Université Paul Sabatier
31062 Toulouse Cedex 09

Master 1 Informatique - MCPR

Modèles et Concepts du Parallélisme et de la Répartition
Travaux pratiques n° 4

Threads Java

Threads Java et synchronisation du type « moniteur »

Aucun langage de programmation n'implante le concept de moniteurs de Hoare en tant que tel. Ce qui se rapproche le plus de ce concept sont la synchronisation des threads Posix par mutex et conditions ou la synchronisation de threads Java par verrous réentrants.

Ce dernier langage sera utilisé dans le cadre de ces TP afin d'illustrer, d'un point de vue plus pratique, la « synchronisation de processus » à l'aide du concept de moniteurs.

Les threads Java et les classes et méthodes utiles à ce type de synchronisation sont présentés dans le support de TP associé.

Exercice

On désire simuler le déroulement d'une partie de ping-pong. Un joueur est simulé par un thread, il est caractérisé par sa position par rapport à la table de jeu.

Le comportement des processus symbolisant les joueurs sont ainsi définis :

```
Processus Joueur (typeDuJoueur) {
      while (partieEnCours)
              JouerCoup(typeDuJoueur) ;
}
```

où `typeDuJoueur` peut avoir la valeur `PING` ou `PONG`.

Première étape (absence de synchronisation)

- Écrire une **classe** Java implantant un **joueur** (jouant `PING` ou `PONG`) à l'aide d'un thread.
- Tester cette classe en écrivant une application dans laquelle deux joueurs (un de chaque type) s'exécutent.
- Modifier l'application pour que `N` joueurs jouant `PING` affrontent `M` joueurs jouant `PONG`, `N` et `M` devenant les paramètres de l'application.

Seconde étape (ajout de la synchronisation)

La table utilisée par les joueurs est une **ressource partagée** et les accès des joueurs doivent être synchronisés de manière à obtenir un jeu cohérent : un `PING` doit être suivi d'un `PONG` qui doit lui-même précéder un `PING` et ainsi de suite. Ceci signifie que les accès des joueurs doivent être synchronisés de manière à assurer l'**alternance** des coups.

- Modifier l'application précédente en ajoutant une **classe** jouant le rôle d'un **moniteur** qui va proposer des opérations (accederTable et libererTable, par exemple) permettant de synchroniser leurs accès à la table.