



Departamento de Programación
Facultad de Informática
Universidad Nacional del Comahue



Programación Concurrente



Inicio



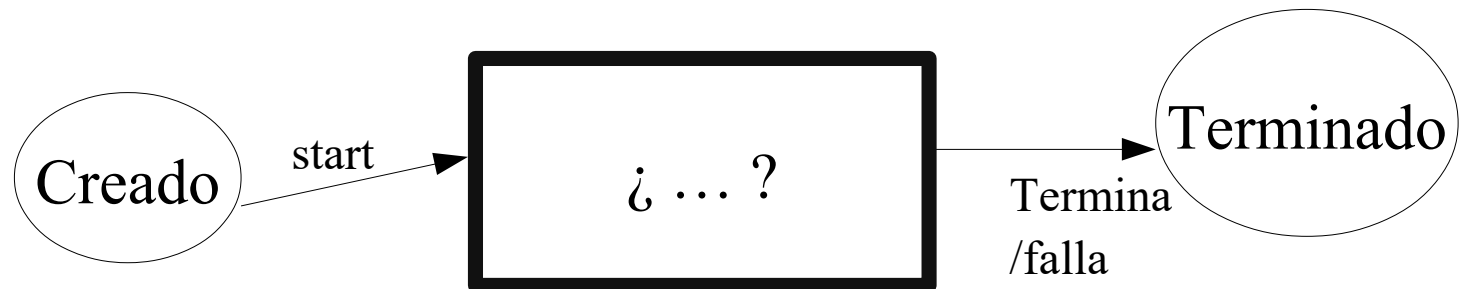
Incumbencias de la PC

- Los procesos pueden “competir” o “colaborar” entre sí por los recursos del sistema.

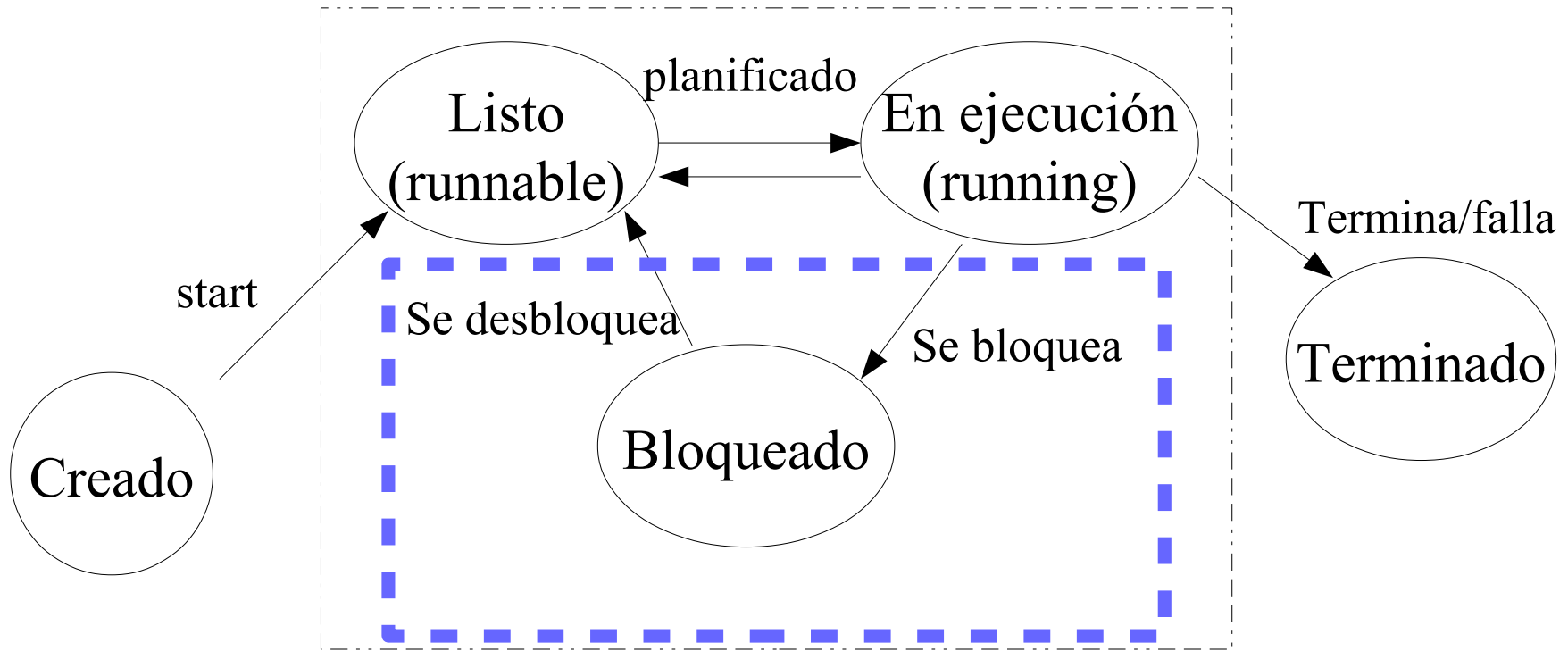
La programación concurrente (PC) se encarga del estudio de las nociones de ejecución concurrente, así como de sus **problemas de comunicación y sincronización.**

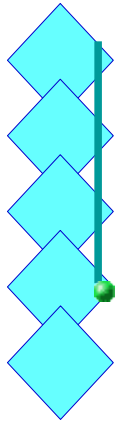
Multitarea en Java: clase Thread

- 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.
- 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



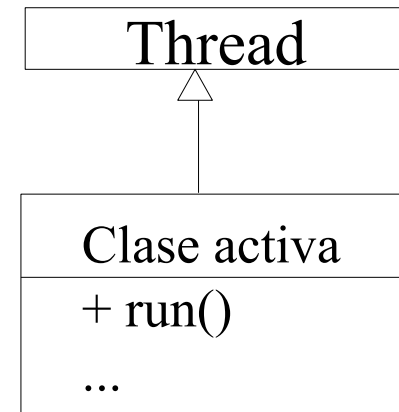


En Java, hay dos formas de trabajar la creación de los hilos:



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();
    .....

}
```

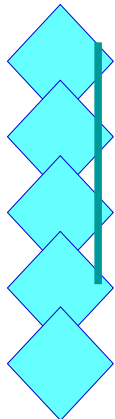
```
public class PingPong extends Thread{
    // variables propias ...
    // constructor
    public PingPong(...){
        .....
    };

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

} //fin clase PingPong
```

¿Cuál termina su ejecución primero?

Crear un hilo por herencia

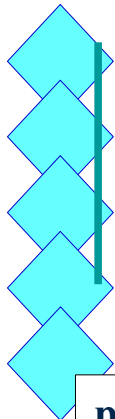


```
public static void main(String[] args){  
    PingPong t1 = new PingPong(...);  
  
    // Activación  
    t1.start();  
    .....  
  
    t1.join();  
}
```

```
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 herencia

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

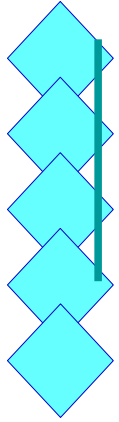
    // Activación
    t1.start();
    t2.start();

    // Espera unos segundos
    try{ Thread.sleep(5000);
        }catch (InterruptedException e){...};

    // Finaliza la ejecución de los threads
    ...
}
```

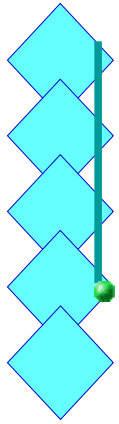
```
public class PingPong extends Thread{
    private String cadena; // Lo que va a escribir.
    private int delay; // Tiempo entre escritura

    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
```



Constructores de la clase Thread

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



En Java, hay dos formas de trabajar la creación de los hilos:



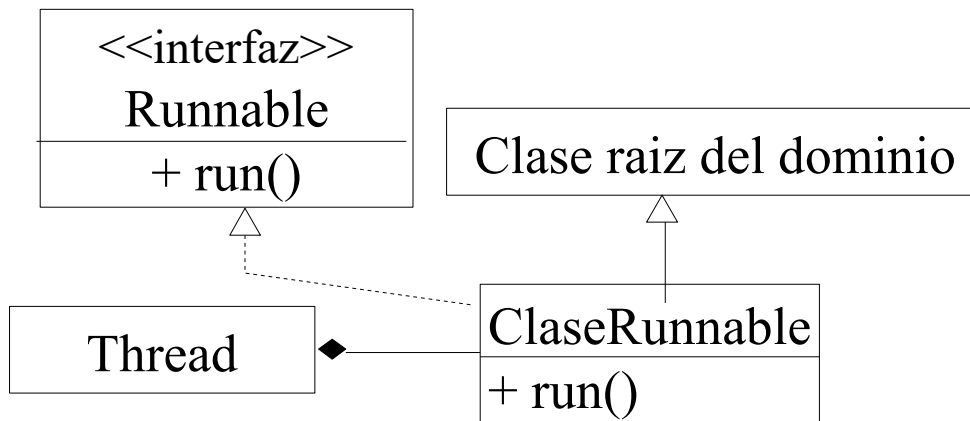
Crear un hilo por Interfaz Runnable

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

```
public interface Runnable){  
    //la provee Java  
    public abstract void run()  
    ...  
}
```

```
public class MiClase implements Runnable){  
    // ...  
    public void run() {  
        ...  
    }  
    ...  
}
```

```
...  
MiClase objRun = new MiClase();  
Thread t1 = new Thread(objRun);  
t1.start();  
...
```



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;
    };

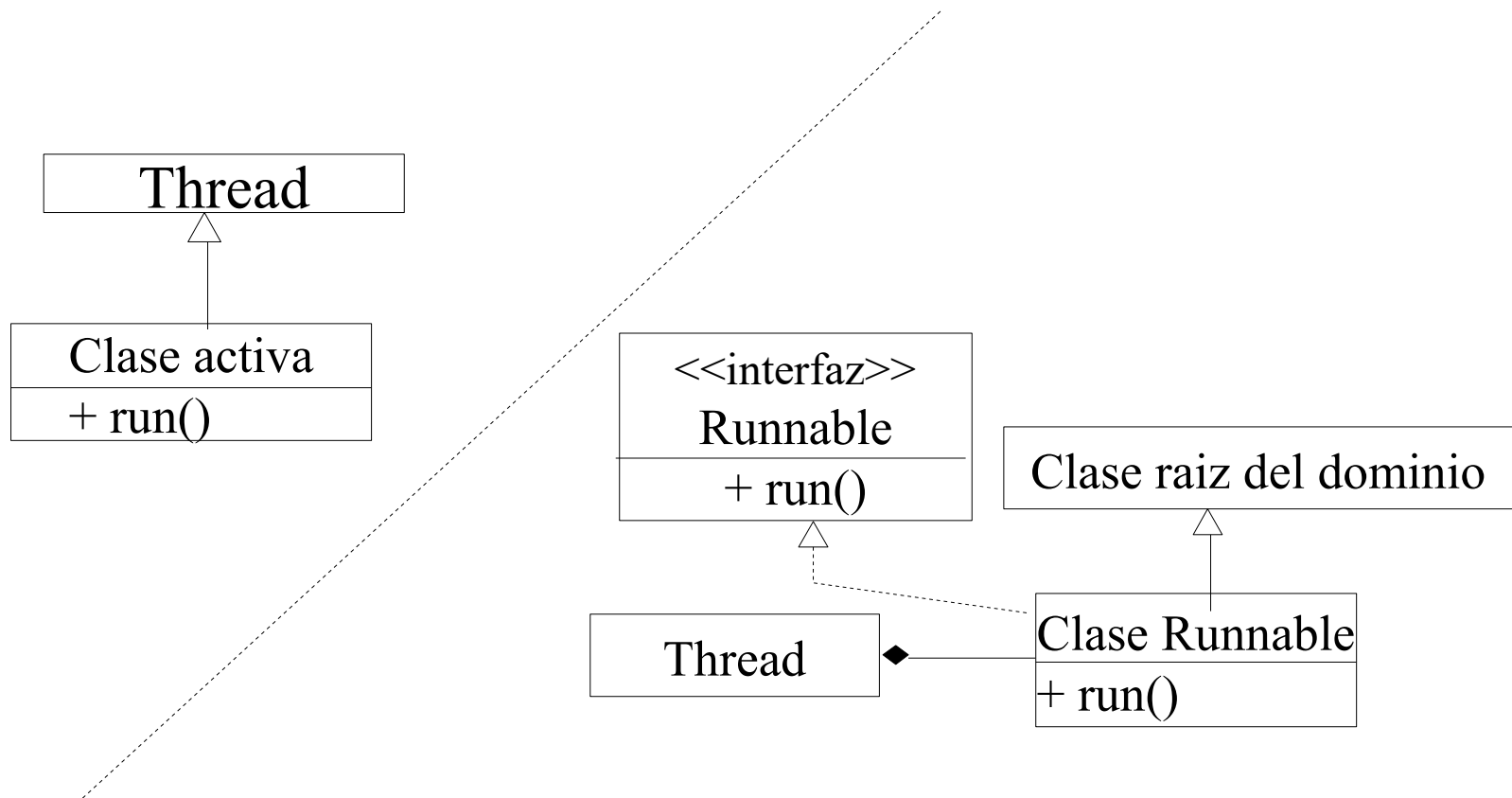
    public void run(){
        for (int i = 1; i < delay * 10; i++){
            System.out.print(cadena + " ");
            try { Thread.sleep(delay);}
            catch (InterruptedException e){
                ... }
        }
    } //fin método run()
} //fin clase PingPong
```



Constructores de la clase Thread

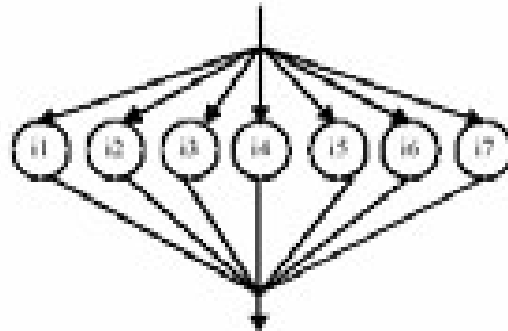
- **Thread()**
- **Thread**(String threadName)
- **Thread**(Runnable threadOb)
- **Thread**(Runnable threadOb, String threadName)

Entonces ... para crear hilos



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: otro ejemplo

```
public class HiloAlfaBeta extends Thread{
    // completar con constructores
    public void run() {
        for (int i=0; i < 30; i++){
            System.out.println(this.getName() + " en ejecucion");
        }
    }
}
```

```
public class TestRunThread {
    ...

    public static void main (String[] args) {

        HiloAlfaBeta alfa = new HiloAlfaBeta("Hilo Alfa");
        HiloAlfaBeta beta = new HiloAlfaBeta("Hilo Beta");

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

    }
}
```

los hilos se crean
con nombre

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

Escenario indeterminístico

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

```
public class TestRunAlfaBeta {  
    ...  
    public static void main (String[] args) {  
        RunnableAlfaBeta alfaBetaRunnable = new RunnableAlfaBeta();  
  
        Thread alfa = new Thread ( alfaBetaRunnable, "HiloAlfa");  
        Thread beta = new Thread ( alfaBetaRunnable);  
  
        beta.setName ("Hilo Beta");  
        alfa.start();  
        beta.start();  
    }  
}
```

instancia Runnable

nombra al segundo hilo

¿Cual es la salida?

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

Posible salida

Se crean dos hilos con la misma implementación Runnable.
Uno se crea con nombre

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 concurrentemente, **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.