Java inside

Lab 6 : **Continuation**

Continuation, Concurrence coopérative

SORAN Altan - ESIPE INFO 2019

https://github.com/asoran/java-inside

JDK 14

Le JDK14 est encore en cours de développement, mais on peut déjà tester les développement en cours pour pouvoir comprendre les notions en avance voir même aider au développement. Elle peut être télécharger depuis ici:

Pour linux: jdk-14-loom-linux.tar.gz
Pour macOS: jdk-14-loom-osx.tar.gz

Si on veut l'utiliser, il ne faut pas oublier de mettre à jour le path.
Ou encore, si on utilise maven par exemple, il ne faut changer la variable d'environnement JAVA_HOME.

Continuation

Le JDK 14 implémente le système de Continuation déjà présents dans certains langages comme C#, Perle ou encore Python.

Une continuation est une représentation abstraite, c'est un point de vue pour décrire l'état de la machine à un point donné.

Il permet de créer des fonctions qui peuvent être arrêtés et stockés dans le tas “au milieu” de la fonction, pour reprendre son exécution plus tard.

Ces arrêts et reprises de fonctions c’est à nous de le coder.
En java par exemple, ça veut dire qu’ils ne dépendent pas du scheduler.

Continuation en JAVA

Pour créer une continuation, il faut 2 choses:

- Un scope, qui va définir les limites de la mémoire de la continuation
- Un runnable, ça va être le code à exécuter !

On lance et reprend les continuations en appelant la méthode `run()`
On stop le code avec l'appelle `Continuation.yield()`

```
var scope = new ContinuationScope("scope");
var contiuation = new Continuation(scope, () -> {
    System.out.println("This will be displayed");
    Continuation.yield(scope); // Pause until run is called again
    System.out.println("This will not be displayed");
});
contiuation.run(); // "This will be displayed" will … be displayed
```

JAVA - Continuation VS Threads

Les continuations et les threads se ressemble en certains points. Ils ne s'opposent pas, il peuvent cohabiter. On peut par exemple, créer plusieurs continuation dans un même thread et inversement.
Les deux exécutent du code passer sous forme de Runnable, mais il existe plusieurs différences importantes :

- Threads: C'est l'OS qui gère complètement le scheduling. Un Thread s'exécute sur un thread système, il est parallélisé. Concurrence non coopérative/compétitive

- Continuation: On décide de l'exécution du code. Il s'exécute sur le thread courant, c'est comme un apelle de fonction, le code qui vient après va s'exécuter après, et pas en même temps !

Bloc Synchronized et Continuation

On peut créer des Thread dans une continuation, les lancer et mettres des bloc synchronized. Mais il y a une règle à retenir.

On ne peut pas appeler `Continuation.yield()` dans un bloc synchronized ! Faire ceci lèvera une IllegalStateException !

Pourquoi ?

Une continuation peut être passé par référence, donc peut potentiellement être exécuté dans un autre thread que celui dans lequel il a été lancé, et donc l'entrée et la sortie du bloc synchronized peut se faire dans deux threads différents, et on ne veut surtout pas ça !

Continuation et Scheduler personnalisé

On a vu que c’est nous qui décidons quand le code s'arrête et reprend, on peut donc implémenter un mini Scheduler:

On stock la liste de Continuation à exécuter dans une Queue, ensuite, on dépile la Queue et on lance les continuation une par une, jusqu'à que la Queue est vide.

Il faut aussi qu’on remplace aussi tous les appelles de Continuation.yield() dans le code par un appelle à une fonction de notre Scheduler qui s’occupe de faire le yield et qui ajoute le Continuation (si elle n’a pas fini de s’exécuter) dans notre Queue pour qu’on puisse l’appeler plus tard.

(On peut aussi changer l’ordre dans lequel les Continuation sont choisis)

Configurer travis pour le JDK14

Le JDK14 étant ni fini ni sorti, il n'est pas gérer par travis.
On doit donc télécharger la dépendance de notre projet directement (la jdk dans notre cas). Voici une façon de faire ceci:
Dans la .travis.yml, on dit a travis de build sur 2 os, puis de lancer un script:

```
os:
  - linux
  - osx
script:
  - chmod u+w ./package.sh
  - ./package.sh
```

On délège le build à un scrip package.sh qui la va télécharger la bonne version dû jdk14 pour le bon os (en faisant des `if`), va mettre à jour la variable JAVA_HOME et va lancer maven pour tester notre application.