

Algo3

Concurrencia

Técnicas de programación

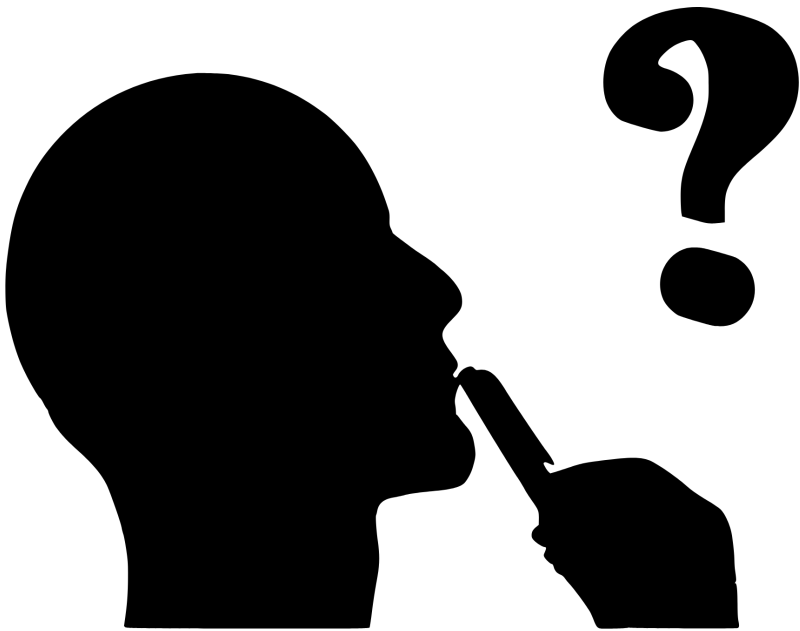
¿ Por qué concurrencia?

- Performance
- Tiempo de respuesta al usuario
- Tiempo de ejecución de una aplicación
- “Mundo paralelo”
- Aprovechar el hardware



¿ Concurrente?

¿ Paralelo?



FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

algo3

*“Concurrencia es tratar de **lidiar** con muchas cosas a la vez.
Paralelismo es **hacer** muchas cosas a la vez”*

- Rob Pike

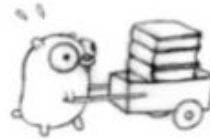
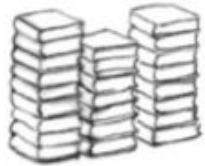


Algunos conceptos

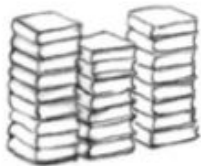
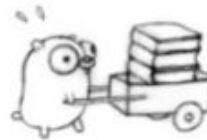
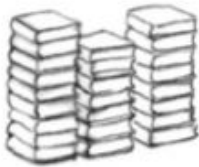
- *Concurrencia* es la composición de la ejecución de “cosas” independientes. Es sobre la estructura de los problemas, descomponer tareas en tareas más pequeñas.
- *Paralelismo* es ejecutar tareas en simultáneo.
- Concurrencia **no implica** paralelismo



Problema

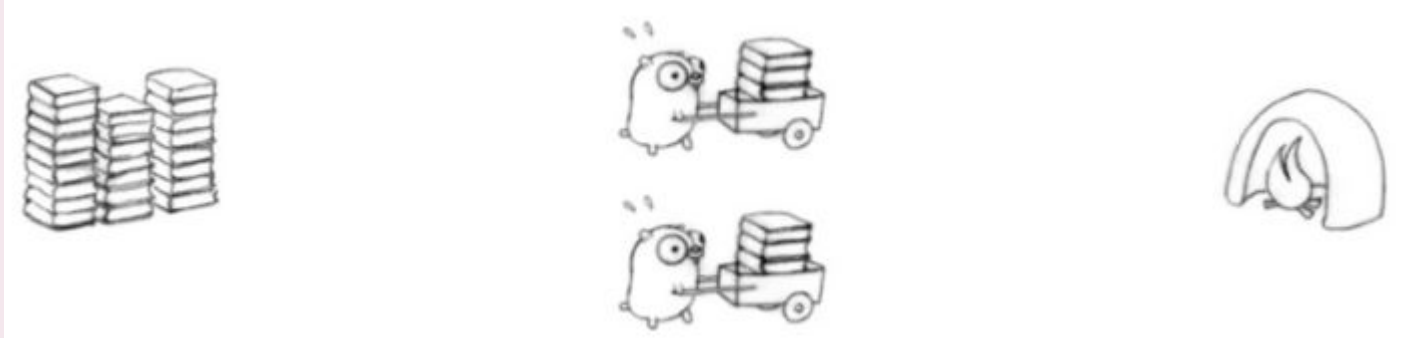


Paralelo



Concurrente

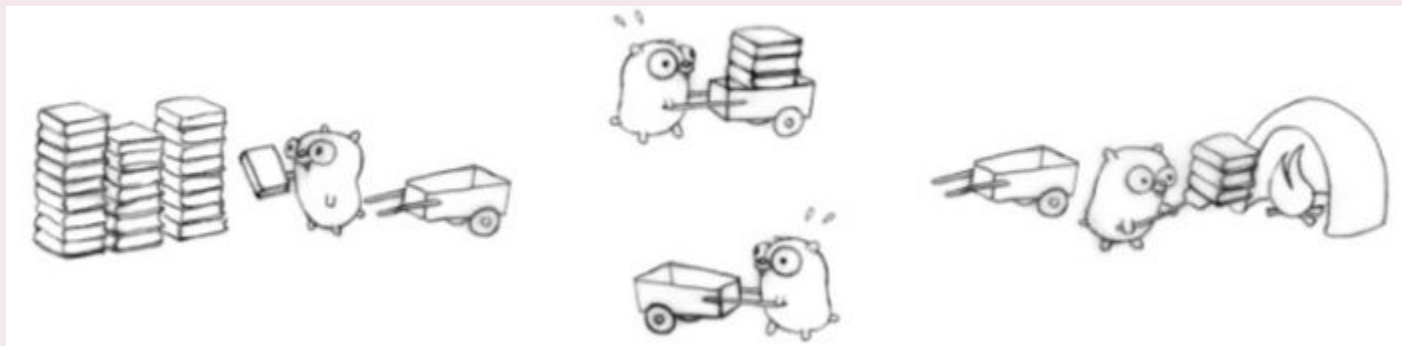
1



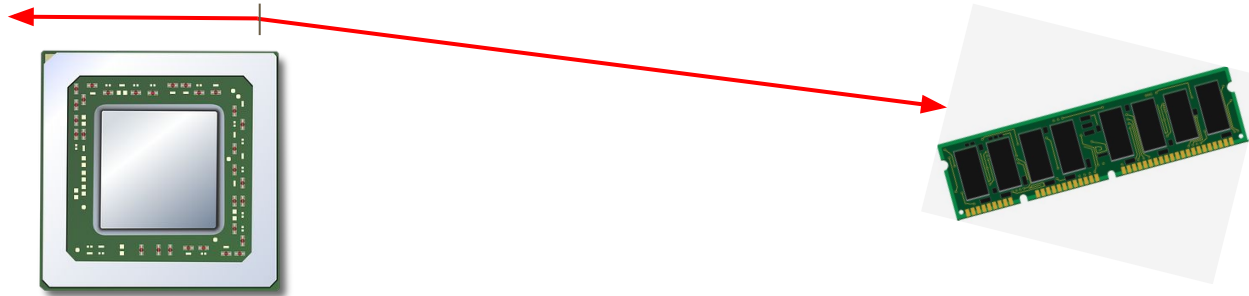
2




3



Ejercicio teórico



¿Concurrente? 

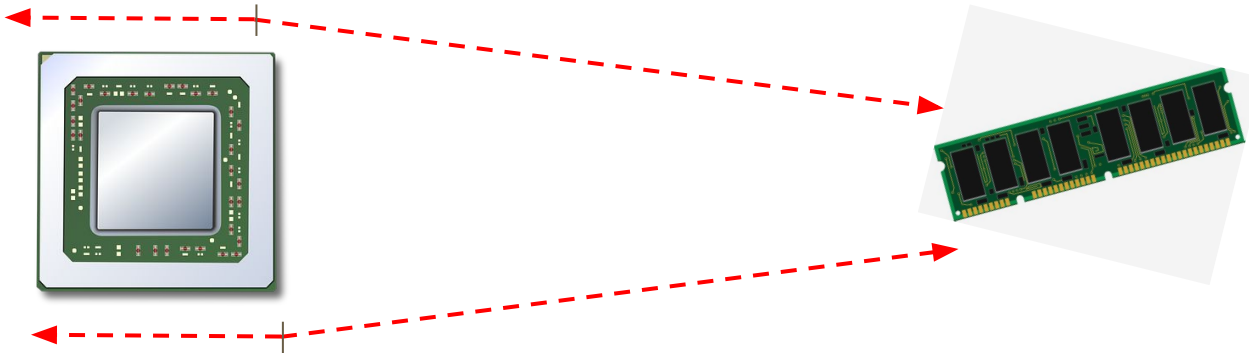
¿Paralelo? 



FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

algo3

Ejercicio teórico



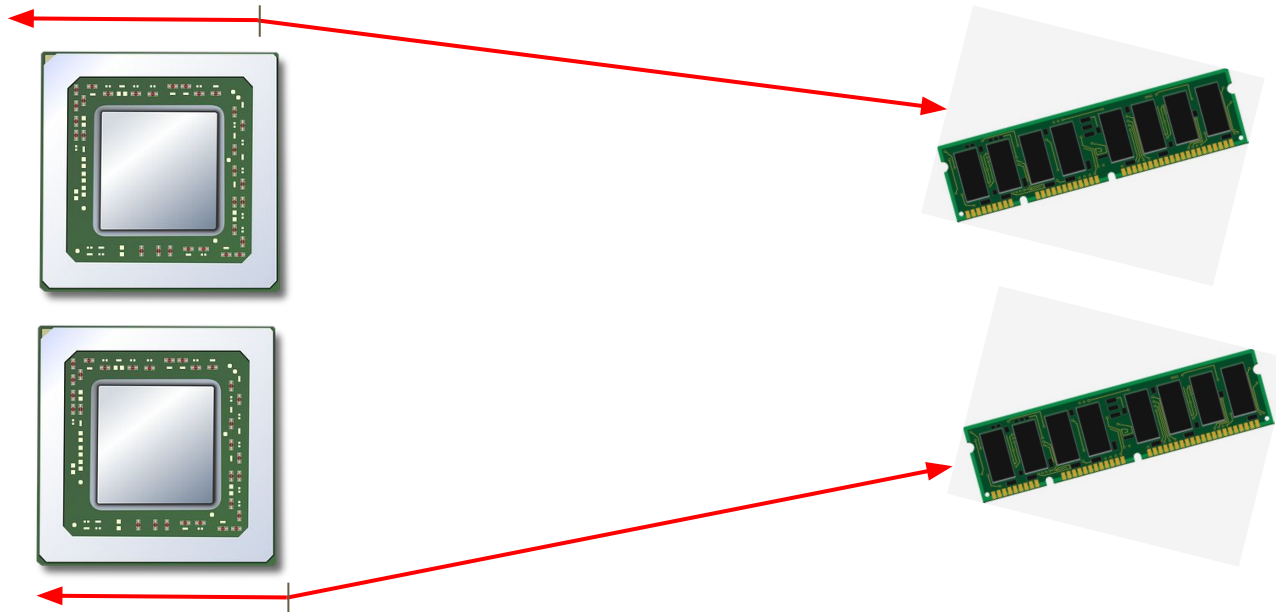
¿Concurrente?



¿Paralelo?



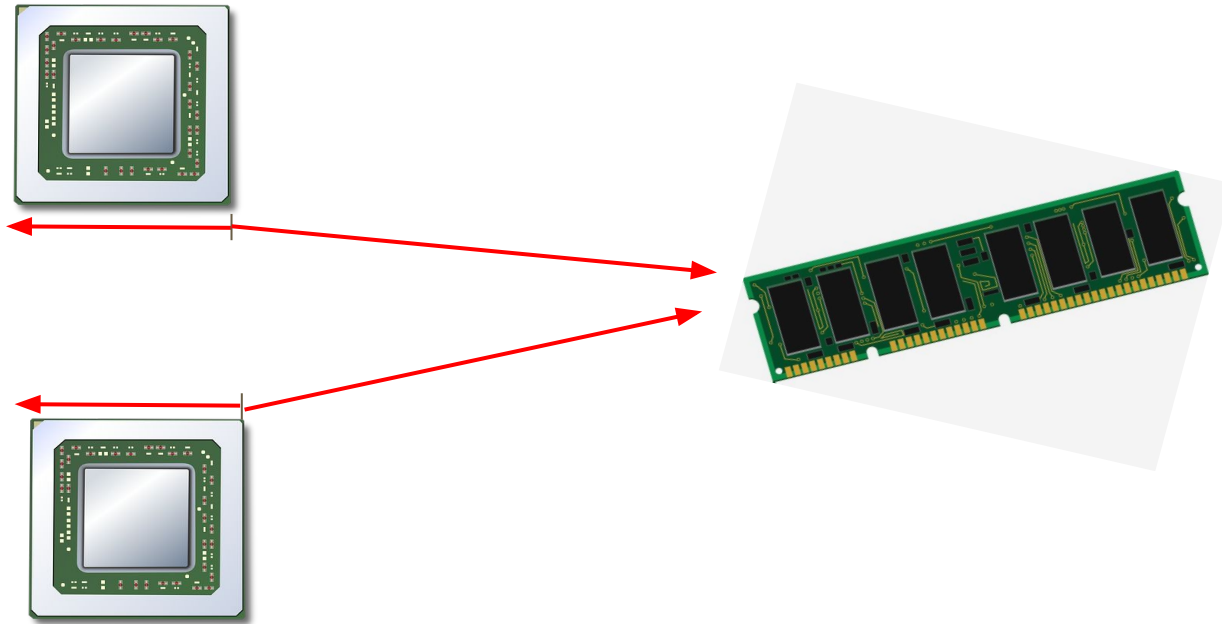
Ejercicio teórico



¿Concurrente? 

¿Paralelo? 

Ejercicio teórico



¿Concurrente?



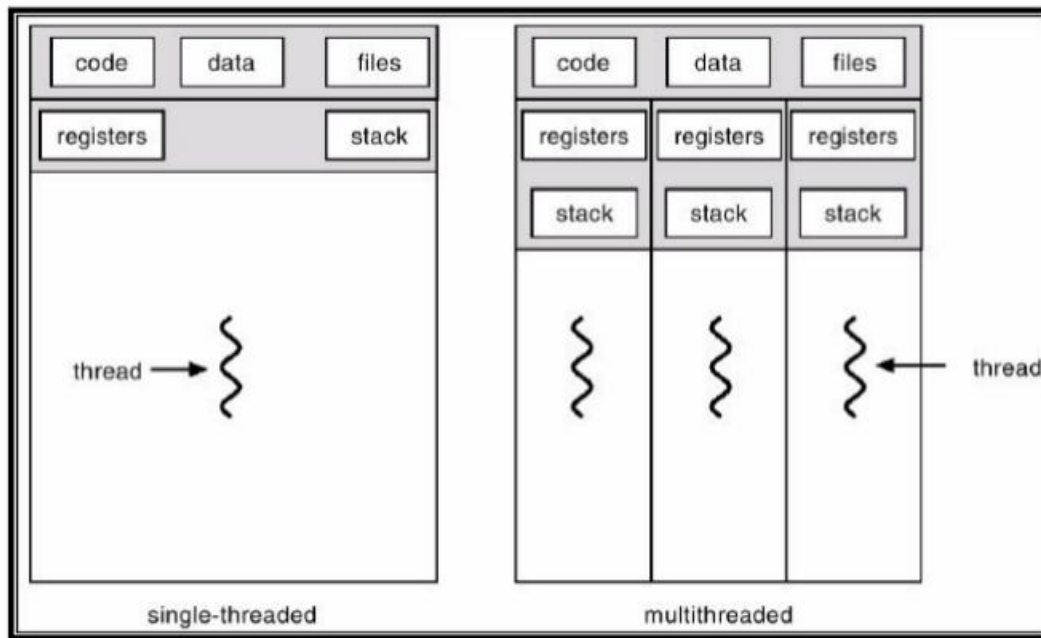
¿Paralelo?



¿Qué es un thread?

Secuencia independiente de instrucciones ejecutándose dentro de un programa.

Función o clase que se ejecuta de manera concurrente.



Problemas

- **Race condition**

Se da cuando varios threads pueden acceder a recursos compartidos (código). El resultado del programa depende de cómo se intercalen los threads.

- **Critical section**

Sección de código que necesita ser ejecutada en forma **atómica** por un solo hilo a la vez.



Solución : Sincronización

Locks

Se basa en el uso de una variable de **exclusión mutua** (mutex).



Monitores

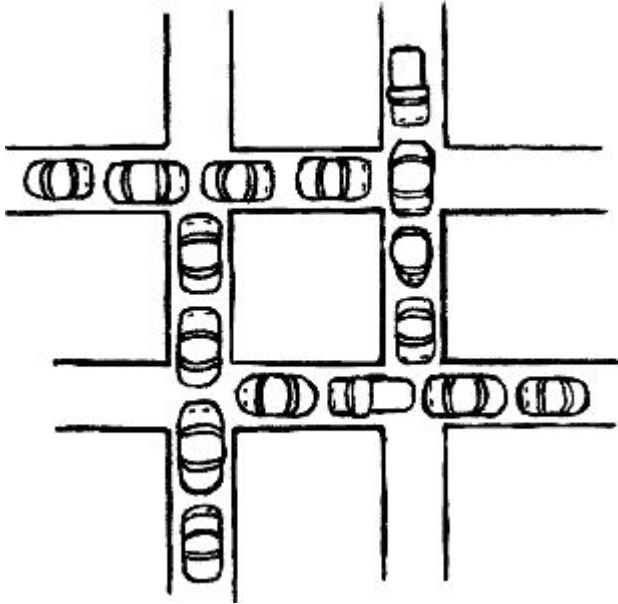
Objetos thread-safe, sus métodos están sincronizados (mutex).

Conditional variable

Mecanismo de bloqueo con una señalización.

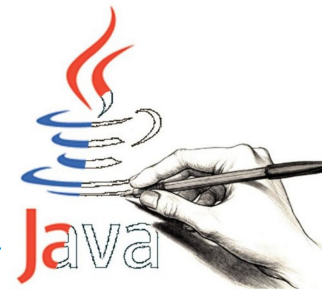
Semáforos

Más problemas - DeadLock

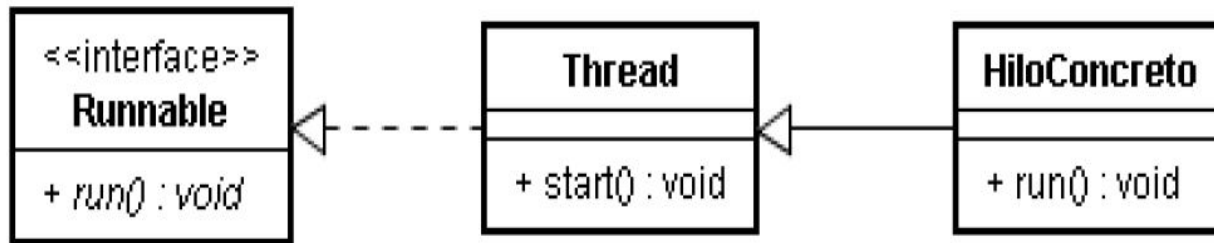


Aparece cuando entre dos o más threads uno obtiene un recurso y no lo libera generando un bloqueo.

¿Cómo se implementa?



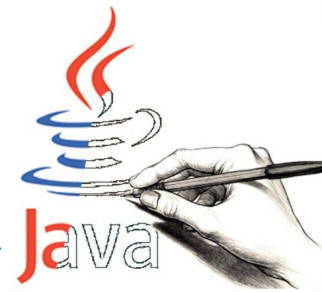
1. Heredar de Thread (Template Method)



```
public class HiloConcreto extends Thread
{
    @Override
    public void run() {
        // código del hilo
    }
}
```

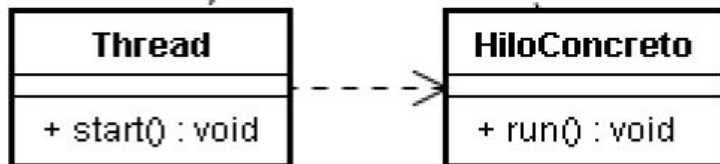
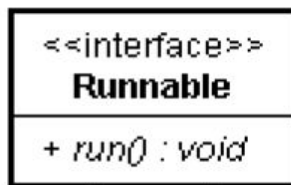
```
....
HiloConcreto unHilo = HiloConcreto();
unHilo.start();
...
```


¿Cómo se implementa?



2. Implementar Runnable

Más flexible



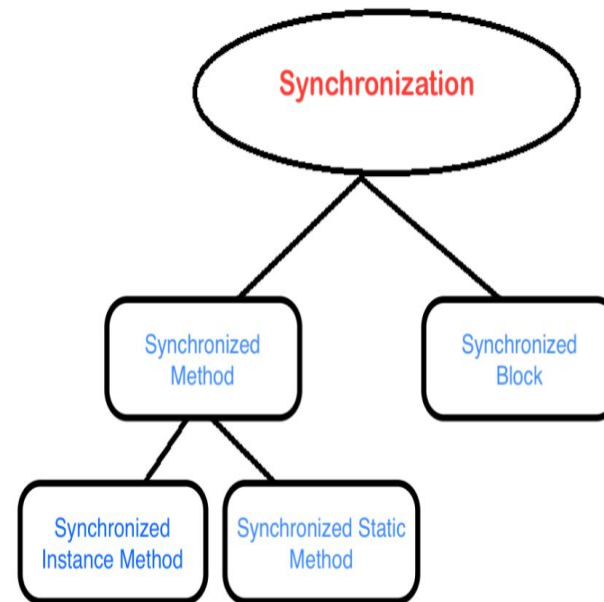
```
public class HiloConcreto implements Runnable {
    @Override
    public void run() {
        // código del hilo
    }
}
```

```
...
new Thread(new HiloConcreto()).start();
```

Sincronización



```
public synchronized void metodo() {  
    //...  
}  
  
public void metodo() {  
    synchronized(this) {  
        //...  
    }  
}  
  
public static synchronized void  
metodo() {  
    //...  
}
```



Más sobre la API

- **t.interrupt()** -> Interrumpe el thread.
- **t.join()** -> Espero a que el thread termine.
- **Thread.sleep(mls)** -> Duerme el thread por mls.

General

- **objeto.wait()** -> El thread queda en espera.
- **objeto.notify()** -> Despierto a uno de los thread que esperan.
- **objeto.notifyAll()** -> Despierto a todos los que esperan.

**Condition
Variables**

Ejemplo Práctico



Consideraciones

- Debuggear es **complicado**.
- Agregar **println()** puede alterar el resultado.
- Demasiada **sincronización** perdemos la ventaja de usar threads.
- Poca **sincronización** genera errores difíciles de detectar.



¿Dudas?



FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

algo3