



TP2-Programmation concurrente

Réaliser par : Mohammed BENAOU

1 Ordonnancement des threads en Java

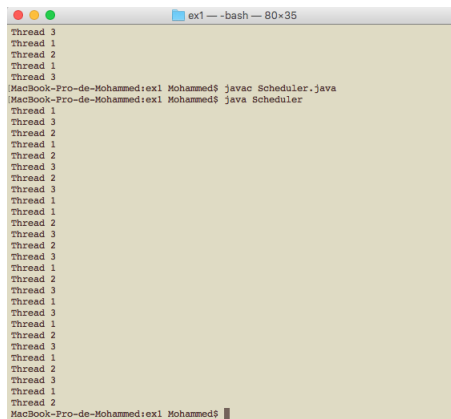
```
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java Scheduler
Thread 1
Thread 1
Thread 1
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 1
Thread 1
Thread 1
Thread 1
Thread 1
MacBook-Pro-de-Mohammed:ex1 Mohammed$
```

1-Exécution total du Run d'un Thread t puis un deuxième puis un troisième

```
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java scheduler.java
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java scheduler
Erreur : impossible de trouver ou charger la classe principale scheduler
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java scheduler.java
Erreur : impossible de trouver ou charger la classe principale scheduler.java
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java Scheduler.java
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java Scheduler
Thread 1
Thread 1
Thread 1
Thread 1
Thread 1
Thread 1
Thread 1
Thread 1
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 3
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
MacBook-Pro-de-Mohammed:ex1 Mohammed$
```

2-Exécution du Run d'un Thread t puis interruption de ce dernier pour commencer un deuxième Thread et ainsi de suite jusqu'à l'exécution de tout les Thread

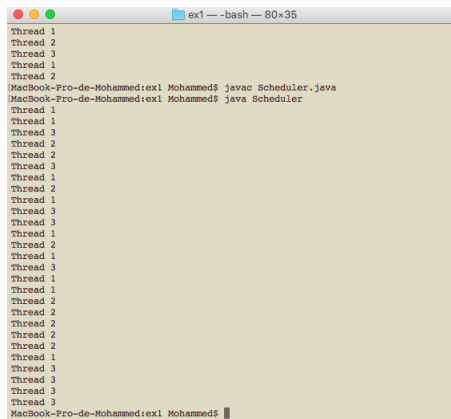
1.1



```
Thread 3
Thread 1
Thread 2
Thread 1
Thread 3
MacBook-Pro-de-Mohammed:ex1 Mohammed$ javac Scheduler.java
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java Scheduler
Thread 1
Thread 3
Thread 2
Thread 1
Thread 2
Thread 2
Thread 3
Thread 2
Thread 1
Thread 1
Thread 1
Thread 2
Thread 3
Thread 2
Thread 3
Thread 1
Thread 2
Thread 3
Thread 1
Thread 3
Thread 1
Thread 2
Thread 3
Thread 3
Thread 1
Thread 2
Thread 3
MacBook-Pro-de-Mohammed:ex1 Mohammed$
```

Thread.sleep() endore le Thread courant, dans ce cas on aura un affichage des 3 threads simultanément.

1.2



```
Thread 1
Thread 2
Thread 3
Thread 1
Thread 2
MacBook-Pro-de-Mohammed:ex1 Mohammed$ javac Scheduler.java
MacBook-Pro-de-Mohammed:ex1 Mohammed$ java Scheduler
Thread 1
Thread 3
Thread 2
Thread 3
Thread 1
Thread 2
Thread 1
Thread 3
Thread 3
Thread 1
Thread 2
Thread 1
Thread 1
Thread 1
Thread 2
Thread 2
Thread 2
Thread 2
Thread 1
Thread 3
Thread 3
Thread 3
MacBook-Pro-de-Mohammed:ex1 Mohammed$
```

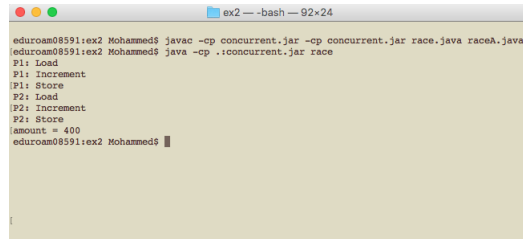
Thread.yield() Oblige le Thread appelant à faire l'exécution à un autre Thread prêt à s'exécuter sur le processeur actuel.
L'affichage des différents messages est effectué aléatoirement et le Thread le plus prêt à être exécuter le sera.

2 Problèmes d'accès concurrent

2.1

Les trois valeurs possibles sont : 200,300 et 400

`javac -cp concurrent.jar -cp concurrent.jar race.java raceA.java java -cp . :concurrent.jar race`



```
eduroam08591:ex2 Mohammed$ javac -cp concurrent.jar -cp concurrent.jar race.java raceA.java
eduroam08591:ex2 Mohammed$ java -cp .:concurrent.jar race
P1: Load
P1: Increment
P1: Store
P2: Load
P2: Increment
P2: Store
amount = 400
eduroam08591:ex2 Mohammed$
```

2.2

Après lancer plusieurs fois l'exécution du programme ce dernier produit bien ces trois valeurs

2.3

Il faut mettre Synchronized dans la classe p1 et aussi dans p2.

Dans ce cas quand la méthode run de P1 sera exécuter pour garder la valeur de la variable amount, du coup l'exécution du p2 sera fait qu'après p1.