Juan Andrés Ariza Gacharná

201911442

Taller 1

1. Completen 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
A fin de cuentas, ambas implementaciones realizan la misma función que es la implementación de threads. Además, son similares en la implementación ya que en ambos se necesita la creación de un método run().	Si diferencian a la hora de crear un thread. Cuando se crea una clase que extiende a *Thread*, para crear un thread solo es necesario crear un objeto de la clase que extiende. Cuando se crea una clase que implementa *Runnable*, a la hora de crear un thread, es necesario crear un objeto tipo *Thread* que reciba como parámetro un objeto de la clase que implementa *Runnable*.

Taller 1b

Parte 1:

Ejemplo 1:

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

   El resultado que imprime es 10.000.000, el cual concuerda con el resultado esperado, ya que se está contando 1.000 veces 10.000 (1.000 * 10.000 = 10.000.000)

Ejemplo 2:

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

   No da el resultado esperado, ya que el resultado esperado era 10.000.000 por lo esperado en el punto anterior. No se obtiene el resultado esperado, ya que los threads no se ejecuntan necesariamente en orden, es decir, no es verdad que van a esperar que el thread anterior sume los 10.000 para empezar a sumar los suyos, porque es muy probable que los threads se superpongan entre sí y por lo tanto que ocurra un encuentro entre ellos y que sumen el mismo contador a la vez.

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

Ejecución	Valor Obtenido
1	9.775.501
2	9.646.768

3	9.623.258
4	9.600.472
5	9.614.567

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

   Si, el acceso ocurrente en la variable del contador, y ocurre en el método run() a la hora de aumentar el valor del contador.

Parte 2:

Ejemplo 3:

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

El valor que se toma en cuenta es el máximo global

Ejecución	Valor Obtenido	Valor Esperado
1	87.657	93.140
2	90.970	90.970
3	49.054	103.743
4	94.577	94.577
5	58.819	90.601

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

Si hay un acceso concurrente, esto ocurre en la variable que representa al mayor global de la matriz (*mayor*). Esto ocurre en el método run() cuando se realiza la comparación entre el mayor de la fila y el mayor global que lleva en ese momento. Porque ya que lo thread se superponen, puede que se realice una comparación válida, y a lo hora de entrar al condicional el valor de alguna variable cambie.

3. ¿Puede obtener alguna conclusión?

Se pueden crear threads que tengan tanto variables compartidas como variables propias de cada hilo. Esto se puede ver con el Ejemplo 2 (contador) y en el Ejemplo 3 (mayor) que todos los threads tienen variables compartidas. Además en estos ejemplos fue clara la necesidad de sincronizar los threads para controlar el acceso concurrente, ya que al no realizarse ninguna sincronización los resultados obtenidos en las distintas ejecuciones no son los esperados.