TP1 : Tests Mémoire

EXERCICE 1

Q1. Sous netbeans, créez un projet Java, codez un Hello World et vérifiez que tout marche bien.

Q2. Lancez le profiler, en indiquant bien que vous souhaitez surveiller la mémoire. Le profiler est capable de faire une analyse détaillée de l'utilisation des ressources processeur ainsi que de l'utilisation de la mémoire, mais il ne peut pas faire les deux au même moment.

Q3. Modifiez le main de votre projet Java pour qu'il effectue une boucle infinie autour du code suivant :

```
try {
   Thread.sleep(1000);
} catch (InterruptedException e) {
}
System.out.println("Iteration: " + i);
```

Q4. Développez une méthode « exo1 » qui définit une liste et qui met dedans 100000 objets différents.

Q5. Appelez la méthode que vous venez de définir dans le main, juste après l'affichage de l'Itération.

Q6. Exécutez en utilisant le profiler, et interprétez les graphiques *Memory (Heap)* et *Memory (GC).*

EXERCICE 2

Q1. Ajoutez dans la classe précédente une liste en tant qu'attribut, nommée `maListe` et remplissez-la dans une méthode « exo2 », de la même manière que dans la méthode de l'exercice précédent.

Q2. Appelez la méthode « exo2 » que vous venez de définir dans le main, à la place de « exo1 ».

Q3. Exécutez en utilisant le profiler, et interprétez les graphiques *Memory (Heap)* et *Memory (GC).*

Q4. Commentez la différence entre les deux méthodes, en vous basant sur l'interprétation des graphiques.

Q5. Cherchez parmi les fonctionnalités du profiler, comment trouver la fuite dans le programme.

EXERCICE 3

Q1. Ajoutez dans la classe précédente une méthode « exo3 » dans laquelle vous placerez le code suivant :

```
for (int i = 0; i < 100000; i++) {
   maListe.add(new Object());
}
for (int i = 0; i < 100; i++) {
   maListe.remove(0);
}
```

Q2. Appelez cette méthode dans le main, à la place de celle de la question précédente.

Q3. Interprétez les graphiques. Il est fortement conseillé de laisser tourner le programme un maximum de temps afin de faciliter l'analyse.

Q4. Ajoutez à la fin de la méthode « exo3 » (après la fin de la deuxième boucle) l'instruction suivante :

```
System.gc();
```

Expliquez ce que fait cette instruction d'après son impact sur les graphiques.

EXERCICE 4

Q1. Codez un programme qui reproduit l'utilisation mémoire ci-dessous :