Thème 5 - TD

Objectifs

— Synchronisation de threads

Exercice

Exercice 8 – Synchronisation de threads : le crible d'Eratosthène

Le crible d'Eratosthène est un algorithme de recherche des nombres premiers bâti sur une propriété simple : si un entier p n'a aucun diviseur premier, alors p est premier.

Nous souhaitons réaliser une application qui permette d'effectuer une recherche parallèle des nombres premiers. Cette application est basée sur une chaîne de threads. Chaque thread se voit attribuer à sa création un identifiant id.

Lorsque le thread `id` reçoit une valeur `p` à traiter :

- si `id` divise `p`, le thread `id` ne fait rien ;
- si `id` ne divise pas `p`, et si le thread `id` n'a pas de successeur, il se crée un successeur d'identifiant `p` ;
- si `id` ne divise pas `p`, et si le thread `id` a un successeur, il transmet la valeur de `p` à son successeur.

On suppose que chaque thread (hormis le thread de la classe principale) exécute la méthode `run` d'une instance d'une classe `Erato`. Cette classe dispose d'un constructeur permettant de transmettre à l'instance la valeur de `id`. Le chaînage démarre avec la création par la classe principale d'un thread exécutant `Erato(2)`.

Question 1
Explicitez le déroulement de l'application lorsque le thread qui exécute `Erato(2)` reçoit successivement toutes les valeurs entre 2 et 8.

Le chaînage des threads est unidirectionnel : un thread possède une référence vers son successeur uniquement. La transmission d'une valeur se fait donc par un appel à une méthode du successeur (que nous appellerons `forwarder`).

Question 2
Quelle solution proposez-vous pour s'assurer que, lorsqu'un thread envoie une suite de valeurs à son successeur, chaque valeur envoyée est effectivement traitée ?

Nous choisissons de stocker dans une unique variable la suite de valeurs reçues par un thread.

Question 3
Dans quel cas un thread peut-il se retrouver (temporairement) bloqué ? Qu'en est-il du thread principal ?

Question 4
Quels sont les variables et objets nécessaires à la réalisation de l'attente ?

Question 5
Donnez les variables d'instance et le constructeur de la classe `Erato`.

Question 6
Écrivez une méthode permettant à un thread exécutant une instance de la classe `Erato` d'attendre qu'on lui envoie une valeur à traiter.

Question 7
Écrivez la méthode `forwarder` appelée par le prédécesseur du thread pour lui envoyer une valeur à traiter.

Question 8
Écrivez une méthode `traiter` qui effectue le traitement d'une valeur reçue (rien, création d'un thread successeur ou transmission au thread successeur suivant les cas).

Question 9
Écrivez la méthode `run` de la classe `Erato` et un programme de test permettant d'identifier les nombres premiers entre `2` et `100`.

Nous nous intéressons maintenant au problème de la terminaison de cette application.

Question 10
À quel moment un thread peut-il se terminer ?

Question 11
Proposez une solution pour gérer la terminaison.