

ut2.

Programación multihilo

PSP

Introducción

La ejecución de tareas en paralelo optimiza la utilización de los recursos del sistema

Ejemplo: cuando un proceso está esperando la finalización de una operación de E/S, otros procesos pueden aprovechar el procesador del sistema que en ese momento no se usa.

Programación paralela mucho más compleja que la convencional.

Procesos e hilos

Procesos

- Disponen de su propio espacio de memoria.
- Se comunican a través de pipes o sockets.

Hilos

- Ejecuciones simultáneas dentro de un mismo proceso (podemos considerarlos como “procesos ligeros”)
- Espacio de memoria compartido

Hilos

Hilos no pueden ejecutarse en solitario
necesitan un proceso.

Utilidad hilos: Realizar programas que tengan
realizar varias tareas simultáneamente.

Ejemplo en fabrica programa controla sensores
cada sensor hilo, procesador gráfico hilo
corrige ortografía hilo guarda fichero...

Hilos en Java

Cuando se inicia un programa en Java, la máquina virtual crea un hilo principal.

- El hilo se encargará de invocar al método main de la clase que se comienza a ejecutar.
- El hilo termina cuando se acaba de ejecutar el método main.
- Si el hilo principal crea otros hilos, éstos comenzarán su ejecución de forma concurrente.
- Sólo cuando no queda ningún hilo activo, es cuando se termina el programa

Thread

La clase principal para conseguir concurrencia en Java es la clase Thread.

Dispone de un método `start()` que ocasiona la ejecución del código que tenga dentro de su método `run()` en un nuevo hilo

Creación de hilos: 2 alternativas

1. Heredando de la clase Thread.
2. Implementando la interfaz Runnable

Creación de hilos. Thread

Esquema general

1. Extender clase de la clase Thread (crear subclase)
2. Se implementa el método run cuyo código define lo que va a hacer el hilo durante su ejecución
3. Se crea el hilo y se llama a su método start , que se encarga, entre otras cosas, de llamar a run

Creación de hilos. Thread

Estructura básica

```
//estructura básica hilo
Class NombreHilo extends Thread {
    //propiedades, constructores y métodos clase

    public void run(){
        //acciones que lleva a cabo el hilo
    }
}
```

Creación hilo. Thread. PrimerHilo

```
//clase PrimerHilo
public class PrimerHilo extends Thread {

    //propiedades clase
    private int x;

    //constructor clase
    PrimerHilo(int x)
    {
        this.x=x;
    }

    //método ejecución run
    public void run(){
        //acciones que lleva a cabo el hilo
        for (int i=0;i<x;i++)
            System.out.println("En el hilo..");
    }

    //Método main
    public static void main(String[] args){

        PrimerHilo p=new PrimerHilo(10);//creo hilo
        p.start();//iniciar hilo

    } //fin main
}
```

```
david@david-OEM:~/Escritorio/psp/ut2$ javac PrimerHilo.java
david@david-OEM:~/Escritorio/psp/ut2$ java PrimerHilo
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
En el hilo..
david@david-OEM:~/Escritorio/psp/ut2$
```

- Fijate extiende de Thread
- Después definimos propiedades de la clase y constructor.
- Método run

Creación hilo. HiloEjemplo1.java. 1/2

```
public class HiloEjemplo1 extends Thread {  
  
    //Propiedades -----  
    private int c; //contador hilo  
    private int hilo;  
  
    //Constructor -----  
    public HiloEjemplo1 (int hilo){  
        this.hilo=hilo;  
        System.out.println("Creando Hilo: " + hilo);  
    }//fin constructor  
  
    //Método Run -----  
    public void run(){  
        c=0;  
        while (c<=5){  
            System.out.println ("Hilo:" + hilo + " C= " + c);  
            c++;  
        }  
    }//fin run
```

Fijate extiende de Thread

Después definimos propiedades de la clase y constructor.

Método run

Creación hilo. HiloEjemplo1.java. 1/2

```
//Método main
public static void main(String[] args){
    HiloEjemplo1 h=null;
    for (int i=0;i<3;i++){
        h=new HiloEjemplo1(i+1); //creo hilo
        h.start(); //iniciar hilo
    }
    System.out.println("3 Hilos creados...");
} //fin main

} //fin clase
```

Muy usual en el run del hilo tener un bucle infinito de forma que el hilo no termina a no ser utilizemos método que veremos.

Método main llama a la creación de hilo. Observa como realiza bucle..y en cada llamada crea hilo y llama al método start para iniciarlo.

```
david@david-OEM ~/pss/ut2 $ javac HiloEjemplo1.java
david@david-OEM ~/pss/ut2 $ java HiloEjemplo1
Creando Hilo: 1
Creando Hilo: 2
Hilo:1 C= 0
Hilo:2 C= 0
Hilo:2 C= 1
Hilo:2 C= 2
Hilo:2 C= 3
Hilo:2 C= 4
Hilo:2 C= 5
Creando Hilo: 3
Hilo:1 C= 1
Hilo:1 C= 2
Hilo:1 C= 3
Hilo:1 C= 4
Hilo:1 C= 5
Hilo:3 C= 0
Hilo:3 C= 1
Hilo:3 C= 2
Hilo:3 C= 3
3 Hilos creados...
Hilo:3 C= 4
Hilo:3 C= 5
```

hilos no se ejecutan en orden creación!!!

Código HiloEjemplo1.java. 1/2

```
public class HiloEjemplo1 extends Thread {
    //Propiedades -----
    private int c; //contador hilo
    private int hilo;

    //Constructor -----
    public HiloEjemplo1 (int hilo){
        this.hilo=hilo;
        System.out.println("Creando Hilo: " + hilo);
    }//fin constructor

    //Método Run -----
    public void run(){
        c=0;
        while (c<=5){
            System.out.println ("Hilo:"+ hilo + " C= " + c);
            c++;
        }
    }//fin run
```

Código HiloEjemplo1.java. 2/2

```
//Método main
public static void main(String[] args){
    HiloEjemplo1 h=null;
    for (int i=0;i<3;i++){
        h=new HiloEjemplo1(i+1); //creo hilo
        h.start(); //iniciar hilo
    }
    System.out.println("3 Hilos creados...");
} //fin main

} //fin clase
```

Llamada a hilo desde otra clase

En ejemplo anterior el main incluido dentro de la clase. Pero podemos tener clase hilo y clase llama a clase hilo.

Modificamos el ejemplo:

```
public class HiloEjemplo1_V2 extends Thread {
    //Propiedades -----
    private int c; //contador hilo
    private int hilo;

    //Constructor -----
    public HiloEjemplo1_V2( int hilo){
        this.hilo=hilo;
        System.out.println("Creando Hilo: " + hilo);
    }//fin constructor

    //Método Run -----
    public void run(){
        c=0;
        while (c<=5){
            System.out.println ("Hilo:"+ hilo + " C= " + c);
            c++;
        }
    }//fin run

} //fin clase
```

Clase llama al hilo

```
public class UsaHiloEjemplo1_V2 {  
    public static void main(String[] args){  
        HiloEjemplo1_V2 h=null;  
        for (int i=0;i<3;i++){  
            h=new HiloEjemplo1_V2(i+1); //creo hilo  
            h.start(); //iniciar hilo  
        }  
        System.out.println("3 Hilos creados...");  
    } //fin main  
} //clase usa hilo
```

```
david@david-OEM ~/pss/ut2 $ javac HiloEjemplo1_V2.java  
david@david-OEM ~/pss/ut2 $ javac UsaHiloEjemplo1_V2.java  
david@david-OEM ~/pss/ut2 $ java UsaHiloEjemplo1_V2  
Creando Hilo: 1  
Creando Hilo: 2  
Hilo:1 C= 0  
Hilo:1 C= 1  
Hilo:1 C= 2  
Hilo:1 C= 3  
Hilo:1 C= 4  
Hilo:1 C= 5  
Creando Hilo: 3  
Hilo:2 C= 0  
Hilo:2 C= 1  
Hilo:2 C= 2  
Hilo:2 C= 3  
Hilo:2 C= 4  
Hilo:2 C= 5  
3 Hilos creados...  
Hilo:3 C= 0  
Hilo:3 C= 1  
Hilo:3 C= 2  
Hilo:3 C= 3  
Hilo:3 C= 4  
Hilo:3 C= 5
```


Métodos útiles en hilos

start() → Hace que el hilo comience la ejecución.

boolean isAlive() → Comprueba si el hilo está vivo. Devuelve True si está vivo.

join() → Espera que termine el hilo.

getState() → *Devuelve el estado del hilo:* NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, TERMINATED

Ejemplo: Métodos en uso

Creamos una
clase hilo
simple
(HiloEjemplo_A
live)

```
public class HiloEjemplo_Alive extends Thread {  
  
    //Propiedades -----  
    private int c; //contador hilo  
    private int hilo;  
  
    //Constructor -----  
    public HiloEjemplo_Alive (int hilo){  
        this.hilo=hilo;  
        System.out.println("Creando Hilo: " + hilo);  
    }//fin constructor  
|  
    //Método Run -----  
    public void run(){  
        c=0;  
        while (c<=5){  
            System.out.println ("Hilo:"+ hilo + " C= " + c);  
            c++;  
        }  
    }//fin run  
  
} //fin clase
```

Ejemplo: Métodos en uso

```
import java.lang.*;
public class UsaHilo_Alive {
    public static void main(String[] args){
        HiloEjemplo_Alive h=null;
        h=new HiloEjemplo_Alive(1); //creo hilo

        //Compruebo estado antes llamar a start
        System.out.println("Antes llamada a start");
        System.out.println("Is Alive? = " + h.isAlive());
        System.out.println("State:" + h.getState());

        System.out.println("llamo a start");
        h.start(); //iniciar hilo

        //Compruebo estado tras llamar a start
        System.out.println("State:" + h.getState());
        System.out.println("Is Alive? = " + h.isAlive());

        // llamo a join y espero termine
        try{
            h.join();
        } catch (Exception ex){}

        // Compruebo estado tras finalizar hilo
        System.out.println("Tras finalizar hilo ");
        System.out.println("State:" + h.getState());
        System.out.println("Is Alive? = " + h.isAlive());
    } //fin main
} //clase usa hilo
```

Ejemplo: UsaHilo_Alive editable

```
import java.lang.*;
public class UsaHilo_Alive {
    public static void main(String[] args){
        HiloEjemplo_Alive h=null;
        h=new HiloEjemplo_Alive(1); //creo hilo

        //Compruebo estado antes llamar a start
        System.out.println("Antes llamada a start");
        System.out.println("Is Alive? = " + h.isAlive());
        System.out.println("State:" + h.getState());

        System.out.println("llamo a start");
        h.start(); //iniciar hilo

        //Compruebo estado tras llamar a start
        System.out.println("State:" + h.getState());
        System.out.println("Is Alive? = " + h.isAlive());

        // llamo a join y espero termine
        try{
            h.join();
        } catch (Exception ex){

        }

        // Compruebo estado tras finalizar hilo
        System.out.println("Tras finalizar hilo ");
        System.out.println("State:" + h.getState());
        System.out.println("Is Alive? = " + h.isAlive());
    } //fin main
} //clase usa hilo
```

```
david@david-OEM ~/pss/ut2 $ java UsaHilo_Alive
Creando Hilo: 1
Antes llamada a start
Is Alive? = false
State:NEW
llamo a start
State:RUNNABLE
Is Alive? = true
Hilo:1 C= 0
Hilo:1 C= 1
Hilo:1 C= 2
Hilo:1 C= 3
Hilo:1 C= 4
Hilo:1 C= 5
Tras finalizar hilo
State:TERMINATED
Is Alive? = false
```

Métodos útiles en hilos

sleep(long mils) → Hace que el hilo en ejecución pase a dormir durante milisegundos especificados

toString() → Devuelve una representación en formato cadena de este hilo, incluyendo nombre del hilo, la prioridad y el grupo de hilos.

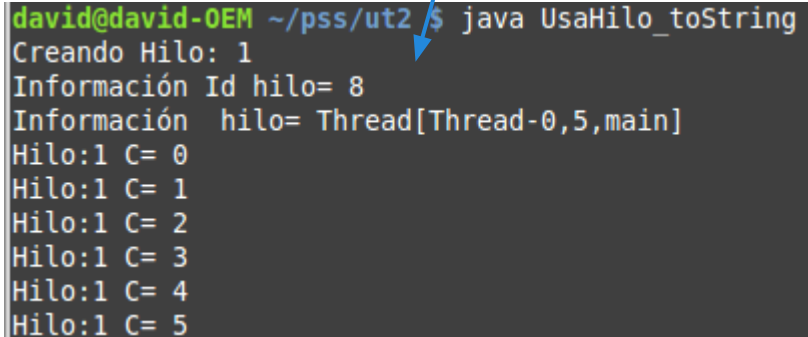
getId() → Devuelve el identificador del hilo

Ejemplo: Métodos en uso

```
public class UsaHilo_toString {  
    public static void main(String[] args){  
        HiloEjemplo_Alive h=null;  
        h=new HiloEjemplo_Alive(1); //creo hilo  
  
        h.start(); //iniciar hilo  
  
        System.out.println ("Información Id hilo= " + h.getId());  
        System.out.println ("Información hilo= " + h.toString());  
    } //fin main  
}  
//clase usa hilos
```

Thread[nombre hilo, prioridad, grupo de hilos]

id del hilo



```
david@david-OEM ~/pss/ut2 $ java UsaHilo_toString  
Creando Hilo: 1  
Información Id hilo= 8  
Información hilo= Thread[Thread-0,5,main]  
Hilo:1 C= 0  
Hilo:1 C= 1  
Hilo:1 C= 2  
Hilo:1 C= 3  
Hilo:1 C= 4  
Hilo:1 C= 5
```

Ejemplo: Métodos en uso. sleep

```
public class UsaHilo_Sleep {  
    public static void main(String[] args){  
        HiloEjemplo_Alive h=null;  
        h=new HiloEjemplo_Alive(1); //creo hilo  
        h.start(); //iniciar hilo  
  
        try {  
            // Pause for 4 seconds  
            h.sleep(4000);  
        } catch (InterruptedException e) {  
  
        } //fin main  
    }  
} //clase usa hilo
```

Métodos útiles en hilos

getName() → Devuelve nombre del hilo

setName(String name) → Cambia el nombre y asigna name.

getPriority() → Devuelve prioridad del hilo

setPriority(int p) → Cambia la prioridad a p

Métodos útiles en hilos

void interrupt() → Interrumpe ejecución hilo

boolean interrupted() → Comprueba si hilo actual ha sido interrumpido.

Thread currentThread() → Devuelve referencia al objeto hilo que se está ejecutando actualmente.

Ejemplo HiloEjemplo2

```
public class HiloEjemplo2 extends Thread {  
  
    //Metodo run  
    public void run(){  
        System.out.println ("Dentro del Hilo: " + this.getName() + "Prioridad: " + this.getPriority() + "ID: " + this.getId());  
    }//fin run  
  
    //Metodo main  
    public static void main (String[] args){  
  
        HiloEjemplo2 h=null;  
  
        for (int i=0;i<3; i++){  
            h= new HiloEjemplo2(); //creo hilo  
  
            h.setName ("HILO" + i); // establezo nombre  
            h.setPriority (i+1); // establezco prioridad  
  
            h.start(); // inicio hilo  
  
            System.out.println ("Información del " + h.getName() + ": " + h.toString());  
        }  
        System.out.println ("3 Hilos creados...");  
  
    }//fin main  
}  
//fin hilo
```

Ejemplo HiloEjemplo2. Editable

```
public class HiloEjemplo2 extends Thread {
    //Metodo run
    public void run(){
        System.out.println ("Dentro del Hilo: " + this.getName() + "Prioridad: " + this.getPriority() + "ID: " + this.getId());
    } //fin run

    //Metodo main
    public static void main (String[] args){

        HiloEjemplo2 h=null;

        for (int i=0;i<3; i++){
            h= new HiloEjemplo2(); //creo hilo

            h.setName ("HILO" + i); // establezo nombre
            h.setPriority (i+1); // establezco prioridad

            h.start(); // inicio hilo

            System.out.println ("Información del " + h.getName() + ": " + h.toString());

            System.out.println ("3 Hilos creados...");

        }

    } //fin hilo
}
```

```
david@david-OEM ~/pss/ut2 $ java HiloEjemplo2
Información del HIL00: Thread[HIL00,1,main]
Dentro del Hilo: HIL00Prioridad: 1ID: 8
Información del HIL01: Thread[HIL01,2,main]
Dentro del Hilo: HIL01Prioridad: 2ID: 9
Información del HIL02: Thread[HIL02,3,main]
3 Hilos creados...
Dentro del Hilo: HIL02Prioridad: 3ID: 10
```

Actividad 2.1

Crea dos clases (hilos). Uno de los hilos debe visualizar un bucle infinito la palabra TIC y el otro TAC. Dentro del bucle utiliza el método `sleep()` para que nos permita visualizar las palabras cuando ejecutemos.

Crea main para que haga uso de hilos anteriores ¿Se visualiza TIC TAC de forma ordenada (es decir TICTACTICTAC)?