et correctement indentées), ainsi que l'information nécessaire à la compilation et l'exécution de votre projet ; Les sources devront contenir deux sous répertoires : partieA et partieB contenant chacun le code complet des parties en question.
- un compte rendu simple au format de votre choix :
  1. Il contiendra votre décomposition en classes, en particulier quels objets sont des Threads, et quels objets sont partagés. Il n'est pas demandé de diagramme UML ;
  2. La liste des motifs d'exclusion, les problèmes de synchronisation rencontrés, et comment vous les aurez résolus. N'hésitez pas à rapprocher les problèmes rencontrés de ceux rencontrés précédemment.
  3. Pour la partie REST, vous expliquerez vos choix de format de requêtes, des réponses attendues du serveur ainsi que les codes de retour.

Le barême provisoire suivant est donné à titre indicatif :

- 6 points pour le compte rendu ;
- 14 points pour le programme (fonctionnement, commentaires, lisibilité).