Thème 10

Objectifs

– Classe `Semaphore`

Exercices TD

Exercice 29 – Le problème de la piscine avec des sémaphores

Nous reprenons le problème de la piscine que nous choisissons maintenant d'implémenter avec des sémaphores.

Question 1

Proposez une nouvelle implémentation de la classe `Piscine` dans le cas où on n'identifie pas le panier ou la cabine utilisés par un nageur, mais où l'on s'assure juste que les accès au bassin restent contraints par le nombre de ressources disponibles.

Question 2

Que faut-il modifier dans l'implémentation du nageur ?

Exercice 30 – Les babouins du parc Kruger

Ce problème est un problème classique de synchronisations entre classes d'utilisateurs. On en trouve différents habillages dans beaucoup de livres de système. Celui-ci est extrait de *The Little Book of Semaphores, A.B. Downey*.

Il y a dans le parc national Kruger en Afrique du Sud un profond canyon dont la rive nord et la rive sud sont reliées par une corde. Les babouins peuvent traverser en avançant une main après l'autre sur la corde, mais si deux babouins se retrouvent face à face, ils se battent, tombent et meurent.

En supposant que l'on puisse apprendre aux babouins à utiliser des primitives de synchronisation, nous voulons construire sous différentes hypothèses des programmes qui doivent garantir que tous les babouins survivront.

Nous utiliserons pour représenter la position des babouins le type énuméré suivant :

```
// An enum constant may be followed by arguments, which are passed to the constructor of the enum when
// the constant is created during class initialization (Java Language Specification, chap. 8.9).

public enum Position {
    NORD(0), SUD(1);

    private Position(int index) {
        this.index = index;
    }

    private final int index;
    public int index() {return index;};
}
```

On note qu'on peut ajouter aux constantes du type `enum` des arguments constants, ici un champ `index` qui vaut `0` pour la constante `NORD` et `1` pour la constante `SUD`. On pourra ainsi lier facilement une position à un indice de tableau.

Quelle que soit la stratégie adoptée pour la traversée du canyon, les babouins exécutent la suite d'opérations ci-dessous :

```
public void run() {
    try {
        laCorde.acceder(position);
        System.out.println(this.toString() + " a pris la corde.");
        traverser();
        System.out.println(this.toString() + " est arrive.");
        laCorde.lacher(position);
    }
    catch (InterruptedException e) {
        System.out.println("Pb babouin !");
    }
}
```

où `laCorde` est une référence sur un objet d'une classe `Corde` et `traverser()` une méthode interne à la classe `Babouin` simulant par un délai aléatoire la traversée du canyon. La position passée en paramètre des méthodes correspond à la rive sur laquelle se trouve le babouin.

Question 1
Si on suppose que la corde n'est pas très solide ou les babouins pas très intelligents (ils ne savent pas compter), on choisit de mettre en place une stratégie où un seul babouin peut à un instant donner se trouver sur la corde.

Proposez une implémentation de la classe `Corde` d'abord en réalisant les synchronisations avec un (des) objet(s) de la classe `Lock`, puis avec un (des) objet(s) de la classe `Semaphore`. Pourrait-on utiliser des méthodes synchronisées pour réaliser cette stratégie ?

Il semble que la corde puisse quand-même supporter plusieurs babouins simultanément. Nous souhaitons donc modifier la stratégie précédente pour permettre une meilleure utilisation de la corde, tout en préservant la sécurité des babouins.

Question 2
A quelle condition un babouin peut-il s'engager sur la corde ? Comment peut-on implémenter cette condition ?

Question 3
Proposez deux implémentations des méthodes `acceder` et `lacher`, l'une utilisant des `Lock`, l'autre utilisant des `Semaphore`.

Exercices TME

Lors de chaque TME, vous devrez créer un répertoire pour chaque exercice, y mettre les fichiers s'y rapportant et soumettre ce répertoire à la fin du TME.

Exercice 31 – Le roi Kong du parc Kruger

Nous reprenons le problème des babouins, avec la stratégie des accès multiples à la corde.

Question 1
Reprenez une des implémentations de la classe `Corde` vue en TD, complétez la classe `Babouin` et programmez une classe de test afin de vérifier que cette stratégie fonctionne correctement.

Il semble finalement que la corde ne soit pas si solide que ça, et qu'il vaille mieux limiter à 5 le nombre de babouins qui s'y accrochent.

Question 2
Proposez une nouvelle implémentation de la classe `Corde` en utilisant des objets de type `Lock`.

Question 3

Proposez une autre implémentation de la classe `Corde` qui réalise la même stratégie en utilisant des objets de type `Semaphore`.

Il y a, parmi la population de babouins, un babouin beaucoup plus grand et beaucoup plus fort que les autres que les gardiens ont surnommé Kong. Lorsque Kong prend la corde, aucun autre babouin ne doit y accéder, sous peine de les retrouver tous au fond du ravin.

Question 4

Complétez la classe `Corde` en ajoutant deux méthodes `accederKong` et `lacherKong` qui permettent d'assurer la sécurité de Kong lorsqu'il accède à la corde.

Vous proposerez une solution à base de sémaphores et une solution à base de verrous. On souhaite éviter que l'écriture de ces méthodes entraine des modifications dans les méthodes existantes.