Camilo Falla Moreno – 201821059

Taller 1b

Parte 1

1. Al ejecutar el programa el resultado es 10.000.000, el cual es igual al esperado. Esto se da ya que solo hay un Thread el cual se ejecuta sin interrupción y sin concurrencia de ningún otro Thread.

2. Al ejecutar el programa el resultado no es igual al esperado. Esto se da ya que múltiples Threads están intentando acceder a la variable compartida "contador" concurrentemente, lo cual genera que cuando un Thread va a actualizar el valor de la variable, puede que otro ya lo haya hecho y, por lo tanto, ésta no se actualice el número total esperado de veces sino un poco menos. Esto se da ya que los Threads no siguen la regla FIFO (First In First Out) sino que intentan actualizar la variable de manera concurrente entre todos.

3.

Ejecución	Valor Obtenido
1	9980000
2	9970000
3	9990000
4	9990000
5	9980000

4.

El programa tiene acceso concurrente a la variable compartida "contador" por medio del método run(), en el cual la variable es incrementada en 10000 por cada Thread concurrentemente.

Parte 2

1.

Ejecución	Valor Obtenido	Valor Esperado
1	87256	102651
2	96248	96248
3	77304	86361
4	93657	105060
5	99859	100296

2.

El programa tiene acceso concurrente a la variable compartida "mayor" por medio del método run(), en el cual la variable es actualizada concurrentemente luego de que cada Thread encuentre el mayor en su fila.

3. En este método se puede ver cómo es que los Threads de una aplicación no siguen la regla FIFO (First In First Out) y por lo tanto corren en distintos órdenes. Esto genera que, al no seguir un orden secuencial, en algunas ocasiones la variable compartida “mayor” no tome el valor esperado. Esto se puede evidenciar en los resultados obtenidos al correr el programa múltiples veces.