Haute école d'ingénierie et de gestion du Canton de Vaud
Département TIC
Laboratoire de programmation concurrente 2 (PCO2)

Temps à disposition : 8 périodes (travail débutant le mercredi 28 septembre 2011)

Objectifs pédagogiques

- Mettre en évidence les problèmes liés à la synchronisation des tâches Java;
- Modéliser un problème connu de synchronisation par blocs de synchronisation Java.

Enoncé

Dans ce laboratoire, nous souhaitons reprendre un problème connu fait en PCO1 et de le réaliser en Java par le biais de moniteurs.

Une ville comprend un ensemble de sites permettant à ses citadins d'aller de site en site en vélo. Chaque site comporte un nombre fixe de bornes et chaque borne ne peut attacher qu'un seul vélo à la fois. Les habitants prennent un vélo d'un site, se rendent à un autre site puis arriment le vélo à une borne du site pour un prochain usage.

De manière succincte, un habitant a le comportement suivant :

Boucle infinie

1. Attendre qu'un vélo au site i devienne disponible et le prendre.
2. Aller au site $j \neq i$.
3. Attendre qu'une borne du site j devienne libre et libérer son vélo.
4. Faire une activité près du site j.
5. $i \leftarrow j$

Fin de la boucle

Quand plusieurs habitants entrent en conflit pour une attente quelconque, par exemple quand ils attendent qu'une borne se libère ou qu'un vélo deviennent disponible, on supposera que ces habitants se servent selon leur *ordre d'arrivée*. Initialement, tous les habitants sont sur un site. Les sites de déplacement sont ensuite choisis de manière aléatoire.

Cette ville a aussi une équipe d'employés qui répartissent au mieux les vélos parmi les sites afin de les ramener aux destinations les moins populaires et aussi de laisser des bornes vacantes (éviter la surcharge d'un site). Cette équipe circule de site en site de manière continue à l'aide d'une camionnette pouvant transporter des vélos.

Hypothèses et contraintes à respecter

Réalisez l'énoncé avec les hypothèses et les contraintes ci-dessous :

1. Le nombre de sites, $S \geq 2$, et le nombre d'habitants sont constants durant l'exécution du programme. Ces nombres devront être obtenus par une saisie clavier.
2. Chaque habitant est modélisé par un thread.
3. Le temps que les habitants prennent pour se déplacer entre deux sites, ainsi que la durée de l'activité qu'ils réalisent à un site, seront modélisés par des temps aléatoires.
4. Le nombre de bornes par site, $B \geq 4$, est aussi obtenu par saisie clavier et demeure invariable durant toute la durée du programme. Tous les site ont le même nombre de bornes.
5. La camionnette de maintenance peut contenir au plus 4 vélos.
6. Le nombre de vélos, V, doit aussi être saisi et doit respecter $V \geq S\ (B - 2) + 3$. Initialement, chaque site contiendra $B - 2$ vélos, et le solde sera déposé dans le dépôt d'où part et revient l'équipe de maintenance.

7. L'équipe de maintenance est modélisée par un thread. En notant par D le nombre de vélos au niveau du dépôt et par V_i le nombre de vélos au site i, l'équipe de maintenance aura le comportement suivant :

   Boucle infinie
   1. Mettre $a = min(2, D)$ vélos dans la camionnette
   2. $D \leftarrow D - a$
   3. Pour $i = 1$ à S faire
      3a. Si $V_i > B - 2$ alors
          Prendre $c = min(V_i - (B - 2), 4 - a)$ et les mettre dans la camionnette
          $a \leftarrow a + c$
      3b. Si $V_i < B - 2$ alors
          Laisser $c = min((B - 2) - V_i, a)$
          $a \leftarrow a - c$
   4. Vider la camionnette $D \leftarrow D + a$.
   5. Faire une pause.

   Fin de la boucle
8. Le temps requis pour se déplacer entre les sites et aussi entre le dépôt et les sites sera un temps aléatoire, mais la durée de la pause de l'équipe sera constante. Afin de simplifier le problème, on supposera que le chargement des vélos dans la camionnette et leur déchargement se fait de manière instantanée (= 0).
9. Les vélos peuvent disparaître et aussi être remplacés. Pour simuler ce comportement, il devrait être possible d'incrémenter le nombre de vélos en réserve dans le dépôt et d'éliminer des vélos libres des différents sites. L'ajout et l'élimination devront être saisie par clavier par l'utilisateur du programme et à n'importe quel moment. Il est suffisant d'ajouter et d'éliminer un seul vélo à fois.
   Remarquons que ces interventions modifient dynamiquement des variables partagées et nécessiteront des synchronisations supplémentaires. Cette saisie clavier pourrait être gérée par le programme principal ou être réalisée par un thread supplémentaire.
10. Pour mettre fin à l'exécution du programme, mettez au point une façon gracieuse de terminer toutes les tâches. La terminaison se fera à la demande explicite de l'utilisateur.
11. Vos threads ne devront comporter aucune attente active, et toutes les synchronisations devront être faites **uniquement par des blocs de synchronisation** Java. *Votre solution ne peut pas se fier sur une modélisation par sémaphores implémentés par blocs de synchronisation.*

Travail à rendre

- Vous devez nous rendre un listage complet de vos sources et aussi nous les transmettre par courrier électronique.
- La description de votre implémentation, ses différentes étapes, la manière dont vous avez vérifié son fonctionnement et toute autre information pertinente doivent figurer dans les programmes rendus. Aucun rapport n'est demandé.
- Inspirez-vous du barème de correction pour connaître là où il faut mettre votre effort.
- Vous pouvez travailler en équipe de deux personnes.

Barème de correction

Conception	10%
Respect des contraintes	25%
Exécution et fonctionnement (démo)	10%
Codage	20%
Documentation et en-têtes des fonctions	25%
Commentaires au niveau du code	10%