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 2 novembre 2011, semaine 6)

Objectif

Modéliser un problème de synchronisation par rendez-vous Ada.

Enoncé

Le problème à modéliser est le même que le laboratoire précédent.

Contraintes à respecter

- Les contraintes relatives au problème restent identiques à l'énoncé précédent.
- Les habitants et les sites devront être représentés par des tâches identiques. Toutes les tâches auront un comportement cyclique.
- La synchronisation des tâches se fait **uniquement par rendez-vous**.
- Pour mettre fin à l'exécution du programme, mettez au point une *façon gracieuse* de terminer toutes les tâches. Cette terminaison se fera à la demande explicite de l'utilisateur. Aussitôt que la terminaison est demandée, les habitants ne devront plus pouvoir prendre de vélos et ne pourront que les déposer auprès des sites. Les tâches représentant les sites se terminent quand elles n'auront plus de rendez-vous avec les habitants. L'équipe de maintenance devra par contre continuer à faire ses tournées répartissant les vélos tant qu'il y a des habitants pouvant déposer des vélos.

Indications

- Pour effectuer un tirage d'une variable aléatoire uniforme, il faut utiliser le paquetage *Ada.Numerics.Float_Random* comme illustré dans le programme *Exemple* ci-dessous. Notons que le générateur de nombres aléatoires de ce paquetage n'est pas le meilleur connu, mais il convient parfaitement pour ce laboratoire.

```
with Ada.Numerics.Float_Random; use Ada.Numerics.Float_Random;
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;

procedure Exemple is
   Generateur_Aleatoire: Generator;
   attente: Float;
begin
   -- Initialiser le générateur (à faire une seule fois)
   Reset(Generateur_Aleatoire);
   -- Affiche 5 nombres aléatoire dans 0..10
   for i in 1..5 loop
      attente := 10.0 * Random(Generateur_Aleatoire);
      Put("Attente de "); Put(attente); Put_Line(" s.");
      -- exemple d'utilisation de l'instruction delay
      delay Duration(attente);
   end loop;
end Exemple;
```

- Rappelons que le langage Ada dispose de tableaux semi dynamiques. Le programme principal peut alors appeler une procédure ayant pour paramètre la taille d'un tableau déclaré localement à cette procédure. S'il s'agit d'un tableau de tâches, ces tâches démarreront aussitôt la fin de l'élaboration des déclarations de la procédure. Toutefois, la

procédure ne pourra se terminer et retourner à l'appelant que lorsque toutes les tâches qu'elle a créées finissent.
- Lors d'un rendez-vous, la tâche appelée peut fournir des indications à l'appelante. Par exemple, si la boutique est modélisée par une tâche, cette tâche peut fournir une indication aux clients faisant rendez-vous avec elle et leur transmettre le fait que la boutique est pleine.

Travail à rendre

- Vous devez nous rendre un listage complet de vos sources et aussi nous les transmettre par courrier électronique.
- La description de l'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	15%
Respect des contraintes	25%
Exécution et fonctionnement (démo)	10%
Codage	15%
Documentation et en-têtes des fonctions	25%
Commentaires au niveau du code	10%