Francisco de Borja Viera Prieto
Alexandru Dragomir

PRÁCTICA 1

1.

1.1- Si, ya que en la salida estándar se van intercambiando las ocurrencias de "Ejecutando Hebra Auxiliar" y "Ejecutando Hebra Principal"

1.2- No, ya que primero aparecen todas las ocurrencias de "Ejecutando Hebra Auxiliar" y luego las de "Ejecutando Hebra Principal"

2.

2.1- Si, ya que en la salida estándar se van intercambiando las ocurrencias de "Ejecutando Hebra Auxiliar" y "Ejecutando Hebra Principal"

2.2- No, ya que primero aparecen todas las ocurrencias de "Ejecutando Hebra Auxiliar" y luego las de "Ejecutando Hebra Principal"

3.

3.1- Variable de instancia

3.2- Utilizando el valor 0 en el constructor del objeto MiHebra

3.3-
```
MiHebra d = new MiHebra(1);
d.start();
```

3.4- La hebra principal se ejecuta de manera secuencial, y las hebras auxiliares se ejecutan de manera concurrente, esto lo sabemos porque la hebra principal imprime por pantalla todas sus salidas antes que las auxiliares, pero las auxiliares se ejecutan de manera concurrente, ya que van alternando sus salidas2.

Si, ambas imprimen sus variables miId

3.5- Es completamente secuencial, ya que muestra por pantalla las 100 llamadas a cada hebra de manera consecutiva.

4.

4.1- En el constructor le pasas el entero 0.

4.2- Se muestra el Id 0

4.3-

```
MiRun r1 = new MiRun(1);
Thread t1 = new Thread(r1);
t1.run();
```

4.4- Si, su comportamiento es similar, y ambas hebras imprimen sus respectivos Id

4.5- Si, sigue siendo secuencial

5.

5.1-

```
MiHebra t0 = new MiHebra(0, 1, 1_000_000);
MiHebra t1 = new MiHebra(1,1,1_000_000);
t0.start(); // Se inicia t0
t1.start();
```

5.2- Siempre finaliza antes la hebra principal.

Algunas veces finaliza antes t0, y otra finaliza antes t1.
Esto lo hemos visto al ejecutar varias veces el código.

5.3- Lo que ocurre es que la hebra que se ha marcado como daemon finalizara antes de llegar al final de se ejecución, por lo tanto no imprimirá el resultado por pantalla.

Si todas las hebras están marcadas como daemon, entonces solo imprimirá por pantalla la hebra principal.

5.4- No existe concurrencia, ya que solo hay una hebra iniciada en cada momento, ya que con el join(), el proceso main no inicia la otra hebra hasta que termine la ejecución de la anterior.
El resultado que se espera para nuestro programa será que primero finalice la hebra 1, a continuación, la 0 y después acaba la hebra principal.
Ahora vemos que en la salida del programa, esta todo ordenado por el orden de llamadas en el código.

La forma de resolver este problema de concurrencia es iniciar ambas hebras antes de llamar al join() en cada una de ellas.

No tiene sentido definir las hebras como daemon, ya que el proceso principal está esperando a que termine la ejecución de estas.

5.5-

Inicio:

```
for (int i =0; i< numHebras; i++){
 v[i] = new MiHebra(i,1,1_000_000);
 v[i].start();
}
```

Esperar finalizacion:

```
for (int i =0; i< numHebras; i++){
 try {
   v[i].join();
 }catch (InterruptedException e ){
   e.printStackTrace();
 }
}
```

5.6-

No, ya que habría problemas de atomicidad, ya que podrían llegar a modificar la variable al mismo tiempo, y la suma resultante no sería la esperable.

6.

6.1- No es interactiva, ya que al calcular el primo la interfaz ya no responde hasta que finaliza el calculo del primo.

6.2- Se tendría que crear una hebra dedicada al cálculo del número primo