

Presentación

Programación concurrente (PC)

Area: Programación Especializada

Correlativas:

- Introducción a la Computación
- Programación Orientada a Objetos

Docentes:

- Silvia Amaro
- Marcela Leiva,
- Valeria Zoratto

Horarios: 4 hs semanales

- Teoría: Lunes 11:15 – 14:00 hs.
- Práctica: Viernes 16.00 -18:00 hs.
- Consultas: Miércoles 16.00 hs.



Qué voy a aprender?

- Conceptos de Concurrencia
- Transferir el conocimiento del paradigma OO a un escenario concurrente
- Entender problemas de concurrencia característicos
- Resolver problemas de concurrencia
 - Aplicando mecanismos de sincronización adecuados
 - Utilizando mecanismos existentes para memoria compartida

Donde más lo utilizo?

- Sistemas Operativos I (3^{er} año)
- Laboratorio Programación (3^{er} año)

Bibliografía

- En Biblioteca:
 - Concurrent Programming in Java: Design Principles and Patterns, *Doug Lea*. Addison Wesley - 1997
 - Orientación a Objetos con Java y UML, *Carlos Fontella*, Nueva Librería S.R.L.
 - Java Concurrency in Practice, *Brian Goetz et al.*, Pearson Education , Inc. 2006
 - Thinking in JAVA . *Bruce Eckel*. President, MindView Inc – 2008
 - A Brain-Friendly Guide – Head First Java -*Katty Sierra & Bert Bates*
- Otros:
 - Sun Educational Service, El lenguaje de Programación Java, Sun Microsystems. Inc – *Guía del Estudiante*
 - ...



Temario



- Programas, procesos y concurrencia
- Características de los sistemas concurrentes
- Lenguajes concurrentes
- Trabajo con hilos en Java



Situaciones reales



- Al levantarse
- Llevar adelante varias materias.....
- Multifunción: estudio, escucho musica, tomo mate, ...



¿Qué es concurrencia?

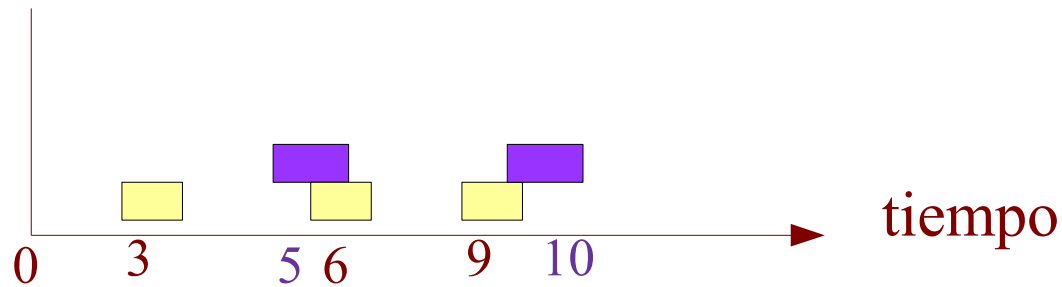
Según Real Academia Española: <Concurso de varios sucesos en un mismo tiempo>

- La concurrencia es la capacidad de ejecutar un **conjunto de actividades** en **forma simultánea**.
- Permite a distintos objetos actuar al mismo tiempo
- Para la Ciencia de la Computación es relevante para el diseño de hardware, SO, multiprocesadores, computación distribuida, programación y diseño.

En informática, cada una de esas actividades se suele llamar **proceso**.

Ejemplo: Problema concurrente

Desplegar cada **3 segundos** un mensaje, ¿*Qué ocurre si se quiere mostrar, además, otro cartel pero cada 5 segundos* ?

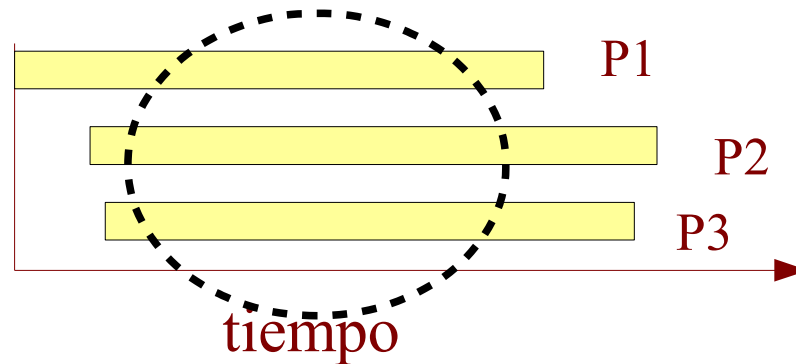


- ¿cómo lo resuelvo de forma secuencial?
- ¿cómo lo resuelvo de forma concurrente?

Concurrencia

Es la **existencia simultánea** de varios procesos en ejecución.

- Dos **procesos** son **concurrentes** cuando la primera instrucción de uno de ellos se ejecuta después de la primera instrucción del otro y antes de la última



Programa vs. Proceso

- **Programa:** Conjunto de sentencias/instrucciones que se ejecutan secuencialmente. Concepto **estático**.

Se asemeja al concepto de **Clase** de la POO

- **Proceso:** Básicamente, se puede definir como un programa en ejecución. Líneas de código en ejecución de manera **dinámica**.

Se asemeja al concepto de **Objeto** de la POO

Qué es un proceso?

- **Programa secuencial:** un solo flujo de control que ejecuta instrucciones en secuencia

PROCESO: es una abstracción lógica. Cada proceso representa típicamente un programa en ejecución separado.

- Un único hilo o flujo de control
 - programación secuencial
- Múltiples hilos o flujos de control
 - programación concurrente

Cómo expresar la concurrencia?

- Las técnicas para producir actividades concurrentes pueden ser:
 - Manuales: Utilizando llamadas al SO o con **bibliotecas de software.**
 - Automáticas: Las detecta el SO en forma automática

Con estas
trabajaremos



Lenguajes de programación

Incorporan características que permiten expresar la concurrencia directamente.

Incluyen mecanismos para ...

Ejemplos: Python, **Java**, Smalltalk,



Concurrencia en Java

Hilo/Thread: proceso liviano – un hilo de control
programación multihilos

La unidad de concurrencia es el Thread, comparte el mismo espacio de variables con los restantes threads

Utiliza la clase **Thread**.



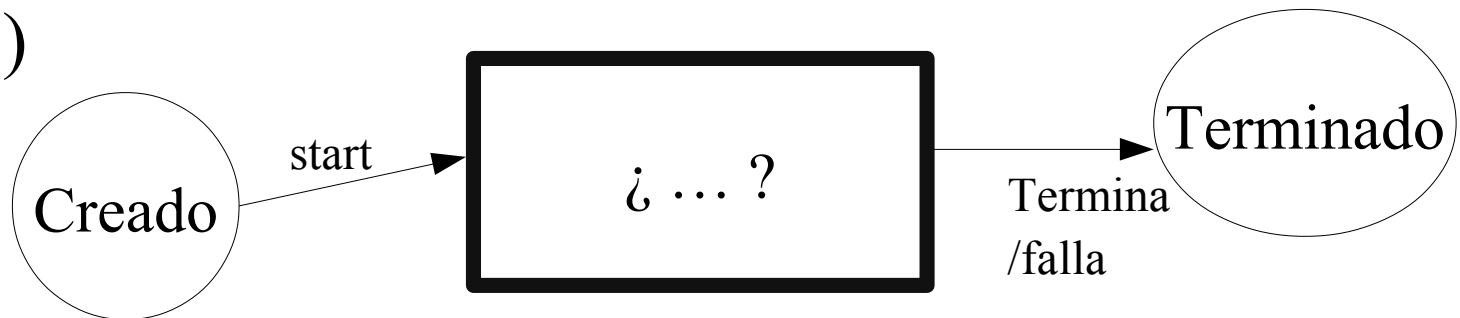
Multitarea en Java: clase Thread

- Un hilo se crea **en Java** instanciando la clase **Thread**.
- El código que ejecuta un thread está definido por el método **run()** que tiene todo objeto que sea instancia de la clase Thread.

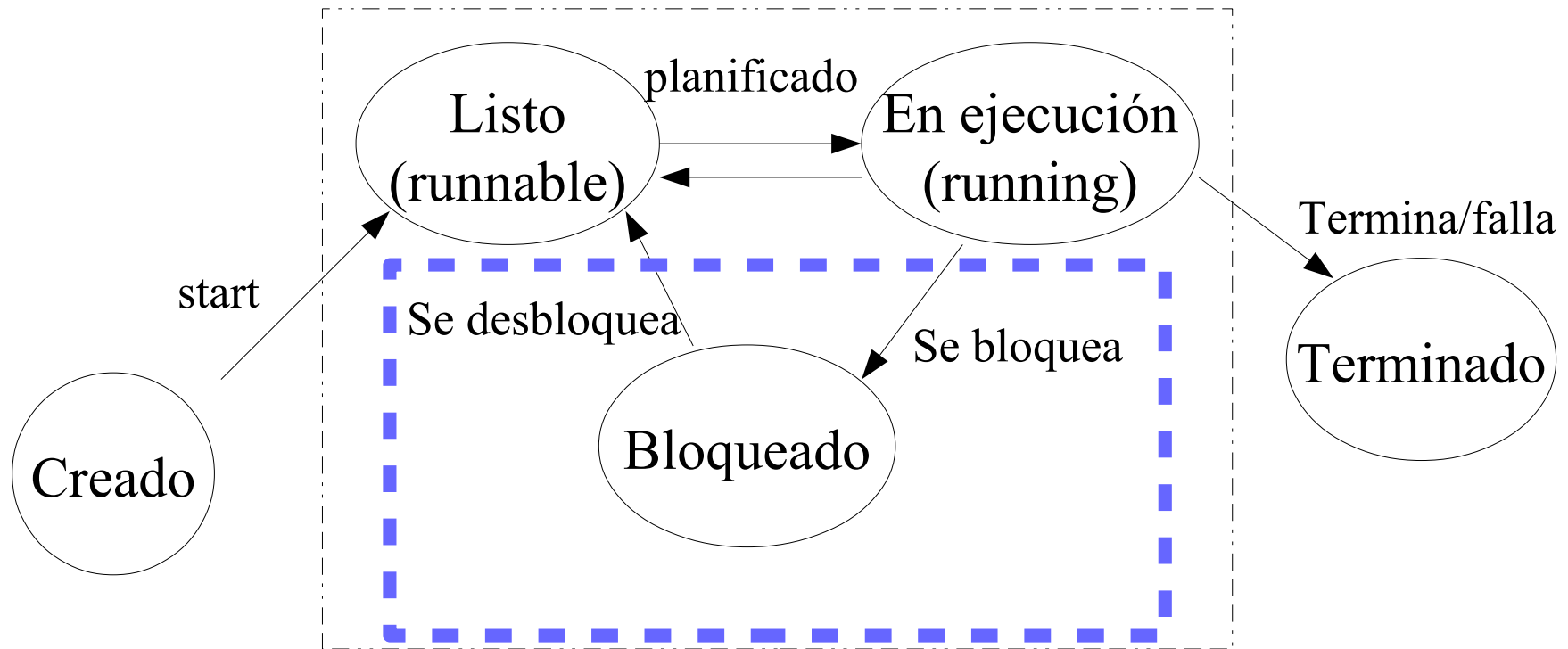
Multitarea en Java: clase Thread

- La ejecución del thread **se inicia** cuando sobre el objeto Thread se ejecuta el método **start()**.
- De forma natural, un **thread termina** cuando en **run()** se alcanza una sentencia **return** o el final del método.

(...)

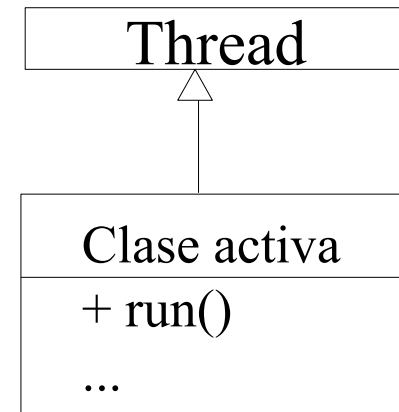


Estados de un hilo



Una forma de crear un hilo

- Crear una *subclase* de Thread
- Implementar el método *run()* con el comportamiento deseado (el método de la clase thread no hace nada)
- *Crear objetos* de esa subclase y activarlos con *start()*.



Crear un hilo por herencia

¿Cuántos hilos hay en ejecución/espera después de t1.start()?

```
public static void main(String[] args){
    PingPong t1 =new PingPong(...);

    // Activación
    t1.start();
    .....

}
```


```
public class PingPong extends Thread{
    // variables propias ...
    // constructor
    public PingPong(...){
        .....
    };

    public void run(){
        // hace algo ...
    }

} //fin clase PingPong
```

¿Cuál termina su ejecución primero?

Crear un hilo por herencia



```
public static void main(String[] args){  
    PingPong t1 = new PingPong(...);  
  
    // Activación  
    t1.start();  
    .....  
  
    t1.join();  
}
```

```
public class PingPong extends Thread{  
    // variables propias ...  
    // constructor  
    public PingPong(...){  
        .....  
    };  
  
    public void run(){  
        // hace algo ...  
    }  
  
} //fin clase PingPong
```

¿y ahora, ... cuál termina su ejecución primero?

Crear un hilo por herencia

```
public static void main(String[] args){
    PingPong t1 = new PingPong("PING",33);
    PingPong t2= new PingPong("PONG",10);
```

// Activación

```
t1.start();
t2.start();
```

// Espera unos segundos

```
try{ Thread.sleep(5000);
    }catch (InterruptedException e){...};
```

// Finaliza la ejecución de los threads

```
...
}
```

```
public class PingPong extends Thread{
    private String cadena; // Lo que va a escribir.
    private int delay; // Tiempo entre escritura
```

```
public PingPong(String cartel,int cantMs){
    cadena = cartel;
    delay = cantMs;
```

```
};
```

```
public void run(){
```

```
    for (int i = 1; i< delay * 10; i++){
```

```
        System.out.print(cadena + " ");
```

```
        try { Thread.sleep(delay);} 
```

```
        catch(InterruptedException e){
```

```
            ... }
```

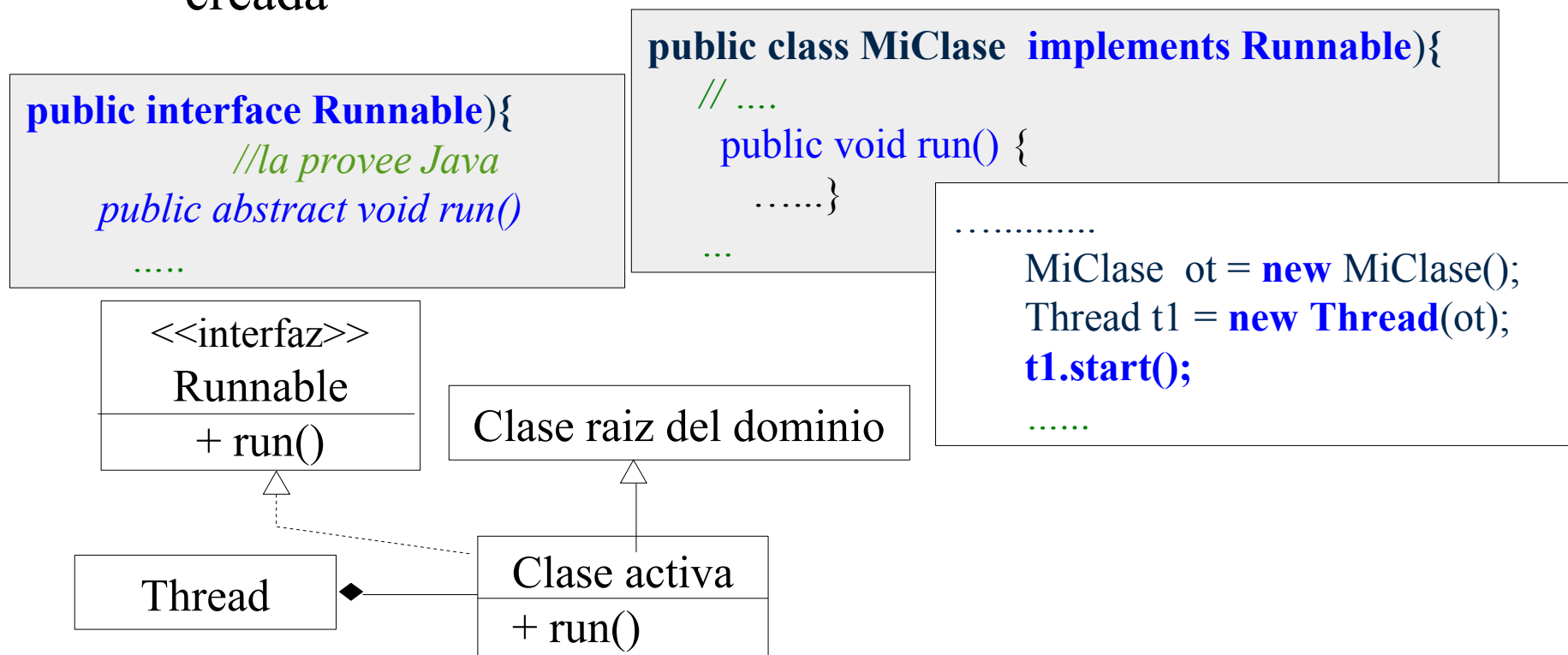
```
        }
```

```
    } //fin método run()
```

```
} //fin clase PingPong
```

Crear un hilo por Interfaz Runnable

- Crear una clase que implemente la *interfaz Runnable*
- Instanciar esa clase
- Crear hilos a partir de Thread utilizando la instancia runnable creada



Crear un hilo por Interfaz Runnable

```
public class PruebaRunnable
{
    public static void main(String[] args){
        // 2 objetos definen los métodos run
        PingPong o1 = new PingPong("PING",33);
        PingPong o2= new PingPong("PONG",10);

        // Se crean los hilos
        Thread t1 = new Thread (o1);
        Thread t2 = new Thread (o2);

        // se activan los hilos
        t1.start;
        t2.start;

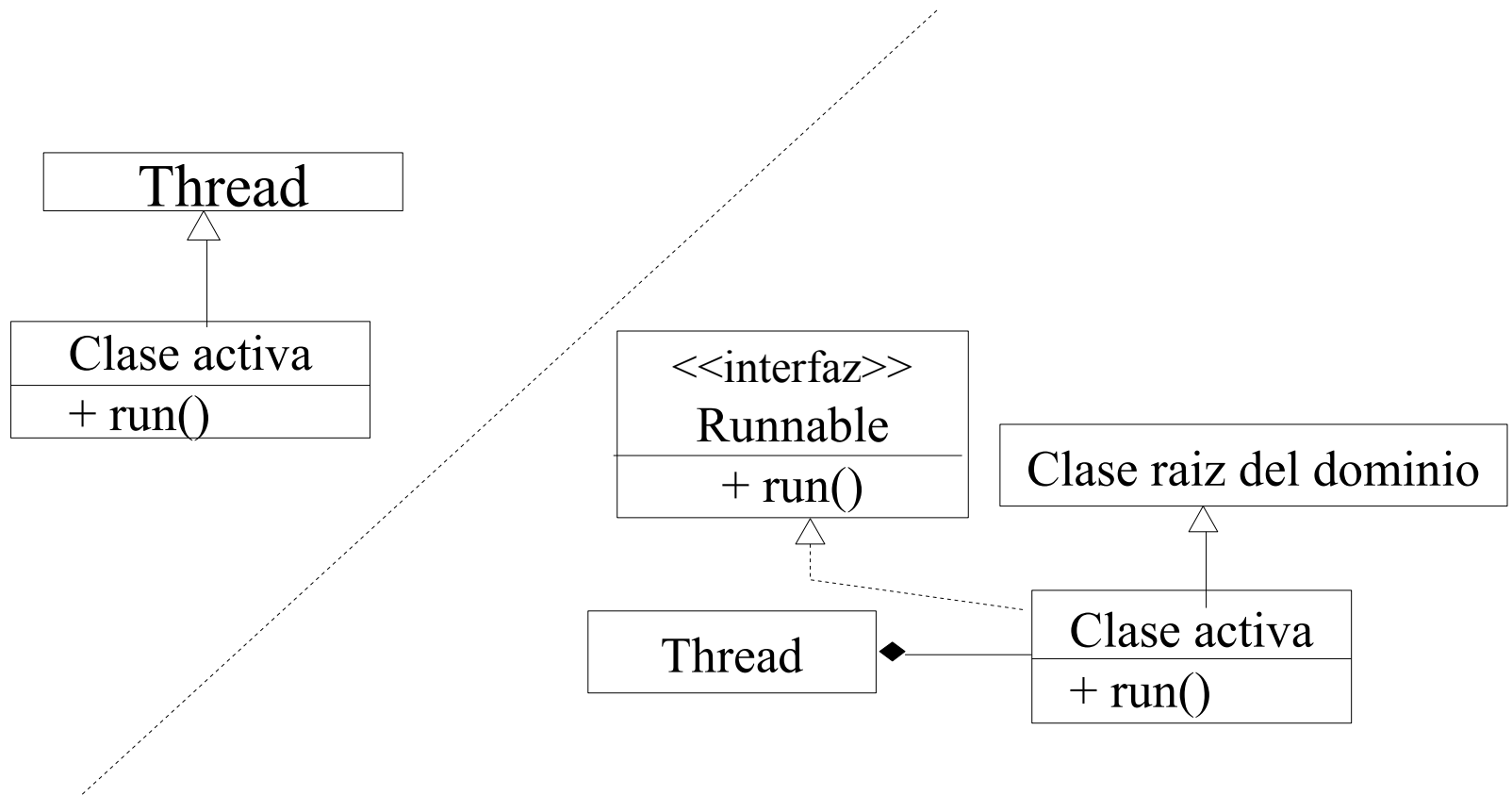
        ...
    }
}
```

```
public class PingPong implements Runnable{
    private String cadena; // Lo que va a escribir.
    private int delay; // Tiempo entre escritura

    public PingPong(String cartel,int cantMs){
        cadena = cartel;
        delay = cantMs;
    };

    public void run(){
        for (int i = 1; i< delay * 10; i++){
            System.out.print(cadena + " ");
            try { Thread.sleep(delay);}
            catch (InterruptedException e){
                ... }
        }
    } //fin método run()
} //fin clase PingPong
```

Entonces ... para crear hilos



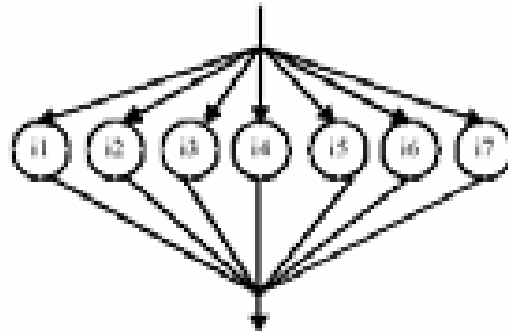


Constructores de la clase Thread

- **Thread()**
- **Thread**(Runnable threadOb)
- **Thread**(Runnable threadOb, String threadName)
- **Thread**(String threadName)

Un problema propio de la Prog Concurrente

Indeterminismo: Un programa concurrente define un orden **parcial** de ejecución. Ante un conjunto de datos de entrada no se puede saber cual va a ser el flujo de ejecución



Los programas concurrentes pueden producir diferentes resultados en ejecuciones repetidas sobre el mismo conjunto de datos de entrada

Escenario indeterminístico

- Cómo un planificador (scheduler) puede ser indeterminístico?

```
public class RunThread implements Runnable{  
    public void run() {  
        for (int i=0; i <30; i++){  
            String threadNombre = Thread.currentThread().getName();  
            System.out.println(threadName + "en ejecucion");  
        }  
    }  
}
```

```
public class TestRunThread {  
    ...
```

```
    public static void main (String[] args) {  
        RunThreads runner = new RunThreads();
```

```
        Thread alfa = new Thread (runner);  
        Thread beta = new Thread (runner);
```

```
        alfa.setName ("Hilo Alfa");  
        beta.setName ("Hilo Beta");
```

```
        alfa.start(); beta.start();
```

```
    }  
}
```

instancia Runnable

dos hilos con la
misma implementación
runnable

nombra los hilos

los hilos pasan a estado "runnable/listo"

¿Cual es la salida?

Hilo Alfa en ejecucion
Hilo Alfa en ejecucion
Hilo Alfa en ejecucion
Hilo Alfa en ejecucion
Hilo Beta en ejecucion
Hilo Alfa en ejecucion

..

Posible salida

Entonces...conurrencia...

- **Proceso:** secuencia de acciones que se realizan independientemente de las acciones realizadas por otros procesos.

En general, la prioridad la asigna el SO!!!!

- Los **procesos** que se ejecutan en paralelo, **son objetos** que poseen una prioridad, la cual puede asignarse en un principio e ir modificándose a lo largo de la ejecución.

- La **prioridad** de un proceso describe la importancia que tiene ese proceso por sobre los demás.