

**Tema**

HILOS CON JAVA

INTERRUPCIONES

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU.CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

27/11/2022

HILOS CON JAVA: INTERRUPCIONES

[1. INTRODUCCIÓN 4](#_Toc120696608)

[2. OBJETIVOS 4](#_Toc120696609)

[2.1 OBJETIVO GENERAL 4](#_Toc120696610)

[2.2 OBJETIVOS ESPECÍFICOS 4](#_Toc120696611)

[3 MARCO TEÓRICO 4](#_Toc120696612)

[3.1 HILOS 4](#_Toc120696613)

[3.1.1 HILOS CON JAVA 5](#_Toc120696614)

[3.1.2 INTERRUPCIONES 5](#_Toc120696615)

[3.2 PATRÓN MODELO VISTA CONTROLADOR 5](#_Toc120696616)

[3.2.1 MODELO 6](#_Toc120696617)

[3.2.2 VISTA 6](#_Toc120696618)

[3.2.3 CONTROLADOR 6](#_Toc120696619)

[4 PARTE PRÁCTICA 7](#_Toc120696620)

[4.1 CODIFICACIÓN VIDEO 171 7](#_Toc120696621)

[4.2 CODIFICACIÓN VIDEO 172 8](#_Toc120696622)

[4.3 CODIFICACIÓN FINAL DE LA PRACTICA 9](#_Toc120696623)

[5 FUNCIONALIDAD 14](#_Toc120696624)

[2.4.1 ESTRUCTURA DE LA APLICACIÓN 14](#_Toc120696625)

[2.4.2 EJECUCIÓN DE LA APLICACIÓN 14](#_Toc120696626)

[3. CONCLUSIONES 15](#_Toc120696627)

[4. RECOMENDACIONES 15](#_Toc120696628)

[5. REFERENCIAS 16](#_Toc120696629)

**INDICE DE IMÁGENES**

[Figura 1. Hilos y Procesos 4](#_Toc120667356)

[Figura 2. Logo identificativo de Java 5](#_Toc120667357)

[Figura 3. Interrupciones de hilos 5](#_Toc120667358)

[Figura 4. Modelo Vista Controlador 7](#_Toc120667359)

[Figura 5. Codificación clase HilosVariosControlador 7](#_Toc120667360)

[Figura 6. Codificación clase método main 8](#_Toc120667361)

[Figura 7. Codificación clase HilosVarios2Controlador 8](#_Toc120667362)

[Figura 8. Codificación clase Hilo2Modelo 9](#_Toc120667363)

[Figura 9. Codificación clase HilosVista 9](#_Toc120667364)

[Figura 10. Codificación método main video 172 9](#_Toc120667365)

[Figura 11. Estructura de la aplicación 14](#_Toc120667366)

[Figura 12. Ejecución practica video 171 14](#_Toc120667367)

[Figura 13. Ejecución practica video 172 15](#_Toc120667368)

**ÍNDICE DE TABLAS**

[Tabla 1. Codificación del archivo Main.java 10](file:///C:\Users\santi\Downloads\Informes\videosinterrupciones.docx#_Toc120667369)

[Tabla 2. Codificación del archivo HilosVista.java 11](file:///C:\Users\santi\Downloads\Informes\videosinterrupciones.docx#_Toc120667370)

[Tabla 3. Codificación del archivo Hilo2Modelo 11](file:///C:\Users\santi\Downloads\Informes\videosinterrupciones.docx#_Toc120667371)

[Tabla 4. Codificación del archivo HilosVarios2Controlador.java 12](file:///C:\Users\santi\Downloads\Informes\videosinterrupciones.docx#_Toc120667372)

[Tabla 5. Codificación del archivo HilosVariosControlador.java 13](file:///C:\Users\santi\Downloads\Informes\videosinterrupciones.docx#_Toc120667373)

# INTRODUCCIÓN

Las aplicaciones de software siempre necesitan realizar varias tareas a la vez y no una después de que finalice la otra, esto puede provocar que el sistema y el usuario no sean eficientes al interactuar el uno con el otro. En un flujo de tareas normal una se ejecutará después de la otra, pero lo ideal es mantener varios flujos de ejecución para poder realizar una tarea sin necesidad de esperar a las otras (Nakayama, 2009).

Una aplicación comúnmente debe realizar varias operaciones a la vez como descargar información o contenido, actualizar información del catálogo de productos, sincronizar información con otros usuarios, etc. Los hilos permiten ejecutar varios procesos de forma paralela, tomando en cuenta el acceso a los recursos para que no se provoque una pérdida de información.

# OBJETIVOS

## OBJETIVO GENERAL

Analizar y comprender la interrupción de hilos, para que se usa y como implementarla con el lenguaje de programación JAVA.

## OBJETIVOS ESPECÍFICOS

* Realizar un estudio del estado del arte sobre las interrupciones de hilos.
* Analizar la importancia del uso de interrupciones de hilos.
* Desarrollar un ejemplo práctico para comprender el comportamiento del hilo al interrumpir su funcionamiento.

# MARCO TEÓRICO

## HILOS

Un hilo es un solo flujo de ejecución de instrucciones dentro de un proceso, debido a que no pueden ejecutarse por sí solos, requieren la supervisión de un proceso (Nakayama, 2009). Existen varios lenguajes de programación que permiten crear y gestionar hilos, entre los más conocidos están JAVA y C++.

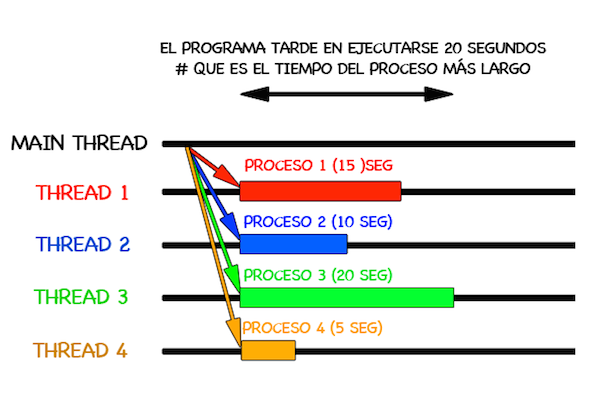


Figura . Hilos y Procesos

### **HILOS CON JAVA**

JAVA es un lenguaje de programación de altor nivel con tipado estático de datos, está en casi todos los sistemas y dispositivos informáticos convirtiéndolo en uno de los lenguajes más usados y demandados de la actualidad. Soporta múltiples paradigmas de programación y ha sido la base para crear lenguajes de programación modernos como SCALA o Kotlin.

El lenguaje de programación JAVA permite crear hilos y administrarlos gracias a su JVM (Java Virtual Machine). Java es capaz de manejar múltiples hilos con varios flujos de instrucciones y un flujo de datos de forma simultánea (Díaz, 2022).



Figura . Logo identificativo de Java

### **INTERRUPCIONES**

En varias ocasiones resulta útil suspender la ejecución de un hilo, debido a que el proceso que maneja ese hilo no es requerido por el usuario o por el sistema. La interrupción de un hilo cambia una bandera para marcar que se ha interrumpido su ejecución (Universidad de Alicante, 2012).

En Java la clase Thread cuenta con varios métodos para gestionar los hilos, entre ellos está interrupt(), este método interrumpe al hilo cambiando el valor de su bandera, cabe mencionar que cada hilo debe ser programado para soportar su propia interrupción.

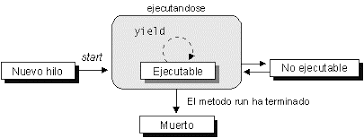


Figura . Interrupciones de hilos

El método join de la clase Thread permite que un determinado hilo finalice su tarea para continuar, bloqueando los procesos hasta que el hilo haya finalizado.

## PATRÓN MODELO VISTA CONTROLADOR

Es un patrón de arquitectura para el desarrollo de aplicaciones software donde su objetivo es el separar la lógica del negocio de la interfaz de usuario lo cual lo hace escalable facilitando la evolución por separado de sus partes, incrementa la reutilización y flexibilidad.

Tuvo su origen en 1979 y es también conocido como patrón MVC, divide una aplicación software en tres grandes partes bien diferenciadas las cuales son Modelo, Vista y Controlador.

### **MODELO**

Es una capa donde se localiza la funcionalidad central y los datos, se comunica con el controlador y la base de datos.

Es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar otras librerías como PDO o algún ORM como Doctrine, que nos permiten trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.

### **VISTA**

Es la capa donde se muestra la información al usuario donde es posible definir una o más vista de usuario en un software, se comunica con el controlador y el usuario.

Las vistas, como su nombre nos hacen entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más tenemos los códigos HTML y PHP que nos permite mostrar la salida.

En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generarán la salida, tal como nuestra aplicación requiera.

### **CONTROLADOR**

Es la capa donde se manejan las entradas del usuario se separa la representación interna de la información y la forma en la que se le muestra al usuario, se comunica con el modelo.

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad, es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

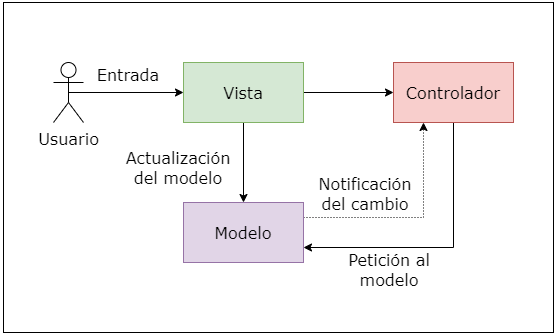


Figura . Modelo Vista Controlador

# PARTE PRÁCTICA

## CODIFICACIÓN VIDEO 171

Para esta práctica se nos propone un caso de estudio el cual nos menciona el siguiente enunciado.

Crear una clase con el método run que ejecute un hilo y lo pause 15 veces, luego usar el método join para para ejecutar un hilo y después otro.

**Paquete Controlador**

El método run hace que el hilo se pause 15 veces por un periodo de tiempo de 700ms, manejando una excepción en caso de errores en tiempo de ejecución, en cada iteración se imprime un mensaje de “Ejecutando hilo” más el nombre del hilo.

Text

Description automatically generated

Figura . Codificación clase HilosVariosControlador

**Main**

En el método main se instancian dos hilos de la clase HilosVariosControlador y se ejecuta uno después del otro, en cada ejecución de los dos hilos se manejan excepciones en caso de errores con los hilos en tiempo de ejecución.

Text

Description automatically generated

Figura . Codificación clase método main

## CODIFICACIÓN VIDEO 172

Para esta práctica se hará uso de la codificación realizada en la sección anterior, se crea una nueva clase para crear un hilo, en su constructor recibe otro parámetro de tipo Thread, se busca dar prioridad para que se ejecute el hilo recibido como argumento en el constructor.

**Paquete Modelo**

Se crea la clase HilosVarios2Controlador, en su constructor recibe un hilo como parámetro, luego se usa el método join para pausar los hilos que se instancien de la clase HilosVariosControlador y se ejecute el hilo que se recibe como parámetro.

Text

Description automatically generated

Figura . Codificación clase HilosVarios2Controlador

Se define una clase Hilo2Modelo, esta contendrá un atributo de tipo Thread llamado hilo1.

Text

Description automatically generated

Figura . Codificación clase Hilo2Modelo

**Paquete Vista**

En el método main se definen los objetos hilo1 de la clase HilosVariosControlador e hilo2 de la clase HilosVariso2, el hilo2 recibe como argumento a hilo1, luego se ejecuta el hilo2 y por último el hilo2.

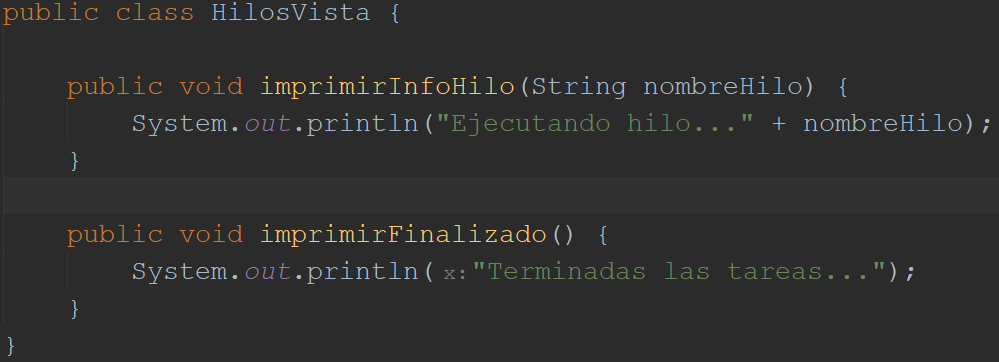


Figura . Codificación clase HilosVista

**Main**

En el método main se instancias varios objetos de las clases definidas en los paquetes de MVC para definir el proceso de ejecución del hilo1 e hilo2.

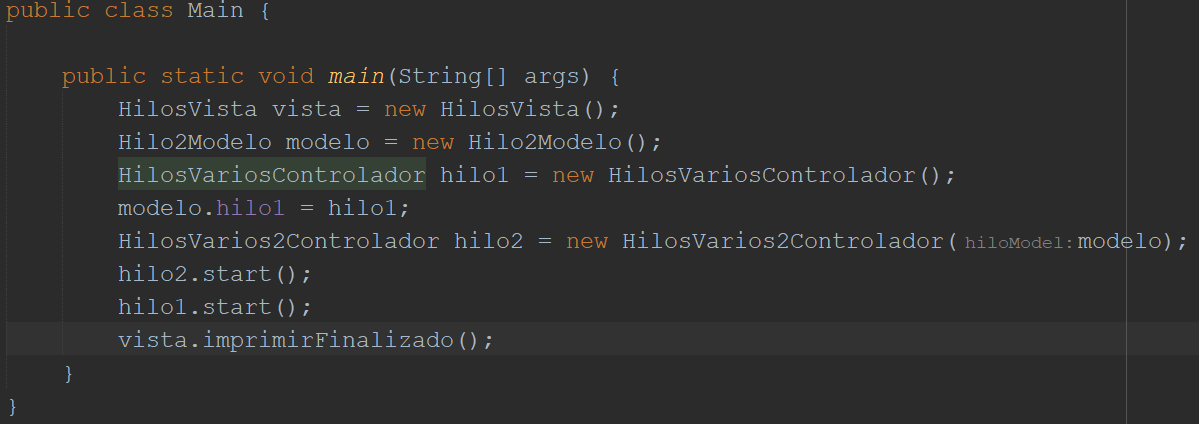


Figura . Codificación método main video 172

## CODIFICACIÓN FINAL DE LA PRACTICA

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template

\*/

package ec.edu.monster;

import ec.edu.monster.controlador.HilosVarios2Controlador;

import ec.edu.monster.controlador.HilosVariosControlador;

import ec.edu.monster.modelo.Hilo2Modelo;

import ec.edu.monster.vista.HilosVista;

/\*\*

\*

\* @author Grupo5

\*/

public class Main {

public static void main(String[] args) {

HilosVista vista = new HilosVista();

Hilo2Modelo modelo = new Hilo2Modelo();

HilosVariosControlador hilo1 = new HilosVariosControlador();

modelo.hilo1 = hilo1;

HilosVarios2Controlador hilo2 = new HilosVarios2Controlador(modelo);

hilo2.start();

hilo1.start();

vista.imprimirFinalizado();

}

}

Tabla . Codificación del archivo Main.java

Tabla . Codificación del archivo HilosVista.java

package ec.edu.monster.vista;

/\*\*

\*

\* @author Grupo5

\*/

public class HilosVista {

public void imprimirInfoHilo(String nombreHilo) {

System.out.println("Ejecutando hilo..." + nombreHilo);

}

public void imprimirFinalizado() {

System.out.println("Terminadas las tareas...");

}

}

package ec.edu.monster.modelo;

/\*\*

\*

\* @author Grupo5

\*/

public class Hilo2Modelo {

public Thread hilo1;

}

Tabla . Codificación del archivo Hilo2Modelo

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.Hilo2Modelo;

import ec.edu.monster.vista.HilosVista;

public class HilosVarios2Controlador extends Thread {

private Hilo2Modelo hiloModel;

public HilosVarios2Controlador(Hilo2Modelo hiloModel) {

this.hiloModel = hiloModel;

}

public void run() {

HilosVista vista = new HilosVista();

try {

hiloModel.hilo1.join();

} catch (Exception e) {

e.printStackTrace();

}

for (int i = 0; i < 5; i++) {

vista.imprimirInfoHilo(getName());

try {

Thread.sleep(700);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

Tabla . Codificación del archivo HilosVarios2Controlador.java

package ec.edu.monster.controlador;

import ec.edu.monster.vista.HilosVista;

public class HilosVariosControlador extends Thread {

public void run() {

HilosVista vista = new HilosVista();

for (int i = 0; i < 5; i++) {

vista.imprimirInfoHilo(getName());

try {

Thread.sleep(700);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

Tabla . Codificación del archivo HilosVariosControlador.java

## FUNCIONALIDAD

### **ESTRUCTURA DE LA APLICACIÓN**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura . Estructura de la aplicación

### **EJECUCIÓN DE LA APLICACIÓN**

**Ejecución Video 171**

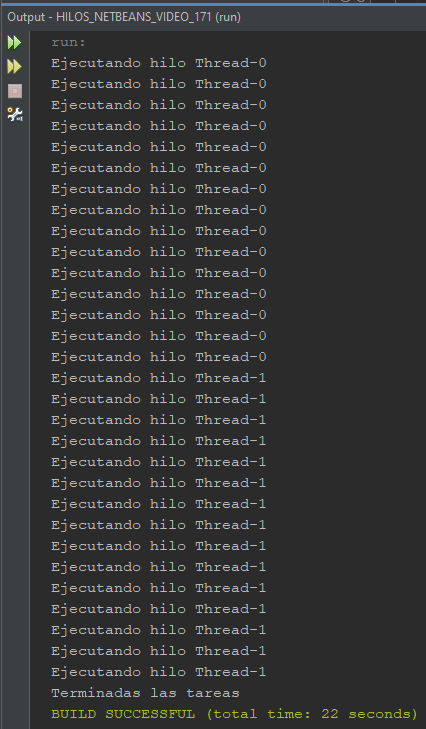


Figura . Ejecución practica video 171

**Ejecución Video 172**

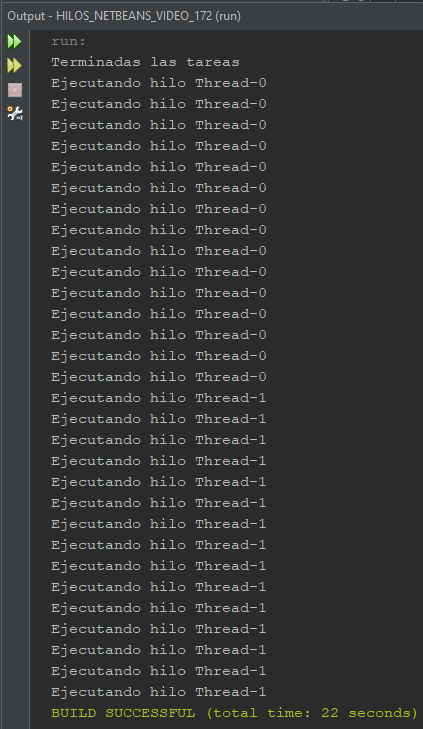


Figura . Ejecución practica video 172

# CONCLUSIONES

* Las interrupciones permiten que un hilo no se ejecute, esto en la práctica ayuda a que se desactiven ciertas funcionalidades del sistema que son gestionadas por hilos y que el usuario o el sistema no requieren para aumentar el rendimiento y capacidad de procesamiento de otras tareas.
* La función join() de la clase Thread permite que un hilo se ejecute después de que el otro hilo finalice su ejecución, de esta manera se pude controlar el orden de ejecución de los hilos para que no se presenten inconsistencias en las salidas por cambios inesperados en el flujo de información.

# RECOMENDACIONES

* Se recomienda tener cuidado con los bucles infinitos y errores, al definir varios hilos que se ejecuten de forma simultánea puede provocar que el sistema operativo falle se cierre el programa de forma inesperada.
* Cuando se usen hilos siempre se deben manejar las excepciones correspondientes para evitar errores en tiempo de ejecución.

# REFERENCIAS

Blancarte, O. (29 de Marzo de 2017). *Concurrencia vs Paralelismo*. Obtenido de oscarblancarteblog: https://www.oscarblancarteblog.com/2017/03/29/concurrencia-vs-paralelismo/

Díaz, J. (2022). *Píldoras Informáticas*. Obtenido de YouTube: https://www.youtube.com/playlist?list=PLU8oAlHdN5BktAXdEVCLUYzvDyqRQJ2lk

Nakayama, A. (2009). Guía práctica de estudio 12: Hilos. México.

Universidad de Alicante. (17 de Octubre de 2012). *Introdoucción al lenguaje Java y Eclipse*. Obtenido de JTECH: http://www.jtech.ua.es/dadm/restringido/java/sesion05-apuntes.html#:~:text=Interrupci%C3%B3n%20de%20un%20hilo,-Los%20objetos%20de&text=interrupt()%20que%20permite%20al,para%20soportar%20su%20propia%20interrupci%C3%B3n.