TP1

Hugo Jacotot : 71802786

Matthieu Le Franc : 71800858

Exercice 1

Comment expliquez vous les différences de comportement ?

Dans un cas, on récupère l'id de la thread créatrice (start) et dans le deuxième on récupère l'id de la thread créée (run) et on travaille avec les données de la thread créée.

Avec la première version de `MyThread2` on obtient, lors de l'affichage de la thread lancée, l'ID du thread courant (avec tID.get()) et l'id stocké dans la variable myid.

```
la thread 10 0
la thread 5 0
la thread 3 0
la thread 6 0
la thread 9 0
la thread 2 0
la thread 7 0
la thread 1 0
la thread 8 0
la thread 4 0
```

l'id courant avec l'appel à get() donne effectivement l'id de la thread courante, mais l'id stocké dans la variable myid est toujours 0. Cela est dû au fait que la variable myid est définie dans le constructeur avec un appel à get() et à ce moment là, l'id retenue advitam eternam celui de la thread créatrice d'id 0.

Avec la deuxième version de `MyThread2`, on obtient :

```
la thread 7 7
la thread 8 8
la thread 9 9
la thread 1 1
la thread 5 5
la thread 4 4
la thread 2 2
la thread 6 6
la thread 3 3
la thread 0 0
```

Cette fois-ci, l'id stocké dans la variable myid est l'id de la thread créée. Cela est dû au fait que la variable myid est définie avec un appel à get() à l'intérieur du run()

Exercice 2

Après avoir exécuté le Main2.main quelles sont les valeurs affichées ?

```
la thread 1 a pour last 5000
la thread 0 a pour last 1000
value5996.0, valuebis5998.0 et  last 0
```

last = 0 car c'est le main qui exécute et donc la valeur de last n'est pas modifiée

Comment assurer qu'à la fin du Main2.main on ait o.value=o.valuebis= nombre totale d'écritures ?

Pour cela il faut ajouter un synchronized pour assurer l'atomisation (exclusion mutuelle) et que cela prenne aussi en compte l'impression de la valeur avec le System.out.print car autrement, la première valeur sera prête à être imprimée mais pas la deuxième qui pourra continuer à être incrémentée.

Quelle est la difference entre les variables value et last ?

La variable value est une variable partagée entre les threads et last est une variable locale à chaque thread.

Exercice 3

1/. Exécutez plusieurs fois le programme, est-ce que t contient toujours à la fin de l'exécution les valeurs 0,1,2,3,4 (dans un ordre quelconque) ?

Non, la plupart du temps toutes les cases du tableau contiennent la même valeur.

2/. Comment l'expliquez vous?

Aucun mécanisme de contrôle d'accès à la section critique n'est mis en place, les variables sont donc modifiées par tous les threads en même temps.

3/. Est ce que si on synchronise la methode echange on aura après les échanges toujours les valeurs 0,1,2,3,4 dans le tableau ?

Non, il y a toujours perte de valeurs

4/. Proposer une solution pour faire en sorte que après les échanges il y ait toujours les valeurs 0,1,2,3,4 dans le tableau.

Il faut synchroniser sur le tableau partagé à l'intérieur de echange

Exercice 4

Avec :

```
public static volatile int check=0;
```

On obtient :

```
check = 1 cur = 0
check = 2 cur = 1
check = 3 cur = 2
check = 4 cur = 3
check = 5 cur = 4
check = 6 cur = 5
check = 7 cur = 6
check = 8 cur = 7
check = 9 cur = 8
check = 10 cur = 9
coucou
received 11 stop
```

Avec :

```
public static int check=0;
```

On obtient :

```
received 11 stop
```

L'utilisation de volatile permet de garantir qu'on récupère une valeur à jour en allant regarder la valeur directement en mémoire partagée et non pas dans un registre local.

Donc, dans le cas où ce mot clef est utilisé, on obtient la valeur de check à jour, supérieure à cur, et on peut donc sortir de la boucle. Sans l'utilisation de volatile, on ne va pas chercher la bonne valeur de check pour effectuer la comparaison.

Exercice 5

a/. Peut il y avoir 10 Threads au niveau 1 simultanément ?

Oui car au niveau 1, tous les Threads peuvent entrer, même s'ils arrivent en même temps au niveau 1.

b/. Peut il y avoir 10 Threads au niveau 2 simultanément ?

Non car au niveau 2, il n'y a que 9 Threads qui peuvent entrer, le 10ème doit attendre qu'un Thread au niveau 2 ou supérieur se déclare victime.

c/. Si il y a au moins 1 thread au niveau 1 et aucune au niveau 2 (et supérieur) est ce qu'une des threads de niveau 1 peut passer au niveau 2

Oui car il n'y a pas de Thread au niveau 2 ou supérieur pour bloquer la progression.

d/. Peut il y avoir 9 Threads au niveau 2 simultanément ?

Oui car au niveau 2, il n'y a que 9 Threads qui peuvent entrer, même s'ils arrivent en même temps au niveau 2 (10 Threads en tout donc 10-1).

e/. Peut il y avoir 9 Threads au niveau 3 simultanément ?

Non car au niveau 3, il n'y a que 8 Threads qui peuvent entrer, le 9ème doit attendre qu'un Thread au niveau 3 ou supérieur se déclare victime.

f/. Si une thread accède à la section critique, à quels niveaux sont les autres threads ?

Les autres Threads sont au niveau 1 ou 2.

g/. Montrer la propriété suivante: pour tout j, $1 \leq j \leq n - 1$, il y a au plus $n - j + 1$ threads au niveau j.

- Dans l'algorithme, chaque Thread passe par une série de niveaux avant d'entrer en section critique (de 1 à n-1) et chaque niveau représente un degré de rapprochement vers la section critique.
- A chaque niveau, un Thread se déclare victime et attend qu'un Thread de niveau supérieur se déclare victime pour pouvoir progresser dans les niveaux.
- A chaque niveau j, le nombre maximum de Threads qui peuvent être présents au même niveau ou supérieur diminue.
- Au niveau j, tous les Threads au niveau supérieur à j sont nécessairement en train d'attendre ou avancer vers la section critique.
- En ajoutant un Thread au niveau j, on a au plus n - j + 1 Threads au niveau j.

Donc, pour tout j, $1 \leq j \leq n - 1$, il y a au plus $n - j + 1$ threads au niveau j.

h/. Montrer que une thread qui demande l'accès à la section critique parviendra en section critique.

- Chaque Thread passe par une série de niveaux avant d'entrer en section critique.
- Un Thread au niveau i est bloqué si un autre thread à son niveau ou au-dessus se déclare victime.
- Lorsqu'un Thread progresse dans les niveaux, il libère le niveau précédent pour les autres Threads.
- Le Thread victime change change à mesure que les Threads progressent dans les niveaux, ce qui empêche les Threads de rester bloqués indéfiniment.

Donc, un Thread qui demande l'accès à la section critique parviendra en section critique, l'attente active garantit que les Threads continue de vérifier l'état poru avancer, ce qui assure qu'aucun Thread ne reste bloqué indéfiniment.