Java inside

lab 4 : **Performance, CallSite et JMH**

SORAN Altan

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

TDD

Test Driven Development

C’est le fait de d’abord rédiger tous les tests, et ensuite codé de façon à ce que les tests passent. Souvent c’est une personne différente qui rédige les tests et une personne différente qui tape le code.

Ça permet d’avoir en avance les use cases et les cas limites définies, on sait déjà ce qu’on veut faire et peut simplifier le développement en plus de pouvoir voir à l’avance des détails d’implémentation.

JMH

https://openjdk.java.net/projects/code-tools/jmh/

JMH est un outil permettant de créer, d'exécuter et d'analyser des tests de performances nano / micro / milli / macro écrits en Java et dans d'autres langages destinés à la machine virtuelle Java.

On doit annoter nos tests avec @Benchmark, ensuite générer le target avec maven, puis on peut lancer le benchmark.
Par exemple en ligne de commande:

```
java -jar target/benchmarks.jar
```

Lambda vs Classe anonyme

Rappel: On peut créer des class anonymes avec lambda.

Utiliser une lambda dans ce TP est mieux, car le JIT peut remarquer que notre lambda de Logger vide ne vas rien faire, et il va donc mémoriser ça et ne rien faire quand on appelle la fonction, donc prendre autant de temps que d'appeler une fonction vide (sans code à l’intérieur), contrairement à la classe anonyme qui va prendre du temps à s'exécuter alors que le code ne faire littéralement rien.

MutableCallSite

Un MutableCallSite est un “CallSite” dont la variable cible se comporte comme un champ ordinaire. Une instruction invokedynamic liée à un MutableCallSite délègue tous les appels à la cible actuelle du site. L'invocateur dynamique d'un site d'appel mutable délègue également chaque appel à la cible actuelle du “site”.

Il permet de créer des MethodHandler qui changent.

MutableCallSite::setTarget & MutableCallSite.syncAll

Avec MutableCallSite::setTarget on peut changer le MethodHandler cible du MutableCallSite.

Dans le cas multi thread, tous les thread peuvent ne pas mettre à jour leur référence, et c’est pour ça que MutableCallSite.syncAll existe, il force les thread a mettre a jour leur cache avec la nouvelle cible actuelle.