Santiago Bobadilla 201820728

Taller 1 – Threads

1. Complete la siguiente tabla, con respecto a la creación de threads usando la extensión de la clase Thread y la implementación de la interfaz Runnable.

Se parecen	Se diferencian
• Ambos requieren un main para correr. • El código por ejecutar se define en el método 'run'. • Ambos ejecutar (y deben dar 'start') a un objeto de la clase 'thread'. • Ambos reciben y pueden usar parámetros, variables propias y compartidas, además de características propias de los 'threads' como 'sleep'.	• El 'Thread' extiende de 'Thread', mientras que el 'Runnable' implementa la interfaz 'Runnable' • El 'Thread' como clase ya es un thread y por tanto se instancia un objeto de la misma clase. El 'Runnable' se debe crear un objeto thread que recibe como parámetro un objeto Runnable. Es decir, se instancian de manera diferente.

Taller 1b – Manejo de Threads

<u>Parte 1: Incremento de un contador</u>

Ejemplo 1: Aplicación monothread para el incremento de un contador.

1. ¿Al ejecutar el programa, el resultado corresponde al valor esperado?
   Sí, el valor esperado era `10000000, y el resultado corresponde a ese valor. Esto es lógico debido a que la ejecución se hace de manera secuencial.`

Ejemplo 2: Aplicación multithread para el incremento de un contador

2. ¿Al ejecutar el programa, el resultado corresponde al valor esperado? Explique
   No, el valor esperado era `10000000, y el resultado no corresponde con ese valor. La razón de esto es que existe concurrencia entre los threads. Todos acceden (lectura) al contador y entre eso accesos consecutivos definidos por el sistema se pierden números.`

3. Ejecute cinco veces el programa y escriba el resultado obtenido en cada ejecución

Ejecución	Valor Obtenido
1	9742523
2	9631842
3	9773488
4	9851248
5	9843371

4. ¿Hay acceso concurrente a alguna variable compartida? Si es así, diga en dónde.
   Sí, la concurrencia como ya se dijo está en el contador. Específicamente se da en el contador++. La razón es que el proceso de ++, es leer el valor del contador, aumentarle

uno y guardarlo. Si tengo el contador 5 y 6 al tiempo, por ejemplo, y ambos leen que el contador está en 42, ambos lo superan a 43, lo que implica que se perdió un número y esto se debe a que ambos threads modificaron el valor al tiempo. Cuando se extrapola esto al código solicitado los valores obtenidos tienen lógica porque muchos 'threads' tiene les pasa lo explicado con el ejemplo.

Parte 2: Elemento mayor en una matriz de enteros

Ejemplo 3: Aplicación multithread para encontrar el elemento mayor de una matriz de enteros

1. Ejecute cinco veces el programa y escriba el resultado obtenido en cada ejecución.
    - Ejecución 1:
        - Valor obtenido:

```
========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 2 - Maximo local actual: 94529 - Maximo global: 94529


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 1 - Maximo local actual: 94529 - Maximo global: 64451


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 0 - Maximo local actual: 94529 - Maximo global: 98423


ID Thread: 2 - Maximo Local: 94529 - Maximo Global: 94529
ID Thread: 1 - Maximo Local: 94529 - Maximo Global: 64451
ID Thread: 0 - Maximo Local: 94529 - Maximo Global: 98423
```

        - Valor esperado:

```
98423   42581   6182
9262    64451   50295
23037   94529   12250
```

        Para este caso sería 98423
    - Ejecución 2:
        - Valor obtenido:

```
========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 1 - Maximo local actual: 101024 - Maximo global: 72106


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 2 - Maximo local actual: 101024 - Maximo global: 101024


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 0 - Maximo local actual: 99786 - Maximo global: 99786


ID Thread: 1 - Maximo Local: 101024 - Maximo Global: 72106
ID Thread: 2 - Maximo Local: 101024 - Maximo Global: 101024
ID Thread: 0 - Maximo Local: 101024 - Maximo Global: 99786
```

        - Valor esperado:

```
99786   48747   45948
55176   72106   66422
101024  86255   86133
```

        Para este caso sería 101024
    - Ejecución 3:

- Valor obtenido:

```
========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 1 - Maximo local actual: 73694 - Maximo global: 73694


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 0 - Maximo local actual: 73694 - Maximo global: 97903


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 2 - Maximo local actual: 73694 - Maximo global: 81538


ID Thread: 0 - Maximo Local: 73694 - Maximo Global: 97903
ID Thread: 1 - Maximo Local: 73694 - Maximo Global: 73694
ID Thread: 2 - Maximo Local: 73694 - Maximo Global: 81538
```

- Valor esperado:

```
71027   97903   48387
73694   24759   47447
81538   2845    3704
```

Para este caso sería 97903

- Ejecución 4:
  - Valor obtenido:

```
========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 1 - Maximo local actual: 77424 - Maximo global: 103534


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 2 - Maximo local actual: 96801 - Maximo global: 96801


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 0 - Maximo local actual: 96801 - Maximo global: 77424


ID Thread: 1 - Maximo Local: 96801 - Maximo Global: 103534
ID Thread: 2 - Maximo Local: 96801 - Maximo Global: 96801
ID Thread: 0 - Maximo Local: 96801 - Maximo Global: 77424
```

  - Valor esperado:

```
62947   11710   77424
103534  16802   71694
88770   96801   79387
```

Para este caso sería 103534

- Ejecución 5:
  - Valor obtenido:

```
========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 2 - Maximo local actual: 59591 - Maximo global: 53539


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 0 - Maximo local actual: 59591 - Maximo global: 95169


========== NUEVO MÁXIMO ENCONTRADO =========
ID Thread: 1 - Maximo local actual: 59591 - Maximo global: 59591


ID Thread: 2 - Maximo Local: 59591 - Maximo Global: 53539
ID Thread: 1 - Maximo Local: 59591 - Maximo Global: 59591
ID Thread: 0 - Maximo Local: 59591 - Maximo Global: 95169
```

- Valor Esperado:

```
95169   39758   19869
45131   20290   59591
31401   53539   24132
```

Para este caso sería 95169

2. ¿Hay acceso concurrente a alguna variable compartida? Si es así, diga en dónde
   Sí, la concurrencia se da en la variable 'mayor'. Específicamente se da en la asignación del local y el mayor global, luego de su comparación para saber si vale la pena hacer el cambio de mayores. Cuando hay varios 'threads' al tiempo. Puede pasar que uno guarde (ejemplo) 10 como mayor local, compare, y vaya a hacer el cambio. Pero mientras espera para hacer el cambio digamos de 10 a 20. Un 'thread' con valor local de 15 llega y ver que es mayor que 10 (ya que no se ha hecho el cambio). Por tanto, luego de ambos cambios (primero de 10 a 20, y luego de 20 a 10; así este segundo ya no parezca lógico) el mayor queda descuadrado.

3. ¿Puede obtener alguna conclusión?
   La conclusión principal, al observar los tres ejemplos, es que cuando se usan 'threads' y hay información compartida normalmente está tiene problemas. Esto se debe a que no hay un orden en la modificación de la información compartida lo que causa que comparaciones y asignaciones puedan descuadrase fácilmente. Sin contar, que no hay forma prever el sistema que 'threads' tiene en uso y cuales no. Por tanto, se debe pensar en solucionar estos ejemplos por ejemplo volviendo ciertas partes 'synchronize'.