Thème 2 - TD

Objectifs

— Création de thread

Exercices

Exercice 3 – Création et lancement de threads : Hello World

Nous souhaitons écrire un programme très simple qui crée et lance l'exécution de plusieurs threads. La tâche affectée à chacun des threads est l'affichage de la chaîne de caractères *"Hello World"*.

Question 1
Quelle est la méthode qui contient le code exécuté par un thread ? Quelle est sa signature ?

Question 2
Proposez un programme qui lance trois threads affichant chacun *"Hello World"*, en passant par une sous-classe de `Thread`. Quel est l'inconvénient de cette solution ?

Question 3
Reprenez le programme précédent en utilisant cette fois une classe qui n'est pas une sous-classe de `Thread`.

Question 4
On veut mainteant individualiser les affichages. Chaque thread a un identifiant de type entier, et chacun salue une ville différente. On peut par exemple obtenir un affichage du type :

```
thread 2 salue Londres
thread 3 salue Tokyo
thread 1 salue Paris
```

Modifiez chacune des implémentations précédentes pour obtenir cet affichage.

Exercice 4 – Producteur/Consommateur : le producteur

Le schéma Producteur/Consommateur est un modèle classique de synchronisation. Un (ou plusieurs) processus producteur(s) dépose(nt) des données dans une zone de stockage, de taille limitée ou non. Un (ou plusieurs) processus consommateur(s) va (vont) récupérer les données dans cette même zone de stockage.

La zone de stockage est une instance d'une classe `Buffer`. La classe `Buffer` contient un tableau dans lequel les données déposées, représentées par des entiers, sont stockées. Si la zone de stockage est pleine, rendant le dépôt impossible, un message est affiché.

Question 1
Quelles sont les informations nécessaires pour pouvoir gérer la zone de stockage ?

Proposez une implémentation de la classe `Buffer` avec les éléments nécessaires pour qu'un producteur puisse effectuer des dépôts. Chaque opération de dépôt donne lieu à un affichage. Vous ajouterez à la classe une méthode `toString` qui construit une chaîne de caractères donnant l'état du stock.

Question 2
Nous nous intéressons au système le plus simple possible : un unique producteur effectue un nombre `nbDepots` de dépôts, où `nbDepots` correspond au nombre de cases dans la zone de stockage. La valeur déposée est un entier tiré aléatoirement.

Écrivez une classe `Producteur`, de manière à pouvoir lancer l'exécution du producteur dans un `thread` séparé.

Question 3
Écrivez un programme permettant de tester l'exécution du producteur. Ce programme crée un buffer de 5 cases, puis un producteur qui est exécuté dans un thread séparé.

Question 4
On souhaite aussi contrôler l'état du buffer, à la fin de l'exécution, pour vérifier que les dépôts ont bien été stockés. On complète donc le programme de test en ajoutant un appel à la méthode `toString`.

L'affichage de l'état du stock est-il correct ?

Question 5
Comment peut-on garantir que l'affichage n'a lieu qu'une fois que le producteur a effectué l'ensemble de ses dépôts ?