Corporación Universitaria Empresarial Alexander von Humboldt
VIGILADA MINEDUCACIÓN

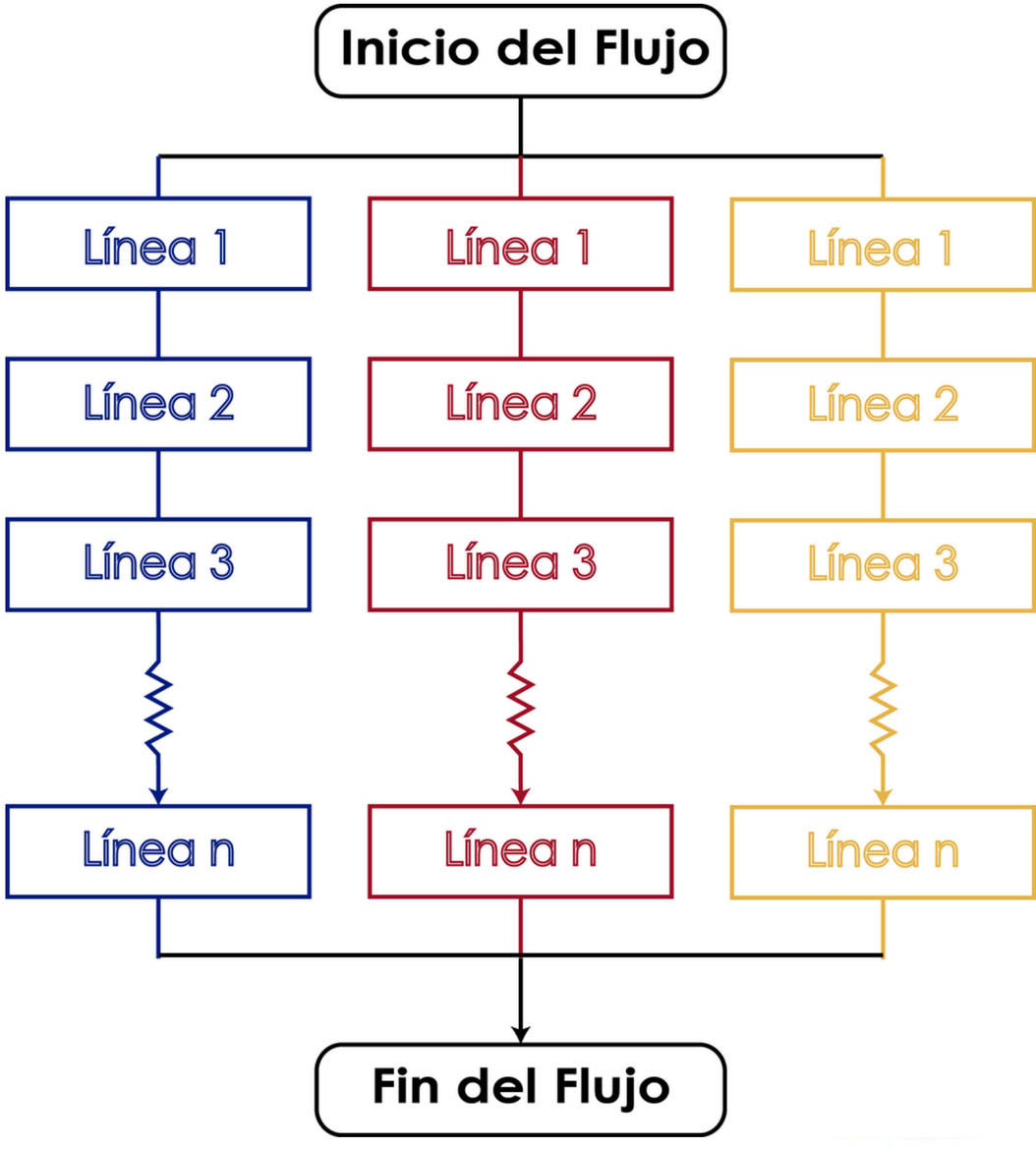
Filial de: CÁMARA DE COMERCIO DE ARMENIA Y DEL QUINDÍO

HILOS EN JAVA

PROGRAMA DE INGENIERÍA DE SOFTWARE
CUE AvH Dual

¿Qué son los hilos en java?

Un hilo es una secuencia independiente de ejecución en un programa. Java soporta el multihilo, lo que significa que puede ejecutar múltiples hilos simultáneamente, aprovechando mejor los recursos del sistema.



Como se crean?

Método 1:

La clase a la que queremos correr la vamos a extender de la clase Thread y dentro de la clase creamos un método “public void run()” en el cual usaremos un for para hacer un recorrido del programa y usamos un try/catch en el cual proparemos un Thread.sleep(500) en cual dara unos microsegundos de espera entre ejecucion del hilo.

```java
package Ejemplos;

public class MultithreadThing extends Thread{
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {

            System.out.println(i);

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e){
                e.printStackTrace();
            }



        }
    }
```

Método 2:

El segundo metodo es muy similar al primero en cuanto a estructura de la clase pero en este caso en vez de extender de la clase Thread, implementaremos de la clase Runnable y usaremos el mismo metodo run.

```java
public class MultithreadThing implements Runnable{
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {

            System.out.println(i);

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e){
                e.printStackTrace();
            }



        }
    }
}
```

Como implementarlos?

Método 1:

Con el primer método simplemente crearemos una nueva instancia de la clase y llamaremos al método start() para correr el programa.

No confundirlo con el metodo run() ya que esto nos correra el programa directamente sin tener en cuenta el hilo.

```
public class Main {
    public static void main(String[] args) {
        MultithreadThing myThing = new MultithreadThing();

        myThing.start();
    }
}
```

Como implementarlos?

Método 2:

En el segundo método sera un poco diferente

En este caso crearemos una instancia de la clase y después crearemos una instancia de la clase ya definida por java la cual es “Thread” y le pasaremos como parámetros la clase que instanciamos. y finalmente usamos el mismo start() para correr el programa

```
1   package Ejemplos;
2
3   public class Main {
4       public static void main(String[] args) {
5           MultithreadThing myThing = new MultithreadThing();
6
7           Thread hiloRunnable = new Thread(myThing);
8
9           hiloRunnable.start();
10
11
12      }
13  }
```

Características

1. Ejecución Concurrente

Los hilos permiten la ejecución de múltiples tareas al mismo tiempo dentro de un programa. Cada hilo puede ejecutar su propia tarea de manera independiente.

2. Ligereza

Los hilos son más ligeros que los procesos. Pueden compartir recursos del mismo proceso, como la memoria, lo que los hace más eficientes en términos de rendimiento y uso de recursos.

3. Contexto Compartido

Todos los hilos dentro de un mismo proceso comparten el mismo espacio de memoria. Esto significa que pueden acceder y modificar las mismas variables y objetos. Sin embargo, esto también puede generar problemas de concurrencia si no se manejan adecuadamente.

4. Cambio de Contexto

El sistema operativo o el administrador de hilos de Java puede cambiar entre hilos (multitarea) de forma automática. El cambio entre hilos es rápido y eficiente, permitiendo que el sistema ejecute múltiples tareas de manera aparente al mismo tiempo.

5. Prioridades de Hilos

Cada hilo tiene una prioridad asociada que puede influir en la forma en que el planificador de hilos asigna tiempo de CPU. Un hilo con mayor prioridad tiene más posibilidades de ejecutarse antes que uno con prioridad menor, aunque esto depende del sistema operativo.

6. Sincronización

Dado que los hilos comparten memoria, puede haber conflictos si varios hilos acceden a los mismos datos simultáneamente. Para evitar estos problemas, Java permite la sincronización de los hilos usando la palabra clave synchronized, asegurando que solo un hilo acceda a una sección crítica de código a la vez.

7. Estados de un Hilo

Un hilo en Java puede estar en uno de los siguientes estados:

Nuevo (New): Cuando el hilo ha sido creado pero no se ha iniciado.

Ejecutable (Runnable): El hilo está listo para ejecutarse o en ejecución.

En Espera (Waiting): El hilo está esperando que ocurra un evento o recurso.

Bloqueado (Blocked): El hilo está bloqueado, por ejemplo, esperando un recurso.

Terminado (Terminated): El hilo ha completado su ejecución.

```
class Contador {
    private int contador = 0;

    // Método sincronizado para
evitar que varios hilos modifiquen
la variable simultáneamente
    public synchronized void
incrementar() {
        contador++;
    }

    public int getContador() {
        return contador;
    }
}

class MiHilo extends Thread {
    private Contador contador;

    public MiHilo(Contador contador)
{
        this.contador = contador;
    }

    @Override
    public void run() {
        for (int i = 0; i < 1000; i+
+) {
            contador.incrementar();
        }
    }
}
```

```
public class Main {
    public static void main(String[]
args) throws InterruptedException {
        Contador contador = new
Contador();

        MiHilo hilo1 = new
MiHilo(contador);
        MiHilo hilo2 = new
MiHilo(contador);

        hilo1.start();
        hilo2.start();

        hilo1.join();
        hilo2.join();

        System.out.println("Valor
del contador: " +
contador.getContador());
    }
}
```

8. Métodos Clave

start(): Inicia la ejecución de un hilo. Llama al método run() en un nuevo hilo.

run(): Contiene el código que será ejecutado por el hilo.

sleep(long millis): Suspende el hilo actual por un tiempo determinado.

join(): Hace que un hilo espere a que otro termine su ejecución antes de continuar.

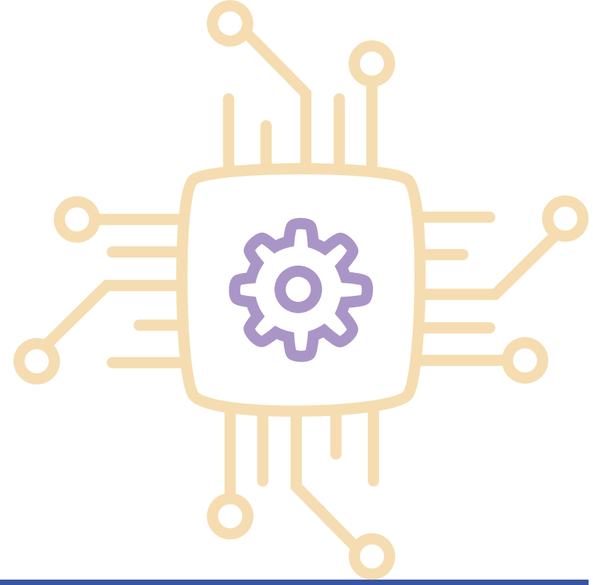
interrupt(): Interrumpe un hilo en ejecución.

```
    @Override
    public void run() {
        for (int i = 0; i < 5; i++)
{
            System.out.println("Hilo
en ejecución: " + i);
            try {
                // Suspende el hilo
durante 1 segundo
                Thread.sleep(1000);
            } catch
(InterruptedException e) {

System.out.println("El hilo fue
interrumpido.");
                return; // Finaliza
si es interrumpido
            }
        }
        System.out.println("El hilo
ha terminado su ejecución.");
    }
}

public class Main {
    public static void main(String[]
args) {
        MiHilo hilo1 = new MiHilo();
        MiHilo hilo2 = new MiHilo();

        // Inicia ambos hilos
        hilo1.start();
        hilo2.start();
```



```
    try {
        // Espera a que el hi
mine
        hilo1.join();
    } catch
terruptedException e) {
        e.printStackTrace();
    }

    // Interrumpe el hilo2 si
 está en ejecución
    if (hilo2.isAlive()) {
        hilo2.interrupt();
    }

    System.out.println("Hilo
ncipal ha terminado.");
}
```

9. Gestión con Executor Framework

Java proporciona el paquete java.util.concurrent para facilitar la creación y gestión de hilos, con clases como ExecutorService, que permite el manejo de grupos de hilos (pools) de manera eficiente.

10. Compatibilidad Multiplataforma

Los hilos en Java son gestionados por la Máquina Virtual de Java (JVM), lo que garantiza que el comportamiento de los hilos sea consistente en diferentes sistemas operativos.

Estas características hacen de los hilos una herramienta poderosa para mejorar el rendimiento de las aplicaciones Java que requieren realizar múltiples tareas en paralelo.

¡Gracias!