Projet SIM (Système et sIMulation)

Simulation de la ligne des caisses d'un supermarché

1 Préambule

L'objectif principal de ce projet consiste à comprendre la technique de **simulation guidée par les événements**. La partie cours concernant ce projet est dans les transparents qui sont dans le pot (répertoire SAP-SIM). Voir aussi, également dans le pot au même endroit, les simulateurs existants : le simulateur du pont et le simulateur d'ascenseur.

Avec l'expérience des années qui précèdent, le projet de SIM est une très bonne occasion d'approfondir vos connaissances en matière de **programmation par objets**.

Même si ce projet consiste à écrire un simulateur, cette architecture permet également le **partage du temps processeur** en évitant l'inconvénient majeur des *threads* (i.e. pas de problème de partage de mémoire dans le modèle de simulation algorithmique).

Les groupes qui avanceront le plus loin dans ce projet pourrons également s'exercer à augmenter les performances de leur projet (temps d'exécution et consommation mémoire).

2 Le sujet

Il s'agit de simuler le comportement des clients d'un supermarché lorsqu'ils arrivent devant la ligne des caisses. Il s'agit d'un grand supermarché, avec beaucoup de caisses. L'objectif consiste à aider le directeur du supermarché à choisir le nombre de caisses rapides qu'il est préférable d'installer et de réguler au mieux la prise de pause des caissiers.

On suppose que les clients ont un certain délai de patience et qu'un client qui attend trop avant de passer en caisse se met à ronchonner.

Grâce aux générateurs aléatoires à germe constant qui sont fournis, il est possible de rejouer une simulation avec les mêmes arrivées de personnes devant la ligne des caisses. Ainsi, il sera possible de recalculer le nombre de personnes ayant ronchonné en utilisant d'autres paramètres.

3 Paramètres de la simulation

3.1 Unité de temps

Pour cette simulation, l'unité de temps sera le dixième de seconde et une date sera représentée avec un `int`. Ainsi, un événement prévu pour se produire à la date 600 est en fait un événement survenant après 1 minute de simulation.

3.2 Principaux paramètres

Vérifier la correspondance des paramètres suivant dans le code source (chercher les définitions de constantes).

3.2.1 Nombre de caisses

Nombre total de caisses dans le supermarché : 15. Nombre de caisses rapides : 1 sur 5. Les caisses rapides ne prennent que les personnes ayant un caddy de 10 articles au plus.

3.2.2 Vitesse de travail du caissier

Pour simplifier, on considère que le caissier met un temps fixe pour scanner un article : 5 dixièmes de seconde.

3.2.3 Fréquentation du supermarché

Fréquence moyenne d'arrivée des clients devant la ligne de caisses : 55 dixièmes de secondes (soit environ un client de plus toute les cinq secondes).

3.2.4 Taille des caddys et délais de patience

Un client avec un petit caddy (≤ 10) est moins patient qu'un client avec un gros caddy (> 10). Un client avec un petit caddy ronchonne au bout de 4 minutes (soit 2400 dixièmes de secondes). Un client avec un gros caddy râle au bout de 10 minutes (soit 6000 dixièmes de secondes).

3.2.5 Passer la monnaie

Temps pour payer : 1 minute. C'est le temps pour donner son argent et recevoir la monnaie ou, si l'on paye avec une CB, le temps de taper son code et de valider l'opération.

3.2.6 Temps de pause des caissiers

A un instant donné, il y a toujours 2 caissiers qui sont en pause. Une pause dure exactement 5 minutes. Quand un caissier rentre de pose, il reprend sa place à sa caisse (la caisse passe alors dans l'état 'O'). Le choix du prochain départ en pause se fait équitablement, selon le nombre de pauses déjà prises à chaque caisse. On inspecte les caisses de la gauche vers la droite pour choisir le prochain caissier devant partir en pause. Un indicateur visuel permet d'indiquer aux clients qu'un caissier va prendre sa pause dès que possible ('P'). Avant de quitter son poste, le caissier qui part en pause termine les clients en attente (ce temps n'est pas décompté de sa pause).

Quand un caissier rentre de pause et reprend son poste, il peut se produire des migrations de clients. Pour simplifier la simulation, il s'agit du seul cas ou l'on prend en compte les migrations. En outre, seuls les clients des caisses directement voisines peuvent migrer à condition bien entendu que cette migration leur fasse gagner des places (position dans la file d'attente). L'algorithme de migration doit favoriser les plus gros gains de places.

4 Liste des événements

4.1 ACC (Arrivée Client Caisses)

L'instant précis où un client arrive devant la ligne des caisses. En plus de tirer au hasard le temps d'occurrence des ACCs (voir plus haut la fréquence d'arrivée aux caisses), on tire au hasard la caisse devant laquelle le client se présente. C'est à partir de ce point de départ que le client va véritablement choisir sa caisse de destination. Donc, soit la caisse initiale convient, soit le client se déplace vers une caisse à droite ou à gauche à l'aide d'un événement CBC. Un client met une seconde pour se déplacer vers la ligne de caisse qui est juste à coté (droite ou gauche).

4.2 CBC (Choisir Bonne Caisse)

L'instant précis ou un client arrive devant une autre caisse afin de voir si elle va finalement convenir. Lors des 5 premiers essais, le client cherche une caisse complètement vide. Lors des essais suivant, [6 .. 10], il se contente d'une caisse avec 2 personnes au plus. Après le 10 ième essais, il s'engage dans la première caisse possible.

Si la caisse devant laquelle on arrive ne convient pas, alors on regénère un autre CBC vers la caisse suivante en suivant la même direction. Bien entendu, en bout de ligne on part en sens inverse. Comme il faut une seconde pour se déplacer vers une caisse directement voisine, le CBC suivant est produit avec un décalage de 60 par rapport au CBC initial. Comme il faut bien que le client passe en caisse un jour, on se limite à 10 essais. Ainsi, lors du 11 essais, même s'il y a du monde dans la file d'attente, le client s'y engage. Attention, si le client possède plus de 10 articles et qu'il arrive devant une caisse rapide lors du 11î essai, alors il fait un 12î essai. Bien entendu personne ne s'engage dans une caisse ayant l'état 'P' (état qui indique que le caissier va partir en Pause).

4.3 CGR (Client Grincheux qui Ronchonne)

L'instant précis où un client se met à ronchonner. Un client peut ronchonner plusieurs fois. Pour prévoir cet événement, il faut le construire à l'instant où un client se place dans la file d'attente. Il faut aussi penser à l'enlever lorsque le client paye.

4.4 FSC (Fin de Service d'un Client)

L'instant précis où un client termine de payer ses courses. A cet instant, si un autre client est en attente devant la même caisse, il est immédiatement pris en charge par le caissier et il est donc possible de calculer un nouveau FSC.

4.5 FPC (Fin Pause Caissier)

L'instant précis ou un caissier rentre de pause. A cet instant précis, il se réinstalle à son poste de travail pouvant ainsi générer des migrations de clients (i.e. changement de file d'attente d'un client). Pour simplifier, on

considère que les seules migrations possibles se font par rapport aux deux caisses directement voisines. Le temps de migration des clients est considéré nul. Si des clients migrent, les premiers qui le font sont ceux qui sont en fin de la file d'attente.

5 Affichage normalisé

5.1 Affichage clients

Le client numéro 22 qui a 55 objets dans son caddy et qui est arrivé devant la ligne des caisses au temps 66 et qui a ronchonné 0 fois pour l'instant :

```
#22:55a:66t:0r
```

Lorsque le client a fini de payer, on indique également son temps de fin de paiment :

```
#22:55a:66t:0r:570f
```

5.2 Affichage de l'état d'une caisse

Une caisse normalement ouverte est symbolisée par la lettre `'O'`. Une caisse fermée sans caissier ni client en attente est symbolisée par la lettre `'F'`. Une caisse avec un caissier qui va bientôt partir en pause est symbolisée par la lettre `'P'`. La lettre `'R'` indique une caisse rapide (moins de 10 articles).

5.3 Vue globale de la ligne de caisses

```
----- Etat actuel du simulateur (nombre total de pas = 3) -----
 1 O -:
 2 O -:
 3 O -:
 4 F -:
 5 O R:
 6 O -:
 7 O -: #2:82a:21t:0r
 8 F -:
 9 O -: #3:32a:40t:0r
10 O R:
```

```
11 O -:
12 O -:
13 O -:
14 O -: #1:50a:0t:0r
15 O R:
Liste des ronchons:
Nombre total de ronchons: 0
Nombre total de ronchonnements: 0
Echeancier = [72,ACC 8],[375,FPC 4],[450,FPC 8]
Taper "Enter" ou le nombre de pas de simulation que vous voulez réaliser
```

6 Consignes de programmation

Le code servant de point de départ est compilable et comporte presque entièrement tous l'affichage. Vous pouvez ajouter des attributs mais toutes les redondances de données sont strictement interdites.

Il doit être possible de changer très facilement les paramètres de la simulation (voir par exemple la classes `Constantes.java`.

Ne jamais laisser une partie non programmée sans code. Il faut dans ce cas utiliser `aProgrammerPlusTard()`.

Il faut également se servir de `incoherenceOuTrucBizarre` pour marquer les endroit du code où l'on ne doit jamais passer.

Éviter autant que possible les fuites de mémoire.

Sinon, il faut absolument que la comparaison `==` soit utilisée quand c'est possible.