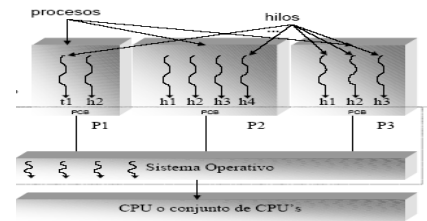




# THREADS

(Hilos en JAVA)



## Objetivos

- Emplear el lenguaje de programación Java para trabajar con conceptos básicos de programación concurrente.
- Crear y manipular procesos concurrentes en Java.
- Sincronizar hilos

## Ejercicio 1. Creación de un hilo

Escribir un programa concurrente que ejecute un hilo que imprima a pantalla su nombre y la hora a la que termino su ejecución.

Salida del programa:

```
run:
Nuevo hilo (19:16:36 07/10/2019)
Hilo principal termino a las 19:16:36
    Soy el hilo Thread-0
Termino Thread-0 a las (07:16:36)
```

## Ejercicio 2. Método sleep()

Modifique el programa anterior para que el hilo duerma un segundo y continúe trabajando sin parar. El hilo principal no espera a su hilo e imprime a pantalla que ha terminado su ejecución.

Usar el método sleep():

```
try {
    sleep(1000); //Segundo a segundo...
} catch (Exception e) {
    e.getMessage();
}
```

Salida del programa:

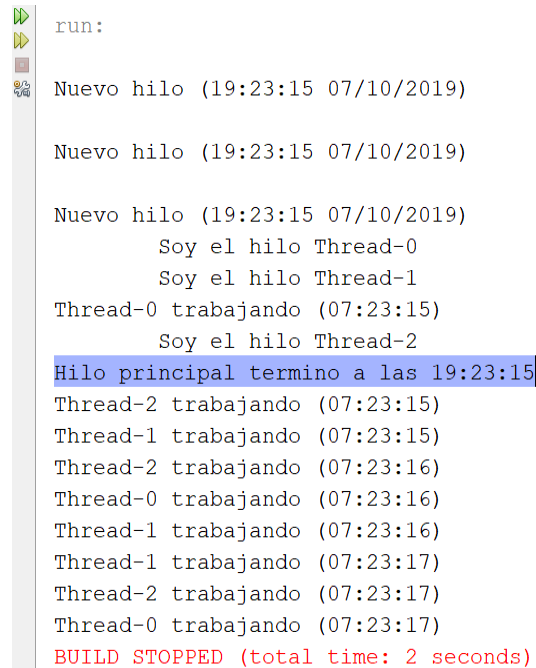
```
run:
Nuevo hilo (19:19:49 07/10/2019)
Hilo principal termino a las 19:19:49
    Soy el hilo Thread-0
Thread-0 trabajando (07:19:49)
Thread-0 trabajando (07:19:50)
Thread-0 trabajando (07:19:51)
Thread-0 trabajando (07:19:52)
Thread-0 trabajando (07:19:53)
Thread-0 trabajando (07:19:54)
BUILD STOPPED (total time: 5 seconds)
```

### Ejercicio 3. 3 hilos con sleep()

---

Modifique el programa anterior para que el hilo principal ponga a trabajar a tres hilos. Analice la salida del programa.

**Salida del programa:**



```
run:
Nuevo hilo (19:23:15 07/10/2019)
Nuevo hilo (19:23:15 07/10/2019)
Nuevo hilo (19:23:15 07/10/2019)
    Soy el hilo Thread-0
    Soy el hilo Thread-1
Thread-0 trabajando (07:23:15)
    Soy el hilo Thread-2
Hilo principal termino a las 19:23:15
Thread-2 trabajando (07:23:15)
Thread-1 trabajando (07:23:15)
Thread-2 trabajando (07:23:16)
Thread-0 trabajando (07:23:16)
Thread-1 trabajando (07:23:16)
Thread-1 trabajando (07:23:17)
Thread-2 trabajando (07:23:17)
Thread-0 trabajando (07:23:17)
BUILD STOPPED (total time: 2 seconds)
```

### Ejercicio 4. manipulación de las propiedades de los hilos

---

Escribir un programa concurrente que ejecute un hilo, el hilo deberá de imprimir a pantalla su nombre, su id y su prioridad, su grupo al que pertenece y su estado. El padre deberá de imprimir cuantos hijos están activos aún.

Hacer uso de los siguientes métodos:

- `getId()`
- `getThreadGroup()`
- `getState()`
- `Thread.activeCount()`

**Salida del programa:**

```
run:
Nuevo hilo (20:31:16 07/10/2019)
Nuevo hilo (20:31:16 07/10/2019)
Nuevo hilo (20:31:16 07/10/2019)
    Soy el hilo Thread-0, ID=11,prioridad=5, State=RUNNABLE, Group=java.lang.ThreadGroup[name=main,maxpri=10]
    Soy el hilo Thread-1, ID=12,prioridad=5, State=RUNNABLE, Group=java.lang.ThreadGroup[name=main,maxpri=10]
main: Hilos activos= 2
main: Hilo principal termino a las 20:31:16
    Soy el hilo Thread-2, ID=13,prioridad=5, State=RUNNABLE, Group=java.lang.ThreadGroup[name=main,maxpri=10]
```

### Ejercicio 5. manipulación de las propiedades de los hilos

---

Escribir un programa concurrente que ejecute 3 hilos cada uno imprima su nombre, su id, y su prioridad.

- El primer hilo se crea con los valores de default,
- El segundo hilo se crea y se le cambia su prioridad la máxima prioridad permitida.
- El tercer hilo se crea y se cambia su nombre y su prioridad por la mínima permitida.

### Ejercicio 6. Saludo Múltiple usando la clase Thread y el método sleep

---

Modificar el programa 3 para que cada hilo imprima su propio mensaje de saludo. El hilo deberá de dormir un determinado tiempo: para el primero duerme 4000, el segundo 2000 y el tercero 100. Deberá de imprimir la siguiente información:

Hola soy el Thread XX después de haber dormido: XXX tiempo.

Salida del programa en pantalla:

```
Saludo de la siguiente forma: Hola soy Thread 3 después de haber dormido: 100
Saludo de la siguiente forma: Hola soy Thread 2 después de haber dormido: 2000
Saludo de la siguiente forma: Hola soy Thread 1 después de haber dormido: 4000
```

### Ejercicio 7. Modificar el ejercicio anterior

---

Para que imprima el nombre del hilo en el momento que está en ejecución, antes de mándalo a dormir con sleep, y que la última instrucción del programa principal mande un aviso que termino su ejecución

Explique qué está sucediendo con el comportamiento de los hilos

Salida del programa en pantalla:

```
Termina el hilo principal....  
Soy el hilo Hola soy Thread 1  
Soy el hilo Hola soy Thread 2  
Soy el hilo Hola soy Thread 3  
Saludo de la siguiente forma: Hola soy Thread 3 después de haber dormido: 100  
Saludo de la siguiente forma: Hola soy Thread 2 después de haber dormido: 2000  
Saludo de la siguiente forma: Hola soy Thread 1 después de haber dormido: 4000
```

### **Ejercicio 8. Modificar el ejercicio anterior colocando un ciclo en el método run()**

---

Modifique el método run() para que implemente un ciclo de 4 iteraciones. Ejecute varias veces el programa observando las salidas.

Explique qué está sucediendo con el comportamiento de los hilos

Salida del programa en pantalla:

**Termina el hilo principal....**

**Soy el hilo Soy Thread 1  
Me voy a dormir Soy Thread 1**

**Soy el hilo Soy Thread 2**

**Soy el hilo Soy Thread 3  
Me voy a dormir Soy Thread 3  
Me voy a dormir Soy Thread 2  
Saludo de la siguiente forma: Soy Thread 3 después de haber dormido: 100**

**Soy el hilo Soy Thread 3  
Me voy a dormir Soy Thread 3  
Saludo de la siguiente forma: Soy Thread 3 después de haber dormido: 100**

**Soy el hilo Soy Thread 3  
Me voy a dormir Soy Thread 3  
Saludo de la siguiente forma: Soy Thread 3 después de haber dormido: 100**

**Soy el hilo Soy Thread 3  
Me voy a dormir Soy Thread 3  
Saludo de la siguiente forma: Soy Thread 3 después de haber dormido: 100  
.....termine Soy Thread 3.....**

**Saludo de la siguiente forma: Soy Thread 2 después de haber dormido: 2000**

**Soy el hilo Soy Thread 2  
Me voy a dormir Soy Thread 2  
Saludo de la siguiente forma: Soy Thread 1 después de haber dormido: 4000**

**Soy el hilo Soy Thread 1  
Me voy a dormir Soy Thread 1  
Saludo de la siguiente forma: Soy Thread 2 después de haber dormido: 2000**

**Soy el hilo Soy Thread 2  
Me voy a dormir Soy Thread 2  
Saludo de la siguiente forma: Soy Thread 2 después de haber dormido: 2000**

**Soy el hilo Soy Thread 2  
Me voy a dormir Soy Thread 2  
Saludo de la siguiente forma: Soy Thread 1 después de haber dormido: 4000  
Saludo de la siguiente forma: Soy Thread 2 después de haber dormido: 2000  
.....termine Soy Thread 2.....**

**Soy el hilo Soy Thread 1  
Me voy a dormir Soy Thread 1  
Saludo de la siguiente forma: Soy Thread 1 después de haber dormido: 4000**

**Soy el hilo Soy Thread 1  
Me voy a dormir Soy Thread 1  
Saludo de la siguiente forma: Soy Thread 1 después de haber dormido: 4000  
.....termine Soy Thread 1.....**

---

**Ejercicio 9. Modificar el ejercicio anterior colocando el mismo retardo para todos los hilos**

---

Nuevamente modifique el programa, duerma a todos los hilos el mismo tiempo de sleep. Ejecútelo varias veces observando las salidas.

Explique qué está sucediendo con el comportamiento de los hilos

---

**Ejercicio 10. Serie entera con un hilo**

---

Escribir un programa concurrente que ejecute un hilo, el hilo deberá imprimir una serie numérica que inicia siempre en el número 1 y termina en el número 10.

---

**Ejercicio 11. Series enteras con tres hilos**

---

Escribir un programa concurrente que ejecute tres hilos, cada hilo deberá imprimir su propia serie numérica. Todas las series inician en 1 y deberá de indicar el valor final de la misma, así como su incremento. Es decir, un hilo puede imprimir la serie del 2, otro la serie del 5 y el tercero la serie del 10.

Estos valores se deben preguntar al usuario: el tope de la serie y su incremento

**NOTA:** No olvidar que cuando diseñamos programas concurrentes crearemos un thread por cada uno de los procesos involucrados (el comportamiento de un thread está encapsulado en su correspondiente método run()).

---

**Ejercicio 12. Modificar nombre y prioridad**

---

Escribir un programa que implemente 3 hilos. Cada hilo deberá de modificar su nombre y su prioridad que tiene por default.

Consideraciones:

Deberá de pasar el nombre y la prioridad al momento de construir el hilo.

Hilo 1:

Nombre: "HiloA"

Prioridad: mínima prioridad permitida en los hilos

Hilo 2:

Nombre: "HiloB"

Prioridad: prioridad normal permitida en los hilos

Hilo 3:

Nombre: "HiloC"

Prioridad: Máxima prioridad permitida en los hilos

**Salida del programa:**

### **Ejercicio 13. Colocación de barreras con join**

---

**Modificar el ejercicio anterior y colocar 2 barreras, de la siguiente forma:**

1. Se ejecutan en paralelo los hilos A y B
2. Cuando terminen su ejecución los hilos A y B se ejecutará el hiloC, es decir el hilo C espera a los hilos A y B.
3. Y el hilo principal tendrá que esperar a que terminen los 3 hilos.

### **Ejercicio 14. Tablas de multiplicar con un hilo**

---

Escribir un programa concurrente que ejecute 1 hilo, el hilo deberá imprimir su nombre y todas las tablas de multiplicar del 1-10.

**Salida del programa en pantalla:**

### **Ejercicio 15. Tablas de multiplicar con 10 hilos**

---

Escribir un programa concurrente que ejecute 1 hilo por cada tabla de multiplicar, es decir un hilo para la tabla 1, un hilo para la tabla 2, etc. Cada hilo deberá de escribir su nombre y el producto

Salida del programa en pantalla:

### **Ejercicio 16. Tablas de multiplicar con 100 hilos**

---

Escribir un programa concurrente que ejecute 100 hilo uno por cada producto de las 10 tablas de multiplicar, es decir un hilo para el producto 1 x 1, otro hilo para el producto 1x2, .... Un hilo para el producto 10x10. Cada hilo deberá de escribir su nombre y el producto

Salida del programa en pantalla:

### **Ejercicio 17. Hilos multiples con barreras en el main**

---

Modificar el “programa 15 tablas de multiplicar con 10 hilos” para que el main espere a que terminen su ejecución todos sus “hilos tabla”. El hilo padre debe de imprimir a pantalla:

(Tiempo) Hilos activos: ##

(Tiempo) el hilo <nombre\_del\_hilo> terminó su ejecución.

---

**Ejercicio 18. 10 Hilos tablas con 10 hilos producto con barreras en el main y en cada hilo tabla**

---

Modificar el programa anterior “tablas de multiplicar con 10 hilos y barrera en el main” para que cada hilo ejecute 10 hilos a su vez (10 hilos producto) SIN esperar a estos hilos y cada uno de estos hilos imprimirá su producto. Imprimir a pantalla el hilo X terminó su ejecución.

**Salida a pantalla:**

<pre>run: Soy Tabla1: estoy creando mis 10 hilos Soy Tabla2: estoy creando mis 10 hilos Soy Tabla3: estoy creando mis 10 hilos Soy Tabla4: estoy creando mis 10 hilos .....Tabla1: termino Soy Tabla5: estoy creando mis 10 hilos .....Tabla5: termino .....Tabla2: termino Soy Tabla6: estoy creando mis 10 hilos .....Tabla3: termino Soy Tabla9: estoy creando mis 10 hilos Thread-10: 1*1= 1 Soy Tabla7: estoy creando mis 10 hilos .....Tabla4: termino .....Tabla9: termino .....Tabla6: termino Soy Tabla10: estoy creando mis 10 hilos</pre>	<pre>Soy Tabla7: estoy creando mis 10 hilos .....Tabla4: termino .....Tabla9: termino .....Tabla6: termino Soy Tabla10: estoy creando mis 10 hilos .....Tabla7: termino Soy Tabla8: estoy creando mis 10 hilos .....Tabla10: termino Thread-11: 2*1= 2 .....Tabla8: termino Termino el main Thread-12: 1*2= 2 Thread-14: 1*3= 3 Thread-13: 2*2= 4 Thread-16: 2*3= 6 Thread-18: 2*4= 8 Thread-21: 2*6= 12 Thread-19: 1*6= 6</pre>
--	--