

TP 6 Java

Exercice 1

Un objet de type Runnable implémente l'interface Runnable en définissant sa méthode abstraite run. C'est la méthode « main » d'un thread, appelée au démarrage du thread. Elle contient ce que doit faire le thread. Un tel objet permet donc la définition à proprement parler d'un thread.

Lorsqu'on lance plusieurs fois HelloThread, on s'aperçoit que le contenu de la console n'est jamais exactement le même, c'est-à-dire que les appels ne se font pas systématiquement dans le même ordre. C'est normal, Java gère les threads de façon optimale, qui n'est pas toujours la même suivant l'état de la mémoire de la machine hôte.

Exercice 3

Différents affichages :

Les threads ont fini de s'exécuter

Taille arrayList : 19498

Exception in thread "0" Les threads ont fini de s'exécuter

Taille arrayList : 15015

```
java.lang.ArrayIndexOutOfBoundsException: 22
    at java.util.ArrayList.add(ArrayList.java:459)
    at tp6.HelloListBug.run(HelloListBug.java:17)
    at java.lang.Thread.run(Thread.java:745)
```

Les threads ont fini de s'exécuter

Taille arrayList : 20000

Exception in thread "1" java.lang.ArrayIndexOutOfBoundsException: 366

Les threads ont fini de s'exécuter

```
    at java.util.ArrayList.add(ArrayList.java:459)
Taille arrayList : 15103
    at tp6.HelloListBug.run(HelloListBug.java:17)
    at java.lang.Thread.run(Thread.java:745)
```

Le problème se situe au niveau de l'incrémentation dynamique de la taille de l'ArrayList dans le cadre d'accès concurrents des différents threads. L'affichage correct est le troisième, celui où l'ArrayList a atteint sa taille définitive de 20000 éléments ; les autres témoignent des conséquences malheureuses d'accès concurrents sur un objet inadapté aux traitements parallèles, par exemple lorsque deux threads demandent en même temps d'ajouter un élément.

On passe donc à une taille fixe de 20000 éléments pour remédier à ce problème. On observe :

Les threads ont fini de s'exécuter

Taille arrayList : 19650

Les threads ont fini de s'exécuter

Taille arrayList : 19942

Les threads ont fini de s'exécuter
Taille arrayList : 19269

Les threads ont fini de s'exécuter
Taille arrayList : 19957

Il n'y a plus d'erreur mais tous les éléments ne sont pas systématiquement ajoutés.

On corrige le problème en utilisant CopyOnWriteArrayList, adapté aux traitements simultanés, à la place d'ArrayList :

Les threads ont fini de s'exécuter
Taille arrayList : 20000

Les threads ont fini de s'exécuter
Taille arrayList : 20000

Les threads ont fini de s'exécuter
Taille arrayList : 20000

On peut rajouter l'affichage de l'ajout des éléments pour vérifier que le traitement se fait bien en parallèle.

Exercice 4

Le programme sert à compter combien d'itérations sont faites en 100ms. Cependant, tel quel, rien ne s'affiche et la boucle ne s'arrête pas. On corrige le problème avec un bloc synchronized :

```
synchronized (this) {  
    localCounter++;  
}
```