


# Métodos Avanzados de Programación Científica y Computación

M<sup>a</sup> Luisa Díez Platas

## Tema 7. Introducción a la programación concurrente

# ¿Cómo estudiar este tema?

IDEAS CLAVE	LO + RECOMENDADO	+ INFORMACIÓN	TEST
<p>¿Cómo estudiar este tema?</p> <p>Introducción a la programación concurrente</p> <p>El concepto de proceso e hilo</p> <p>Interacción entre procesos o hilos</p> <p>Los hilos en Java</p> <p>Ventajas e inconvenientes de la programación concurrente</p> <p>Computación de alto rendimiento</p>	<p><b>No dejes de leer...</b></p> <p>Concurrencia</p> <p><b>No dejes de ver...</b></p> <p><b>TV</b> Hilos/Threads en Java</p> <p><b>TV</b> Java-Threads (hilos) implementados CON Runnable</p>	<p><b>A fondo</b></p> <p>Programación concurrente</p> <p><b>Bibliografía</b></p>	

# Conceptos básicos

No siempre es real

Concurrencia-ejecución simultánea de conjuntos de instrucciones que guardan cierta independencia.

## Entidades concurrentes

- **Proceso (unidad de asignación de recursos)**
  - Entidad de ejecución independiente
  - Cuenta con espacio de direcciones o registro de activación (cambio de contexto caro)
  - Unidad básica en entorno multitarea
- **Hilo (unidad de planificación independiente)**
  - Entidad de ejecución independiente
  - Tienen espacios de memoria compartido con otros hilos (mismo proceso “pesado”)
  - Unidad básica en entorno multihilo

Programación concurrente-paradigma de programación que permite la creación de programas con ejecución simultánea de múltiples tareas.

# Conceptos básicos

**Multiprogramación** → gestión de procesos en un sistema monoprocesador.

**Multiprocesamiento** → gestión de procesos en un sistema multiprocesador (puede existir memoria común).

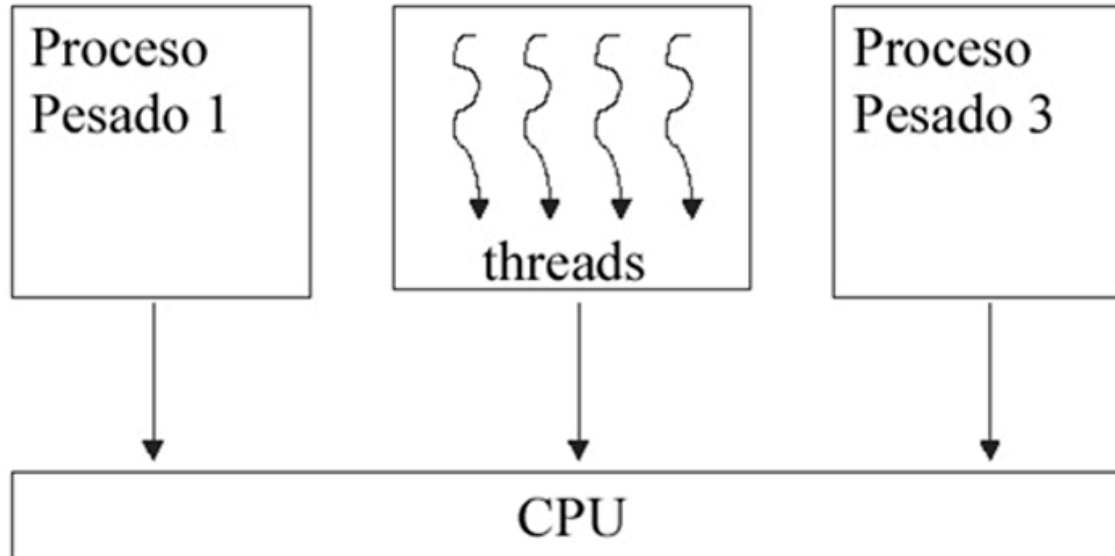
**Procesamiento distribuido** → gestión de procesos en un procesadores separados (memoria no compartida).

**Programación concurrente** → acciones que pueden ser ejecutadas de forma simultanea.

**Programación paralela** → programación concurrente en un sistema multiprocesador.

**Programación distribuida** → programación paralela en un sistema distribuido.

# Conceptos básicos



# Interacción entre procesos o hilos

## Competencia por recursos

Independientes

→ Problemas más importantes: accesos a sección crítica (exclusión mutua), interbloqueo o inanición.

## Cooperación para compartir recursos-


Comparten recursos

Se comunican

→ Problemas más importantes: interbloqueo o inanición.


# Problemas en programación concurrente

**Problemas de seguridad-** Nunca sucede nada malo a un objeto

- Conflictos de lectura/escritura
  - Conflictos de escritura/escritura
- 
- Técnicas de exclusión
  - Garantizar la atomicidad de las acciones

**Problemas de vivacidad-** Algo sucede eventualmente durante una actividad

- Interbloqueo
- Señales perdidas
- Cerrojos anidados en un monitor
- Falta de vivacidad
- Inanición
- Falta de recursos

- 
- Bloqueo
  - Espera
  - Entrada
  - Competencia por CPU
  - Fallo

# Hilos en Java

Los hilos son objetos de la clase **Thread**.

- ▶ Está en el paquete ***java.lang.Thread***.
- ▶ Para declarar un Thread:  
Thread trabajador;  
trabajador = new Thread ();
- ▶ O en una sola línea:  
Thread trabajador = new Thread();

Existen 2 formas de trabajar con Threads:

- ▶ **Heredando** de la clase ***Thread*** y redefiniendo el método run.
- ▶ **Implementando** la interfaz ***Runnable***.



## Hilos en Java. Heredando de la clase *Thread* y redefiniendo el método run.

```
class HiloThread extends Thread {  
    private String palabra_;  
    private int intervalo_;  
  
    // constructor HiloThread (String, int)  
    public HiloThread (String palabra, int intervalo) {  
        palabra_ = palabra;  
        intervalo_ = intervalo;  
    }  
  
    // se redefine el método run ()  
    public void run() {  
        try {  
            for (int i = 0; i < 10; i++) {  
                System.out.print (palabra_ + " ");  
                java.lang.Thread.sleep (intervalo_); // puede elevar una excepción  
            } // for  
        } // try  
        catch (InterruptedException e) {  
            System.err.println ("Se ha producido un error: " + e.toString ());  
        } // catch  
    } // run ()  
}
```

# Hilos en Java. Heredando de la clase *Thread* y redefiniendo el método *run*.

```
public static void main (java.lang.String[] args) {  
    new HiloThread ("ton",33).start();  
    new HiloThread ("tin",100).start();  
} // main
```

```
} // HiloThread
```

```
public static void main (java.lang.String[] args) {  
    HiloThread a = new HiloThread ("ton",33); // usando Thread  
    HiloThread b= new HiloThread ("tin",100); // con HiloThread  
  
    a.start();  
    b.start();  
} // main  
} // HiloThread
```

O



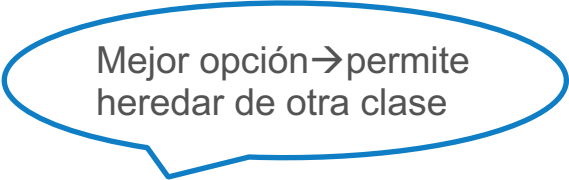
**Salida:**

**ton tin ton ton ton tin ton ton ton tin ton ton tin tin tin tin tin tin**

# Hilos en Java. Implementando la interfaz *Runnable*.

```
import java.lang.Thread;

class HiloRunnable implements Runnable {
    private String palabra_;
    private int intervalo_;
    // constructor
    public HiloRunnable (String palabra, int intervalo) {
        palabra_ = palabra;
        intervalo_ = intervalo;
    } // HiloRunnable
    // redefinición del método run ()
    public void run () {
        try {
            for (int i = 0; i < 10; i++) {
                System.out.print (palabra_ + " ");
                java.lang.Thread.sleep (intervalo_);
            } // for
        } // try
        catch (InterruptedException e) {
            System.err.println ("Se ha producido un error" + e.toString ());
        } // catch
    } // run ()
}
```



Mejor opción → permite heredar de otra clase

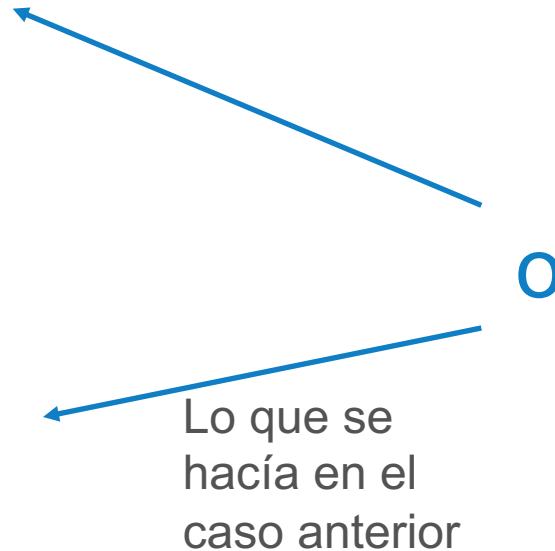
# Hilos en Java. Implementando la interfaz *Runnable*.

```
public static void main (String [] args) {  
    HiloRunnable ton = new HiloRunnable ("ton", 33);  
    HiloRunnable tin = new HiloRunnable ("tin", 100);  
    Thread h1 = new Thread (ton);  
    Thread h2 = new Thread (tin);  
    h1.start ();  
    h2.start ();  
} // main ()  
} // HiloRunnable
```

```
public static void main (java.lang.String[] args) {  
  
    new HiloThread ("ton",33).start();  
  
    new HiloThread ("tin",100).start();  
  
} // main  
} // HiloThread
```

**Salida:**

**ton tin ton ton ton tin ton ton ton tin ton ton tin tin tin tin tin tin**



## Hilos en Java. Implementando la interfaz *Runnable*. (otro ejemplo)

```
class OtroHiloRunnable implements Runnable {  
    // constructor que no hace nada  
    public OtroHiloRunnable () {}  
    public void run () {  
        for (int i = 0; i < 10; i++) {  
            try {  
                System.out.println ("Estoy dentro de OtroHiloRunnable.run()");  
                java.lang.Thread.sleep (7);  
            } // try  
            catch (InterruptedException e) {  
                System.err.println ("Se ha producido un erro: " + e.toString ());  
            } // catch  
        } // for  
    } // run ()  
}
```

# Hilos en Java. Implementando la interfaz *Runnable*. (otro ejemplo)

```
public static void main (String[] args) {  
    OtroHiloRunnable otroHilo = new OtroHiloRunnable (); // primero creamos el objeto  
    Thread hilo = new Thread (otroHilo); // luego el Thread contenedor  
    hilo.start();  
    for (int i = 0; i < 10; i++) {  
        try {  
            System.out.println ("Dentro del método main (");  
            java.lang.Thread.sleep (10);  
        } // try  
        catch (InterruptedException e) {  
            System.err.println ("Se ha producido un error: " + e.toString ());  
        } // catch  
    } // for  
} // main (String [])  
} // OtroHiloRunnable
```

```
Dentro del método main ()  
Estoy dentro de OtroHiloRunnable.run()  
Estoy dentro de OtroHiloRunnable.run()  
Dentro del método main ()  
Estoy dentro de OtroHiloRunnable.run()  
Dentro del método main ()  
Estoy dentro de OtroHiloRunnable.run()  
Estoy dentro de OtroHiloRunnable.run()  
Dentro del método main ()  
Estoy dentro de OtroHiloRunnable.run()  
Dentro del método main ()  
Estoy dentro de OtroHiloRunnable.run()  
Estoy dentro de OtroHiloRunnable.run()  
Dentro del método main ()  
Estoy dentro de OtroHiloRunnable.run()  
Dentro del método main ()  
Dentro del método main ()  
Dentro del método main ()
```

# Hilos en Java. Objeto autónomo

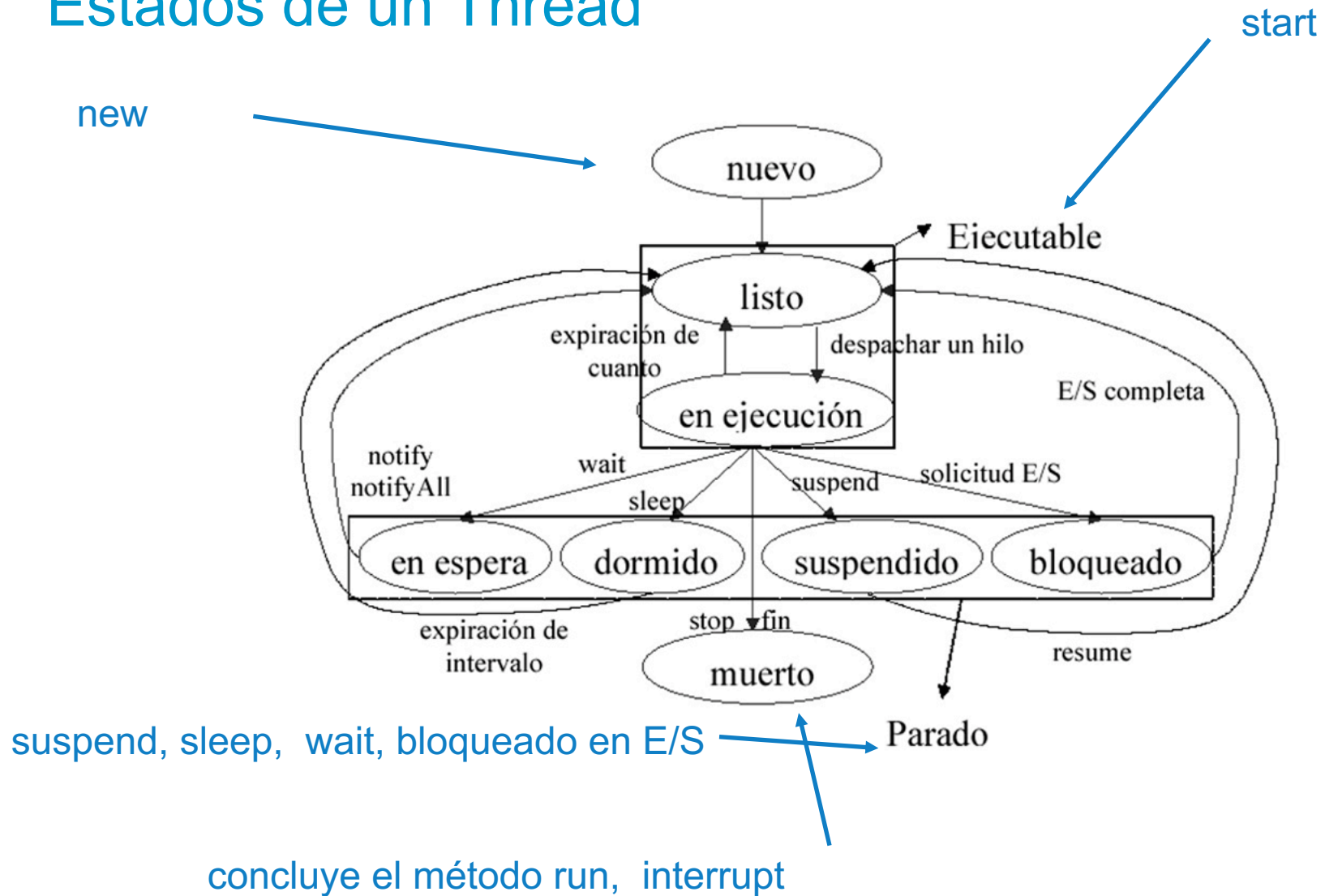
```
class AutoHiloRunnable implements Runnable {  
    private Thread yo_;  
    public AutoHiloRunnable () {  
        yo=new Thread(this);  
        yo.start();  
    }  
  
    public void run () {  
        if (yo_ == Thread.currentThread ()) {  
            for (int i = 0; i < 10; i++) {  
                System.out.println ("Estoy dentro de AutoHiloRunnable.run ()");  
                java.lang.Thread.yield();  
            } // for  
        } // if  
    } // run ()  
  
    public static void main (String [] args) {  
        AutoHiloRunnable unHilo = new AutoHiloRunnable ();  
        for (int i = 0; i < 10; i++) {  
            System.out.println ("Dentro del método main ()");  
            java.lang.Thread.yield ();  
        } // for  
    } // main ()  
} // AutoHiloRunnable
```



Se ejecuta a sí mismo

```
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()  
Dentro del método main ()  
Estoy dentro de AutoHiloRunnable.run ()
```

# Estados de un Thread





UNIVERSIDAD  
INTERNACIONAL  
DE LA RIOJA

**unir**

[www.unir.net](http://www.unir.net)