

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

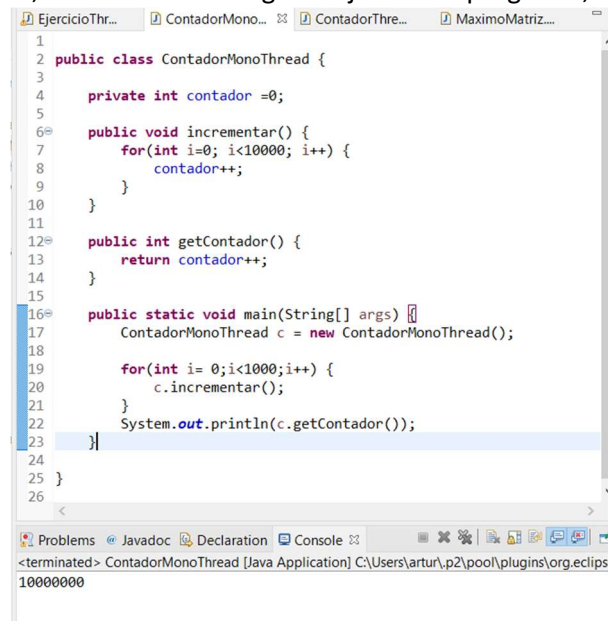
- Ambas implementaciones tienen el método run () el cual imprime un mensaje en consola.
- Tienen el método main ().
- Crean un objeto en el método main ().
- Ejecutan el método start () sobre el objeto previamente creado.

Se diferencian

- Uno implementa la interfaz Runnable y el otro extiende de la clase Thread.
- En el caso donde se extiende de Thread, se crea una instancia de la misma clase y sobre esa creación es que ejecuta el método start ().
- En el caso donde se implementa Runnable, se crea una instancia de la clase Thread y sobre esa creación es que ejecuta el método start ().

Taller 1b

1. ¿Al ejecutar el programa, el resultado corresponde al valor esperado?
Si, efectivamente luego de ejecutar el programa, retorna el valor 10000000




```
1 public class ContadorMonoThread {
2
3     private int contador =0;
4
5     public void incrementar() {
6         for(int i=0; i<10000; i++) {
7             contador++;
8         }
9     }
10
11     public int getContador() {
12         return contador++;
13     }
14 }
15
16 public static void main(String[] args) {
17     ContadorMonoThread c = new ContadorMonoThread();
18
19     for(int i= 0;i<1000;i++) {
20         c.incrementar();
21     }
22     System.out.println(c.getContador());
23 }
24
25 }
26
```

<terminated> ContadorMonoThread [Java Application] C:\Users\artur\p2\pool\plugins\org.eclipse

10000000

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

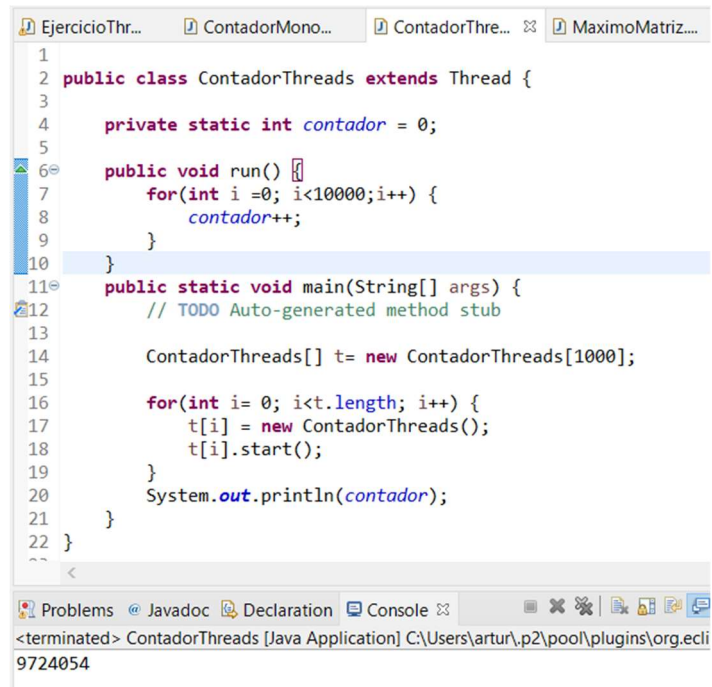
No, para este caso no se imprime el valor esperado, de hecho, para cada ejecución del programa se imprime un valor diferente, aunque siempre los resultados se encuentran muy cerca del valor obtenido.



```
1
2 public class ContadorThreads extends Thread {
3
4     private static int contador = 0;
5
6     public void run() {
7         for(int i=0; i<10000;i++) {
8             contador++;
9         }
10    }
11
12    public static void main(String[] args) {
13        // TODO Auto-generated method stub
14
15        ContadorThreads[] t= new ContadorThreads[1000];
16
17        for(int i= 0; i<t.length; i++) {
18            t[i] = new ContadorThreads();
19            t[i].start();
20        }
21        System.out.println(contador);
22    }
23 }
```

Problems @ Javadoc Declaration Console

<terminated> ContadorThreads [Java Application] C:\Users\artur\p2\pool\plugins\org.eclipse.justi
9521119



```
1
2 public class ContadorThreads extends Thread {
3
4     private static int contador = 0;
5
6     public void run() {
7         for(int i=0; i<10000;i++) {
8             contador++;
9         }
10    }
11
12    public static void main(String[] args) {
13        // TODO Auto-generated method stub
14
15        ContadorThreads[] t= new ContadorThreads[1000];
16
17        for(int i= 0; i<t.length; i++) {
18            t[i] = new ContadorThreads();
19            t[i].start();
20        }
21        System.out.println(contador);
22    }
23 }
```

Problems @ Javadoc Declaration Console

<terminated> ContadorThreads [Java Application] C:\Users\artur\p2\pool\plugins\org.ecli
9724054

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

Ejecución	Valor obtenido
1	9521119
2	9724054
3	9782616
4	9916549
5	9748562

4. ¿Hay acceso concurrente a alguna variable compartida? Si es así, diga en dónde Si, y esto es una posible respuesta a porqué no se está retornando el valor esperado. Esto debido a que la variable llamada 'contador' que es donde tenemos el registro, llega a sobrescribirse y no llega a sumar los valores exactamente las veces que se ejecutan.

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

Ejecución	Valor obtenido	Valor Esperado
1	101044	101044
2	59745	60278
3	89743	95735
4	49079	76475
5	101798	101798

Busqueda concurrente por una matriz

Matriz

```
=====
38820 86608 41927
65554 32929 80844
58979 88305 11947
```

Iniciando la busqueda por la matriz

```
===== Nuevo Maximo encontrado =====
ID Thread: 2 - Maximo local actual: 88305 - Maximo global: 88305
```

```
===== Nuevo Maximo encontrado =====
ID Thread: 1 - Maximo local actual: 88305 - Maximo global: 80844
```

```
===== Nuevo Maximo encontrado =====
ID Thread: 0 - Maximo local actual: 88305 - Maximo global: 86608
```

```
ID Thread: 2 - Maximo Local: 88305 - Maximo Global: 88305
ID Thread: 0 - Maximo Local: 86608 - Maximo Global: 88305
ID Thread: 1 - Maximo Local: 80844 - Maximo Global: 88305
```

Busqueda concurrente por una matriz

Matriz

```
=====
44349 104475 11570
9647 3757 8537
59585 105123 96132
```

Iniciando la busqueda por la matriz

```
===== Nuevo Maximo encontrado =====
ID Thread: 0 - Maximo local actual: 9647 - Maximo global: 104475
```

```
===== Nuevo Maximo encontrado =====
ID Thread: 2 - Maximo local actual: 9647 - Maximo global: 105123
```

```
===== Nuevo Maximo encontrado =====
ID Thread: 1 - Maximo local actual: 9647 - Maximo global: 9647
```

```
ID Thread: 2 - Maximo Local: 105123 - Maximo Global: 9647
ID Thread: 0 - Maximo Local: 104475 - Maximo Global: 9647
ID Thread: 1 - Maximo Local: 9647 - Maximo Global: 9647
```

6. ¿Hay acceso concurrente a alguna variable compartida? Si es así, diga en dónde. Si, hay un acceso concurrente de datos, este se encuentra en las variables que utilizamos para guardar el valor máximo, en este caso, las variables 'mayor' y 'mayorFila'.

7. ¿Puede obtener alguna conclusión?

La conclusión se encuentra en resaltar la importancia de un buen manejo de la concurrencia y el análisis del uso compartido de datos, ya que, si estos no son usados de la manera correcta y no se verifica que no se sobrescriban variables, el programa muy probablemente no va a dar los resultados correctos y esperados.

