Université Bordeaux 1
Master 2 SDRP

arnaud.casteigts@labri.fr
2007–2008

TD/TP - Java/Threads (sujet de Pascal Grange)

1 Introduction

L'objectif du TP est de se familiariser avec les Threads en java. L'idée est de créer une "course de poulets" et d'étudier les différents comportements de l'application selon les techniques utilisées.

Récupérez l'archive `http://www.labri.fr/perso/casteigt/enseignement/m2sdrp/PADIS/Threads/chickenrun.tgz` et décompressez la où vous voulez.

2 Travail

A rendre : un fichier avec les réponses (par mail).

2.1 Analyse du code

1. Compilez et testez l'application (Méthode `main` dans la classe `Circuit`).
2. Expliquez brièvement le rôle de chacune des trois classes et de chacune des méthodes de ces classes.
3. Changez les couleurs des poulets afin qu'ils aient chacun la leur.

2.2 Freinons les poulets

Vous pouvez observer que la course ne se déroule pas sous nos yeux : elle se termine immédiatement sans que l'on ait vu les poulets progresser... Vous allez donc tenter de les faire ralentir.

1. La méthode Thread.sleep permet d'endormir le thread qui fait cet appel. Grâce à celle-ci, nous pouvons ralentir l'exécution de notre programme et visualiser le déroulement de la course. Repérez, dans le code, un endroit approprié pour ajouter cet appel.
2. Ajoutez l'appel à Thread.sleep dans le programme, testez à nouveau. Qu'observez-vous ? En consultant la documentation de la méthode `repaint`, formulez une hypothèse sur les causes possibles du comportement que vous observez.

2.3 Utilisation des threads dans le circuit

Seule l'utilisation d'une thread dédiée à la course permettra de résoudre le problème mis en évidence précédemment.

1. Créez un répertoire `threadedrace` et recopiez-y les trois fichiers.
2. Modifiez la méthode `startRace` appelée lorsqu'on clique sur le bouton *start* afin que celle-ci n'effectue pas la course mais crée un nouveau *thread* qui s'en chargera (le code peut-être entièrement écrit dans la définition du thread).
3. Si vous n'avez pas pris de précautions particulières lors de la question précédente, en cliquant plusieurs fois de suite sur le bouton *start* de l'interface graphique, un comportement étrange peut apparaître. Testez votre programme pour mettre en évidence ce comportement.

4. Tentez de découvrir la cause du comportement observé et proposez une solution. L'objectif étant qu'un clic sur le bouton *start* ne perturbe pas une course en cours.

2.4 Utilisation des threads dans les poulets

Nous allons maintenant faire en sorte que chaque poulet ait son flux d'exécution.

1. Créez un répertoire `threadedchicken` et recopiez-y les trois mêmes fichiers (ceux issus de l'exercice 2.2).
2. Quelles solutions proposez-vous pour modifier la classe `Chicken` afin que ses instances puissent être des *threads* Java ?
3. Modifiez, selon l'une de vos propositions, la classe Chicken dans cette optique. Faites en sorte que la méthode `run` de cette classe (corps du thread) invoque la méthode `move` du poulet jusqu'à ce que la course soit terminée.
4. Modifiez maintenant la méthode `startRace` de la classe `Circuit` afin que celle-ci ne *joue* plus la course, mais qu'elle se contente de démarrer les poulets (qui sont des threads indépendants). Faites attention au problème soulevé à la question 2.3-3.
5. Testez le programme, qu'observez-vous ? Tentez d'isoler la cause de ce problème et d'y remédier.