


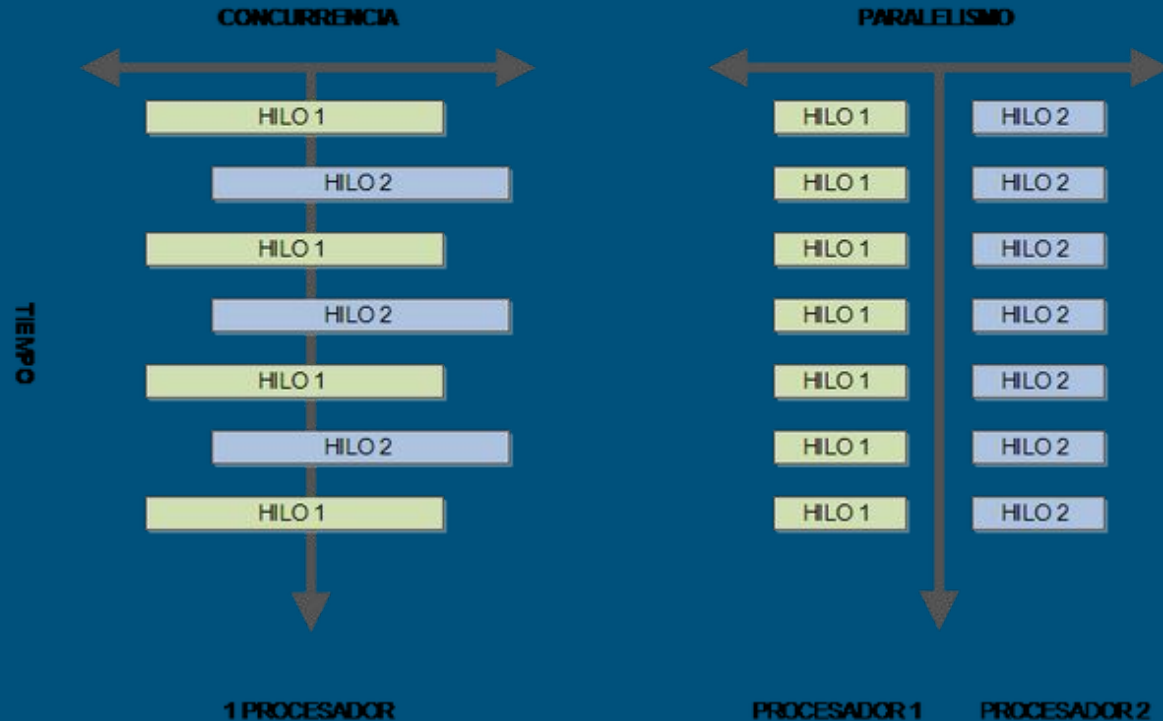
# Lenguaje de programación Java SE 8



Módulo 2 - Clase #05  
Carrera Java Programmer SE 8



# Concurrencia vs Paralelismo



# Concurrencia (Threads)

## Características:

- Programación concurrente de código a través de hilos.
- Un mismo proceso puede tener:
  - Un único hilo -> Monotarea
  - Varios hilos -> Multitarea
- Flujos de ejecución secuencial dentro de un proceso.
- Conocidos también como procesos ligeros: Lightweight Process = LWP
- Se encuentra en el paquete **java.lang.\***

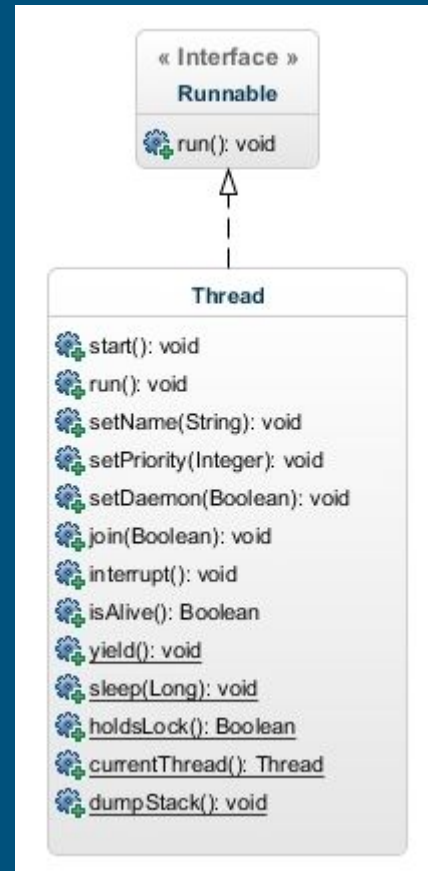
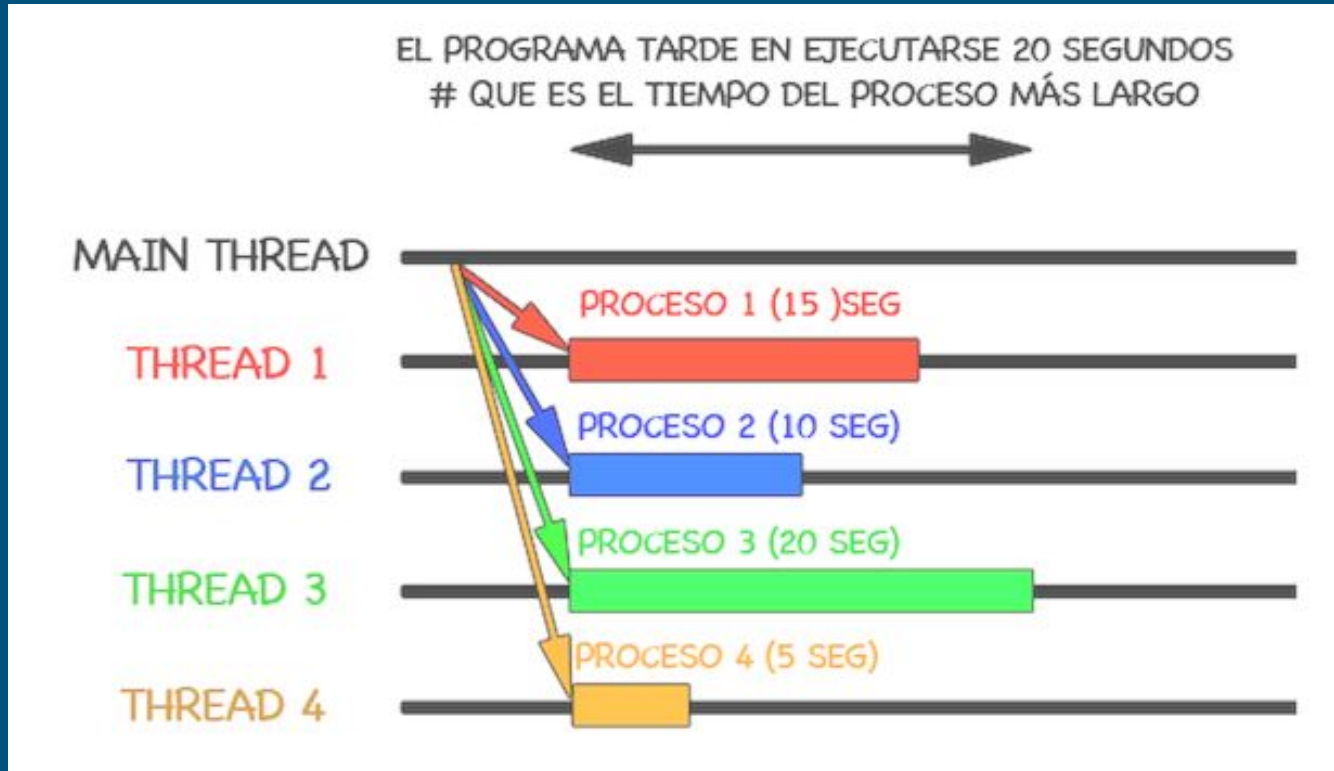
## Los hilos de un proceso comparten:

- Espacio de memoria.
- Variables globales.
- Archivos abiertos.
- Procesos hijos.
- Temporizadores.
- Señales y semáforos.
- Contabilidad.

# Concurrencia (Threads)



# Concurrencia (Threads)



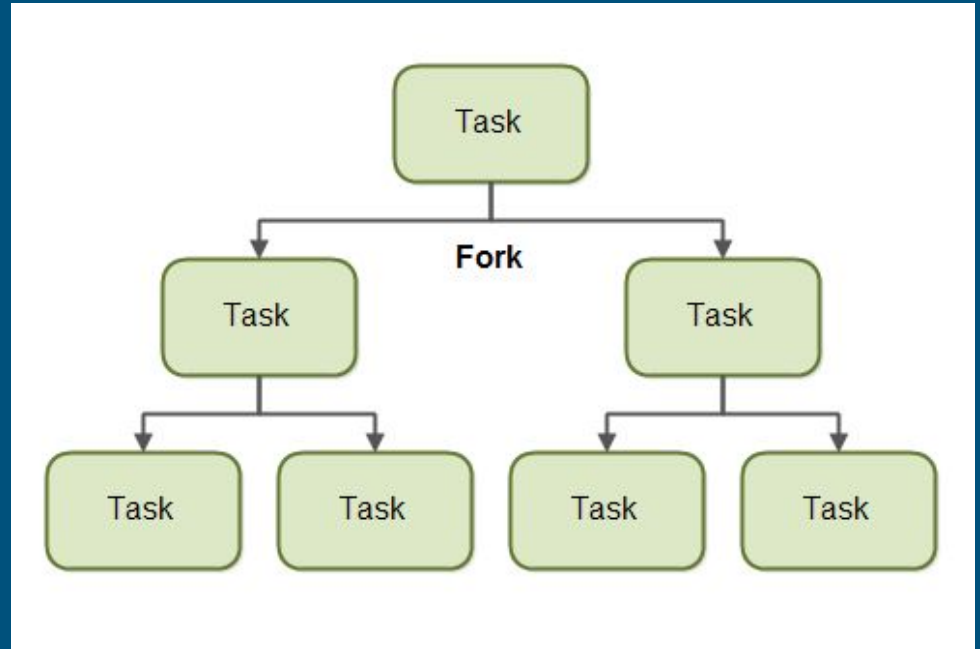
# Ejemplo



# Fork-Join Framework

## Características:

- Disponible desde Java 7.
- Ejecuta tareas de forma paralela.
- Aplica el principio de “divide y vencerás”.
- Implementa el algoritmo work-stealing.
- Hay dos operaciones:
  - Dividir una tarea en tareas más pequeñas “fork”
  - Esperar a que las tareas finalicen “join”.



# Fork-Join Framework

- Clases:
  - **RecursiveAction**: No regresa un valor.
  - **RecursiveTask<T>**: Regresa un valor
- Métodos:
  - **compute()**: Sobreescibir para la tarea
  - **fork**: tarea
  - **join**: resultado -> error
  - **get**: resultado -> exception
- Se encuentran en el paquete **java.util.concurrent.\***

```
public class MyRecursiveAction extends  
    RecursiveAction {}
```

```
public class MyRecursiveTask extends  
    RecursiveTask<T> {}
```



# Ejemplo



# Parallel Streams

## Características:

- Es un helper para nuestras colecciones.
- Se ejecuta de forma asíncrona.

## Métodos:

- `parallelStream().sorted()`
- `parallelStream().filter()`
- `parallelStream().map()`
- `parallelStream().limit()`
- `parallelStream().count()`

## Ejemplos:

```
.parallelStream().filter(x ->  
x.startsWith("m")).forEach(System.out::println);
```

```
.parallelStream().sorted().forEach(x ->  
System.out.print(x + " "));
```

```
.parallelStream().map(String::toUpperCase).forE  
ach(x -> System.out.print(x + " "));
```

```
.parallelStream().limit(2));
```

```
.parallelStream().count();
```

# Ejemplo

