**Andres Takashi Bravo Yaguchi.**

**201911714.**

**ISIS-2203.**

**Laboratorio 1.**

**Taller 1:**

1. Complete la siguiente tabla, con respecto a la creación de Threads usando la extensión de la clase **Thread** y la implementación de la interface **Runnable.**

|  |  |
| --- | --- |
| **Se parecen** | **Se diferencian** |
| -El resultado final de ambas implementaciones es el mismo.  -Ambos implementan el método run (). | Para la implementación mediante la extensión de la clase Thread no es obligatorio implementar ningún método, en cambio, si se implementa la Interface Runnable es necesario implementar el método run(). Además, si se una clase A extiende la clase Thread es posible correr instancias de la clase A ya que esta hereda métodos de la clase Thread, por el contrario, para correr una instancia de una clase que implemente la Interface Runnable es necesario crear un objeto de la clase Thread a partir de la clase que implementa la Interface. |

**Taller 1b:**

**Parte 1: Incremento de un contador**

1. **¿Al ejecutar el programa, el resultado corresponde al valor esperado?**

Sí, tras ejecutar el programa se obtiene el resultado esperado que es 10’000.000. El valor esperado es 10 millones porque el programa suma 1000 veces 10.000 al contador.

1. **¿Al ejecutar el programa, el resultado corresponde al valor esperado? Explique.**

No, el resultado obtenido no es el esperado, pues se esperaba que el programa imprimiera en consola el numero 10’.000.000 pero tras ejecutar el programa el valor obtenido fue 999000. Esto ocurre por el uso de una variable compartida por todos los hilos de ejecución en el programa (ver punto 4).

1. **Ejecute cinco veces el programa y complete la tabla:**

|  |  |
| --- | --- |
| **Ejecución** | **Valor Obtenido** |
| 1 | 999000 |
| 2 | 999000 |
| 3 | 998000 |
| 4 | 998312 |
| 5 | 999000 |

1. **¿Hay acceso a alguna variable compartida? Si es así, diga en dónde.**

El acceso a la variable contador en el método run() es compartido por todos los hilos de ejecución del programa. Ya que el programa ejecuta 1000 hilos de ejecución simultáneamente, la probabilidad de que estos se superpongan es alta, esto hace que los hilos no se ejecuten en el orden en que fueron creados y alteren el contador. Sin embargo, es posible corregir este problema incluyendo una pausa (mediante el método Thread.sleep()) entre la ejecución de un Thread y la creación del siguiente, de esta manera se asegura que cada Thread se ejecute después de que el anterior haya terminado de modificar el contador.

**Parte 2: Elemento mayor en una matriz de enteros**

1. **Ejecute cinco veces el programa y escriba el resultado obtenido en cada ejecución:**

|  |  |  |
| --- | --- | --- |
| **Ejecución** | **Valor Obtenido** | **Valor Esperado** |
| **1** | 98010 | 98010 |
| **2** | 85858 | 93986 |
| **3** | 75643 | 99183 |
| **4** | 33370 | 70635 |
| **5** | 87694 | 93442 |

1. **¿Hay acceso concurrente a alguna variable compartida? Si es así, diga en dónde.**

Sí, el acceso a la variable que indica el máximo global es concurrente. Esto ocurre en el método Run() cuando se evalúa si el mayor número encontrado en el hilo de ejecución en curso es mayor que el máximo numero encontrado entre todos los hilos de ejecución, pues, es posible que un hilo de ejecución evalúe la condición y considere que el máximo hallado por este hilo es mayor que el máximo global actual, pero, que la variable del máximo global sea modificado por otro hilo de ejecución antes de que el hilo inicial termine de ejecutarse.

1. **¿Puede obtener alguna conclusión?**

Para garantizar que el resultado que se obtiene al ejecutar un programa multi-thread es necesario controlar el acceso a variables compartidas por los hilos de ejecución, pues como se pudo evidenciar a lo largo del taller, la no sincronización del acceso a las variables compartidas puede ocasionar accesos concurrentes a variables compartidas que alteran el resultado final del programa.