

**Tema**

HILOS JAVA (171-172)

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU.CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

10/07/2023

DOMINIOS DE SEGURIDAD EN APLICACIONES JEE

[1. MARCO TEÓRICO 4](#_Toc120481017)

[1.1. OBJETIVOS 4](#_Toc120481018)

[1.2.1. OBJETIVO GENERAL 4](#_Toc120481019)

[1.2.2. OBJETIVOS ESPECÍFICOS 4](#_Toc120481020)

[1.2. JAVA. 4](#_Toc120481021)

[1.3. APACHE NETBEANS 4](#_Toc120481022)

[1.4. CONCURRENCIA 5](#_Toc120481023)

[1.5. HILOS 5](#_Toc120481024)

[1.6. HILOS PARALELOS Y CONCURRENTES 5](#_Toc120481025)

[1.7. TERMINOLOGÍA Y METODOS DE HILOS EN JAVA. 5](#_Toc120481026)

[2. PARTE PRÁCTICA 7](#_Toc120481027)

[2.1. Video 171 7](#_Toc120481028)

[2.1.1 CREACIÓN DEL PROYECTO. 7](#_Toc120481029)

[2.1.2 Creación de paquetes MVC. 8](#_Toc120481030)

[2.1.3 CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA 10](#_Toc120481031)

[2.1.4. CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO 12](#_Toc120481032)

[2.1.5. CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR 16](#_Toc120481033)

[2.1.6. CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN 18](#_Toc120481034)

[2.1.7. EJECUCIÓN DEL PROYECTO. 20](#_Toc120481035)

[2.2. Video 172 21](#_Toc120481036)

[2.1.1 CREACIÓN DEL PROYECTO. 21](#_Toc120481037)

[2.1.4 Creación de paquetes MVC. 22](#_Toc120481038)

[2.1.5 CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA 23](#_Toc120481039)

[2.1.8. CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO 26](#_Toc120481040)

[2.1.9. CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR 28](#_Toc120481041)

[2.1.10. CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN 32](#_Toc120481042)

[2.1.11. EJECUCIÓN DEL PROYECTO. 34](#_Toc120481043)

[3. CONCLUSIONES 35](#_Toc120481044)

[4. RECOMENDACIONES 35](#_Toc120481045)

[5. BIBLIOGRAFIA 35](#_Toc120481046)

**INDICE DE IMÁGENES**

[Figura 1. Logotipo identificativo de Java 4](#_Toc120481112)

[Figura 2. Logotipo identificativo de Netbeans 7](#_Toc120481113)

[Figura 3. Selección del tipo de aplicación a desarrollar 7](#_Toc120481114)

[Figura 4. Nombre del proyecto 8](#_Toc120481115)

[Figura 5. Estructura del proyecto generado 8](#_Toc120481116)

[Figura 6. Creación de paquete. 9](#_Toc120481117)

[Figura 7. Selección del nombre de los paquetes del video 171 9](#_Toc120481118)

[Figura 8. Creación de clase Java. 10](#_Toc120481119)

[Figura 9. Codificación archivo HilosVista. 10](#_Toc120481120)

[Figura 10. Creación de clase Java. 12](#_Toc120481121)

[Figura 11. Codificación archivo HilosModelo. 12](#_Toc120481122)

[Figura 12. Creación de clase Java. 14](#_Toc120481123)

[Figura 13. Codificación archivo HilosVarios 14](#_Toc120481124)

[Figura 14. Creación de clase Java. 16](#_Toc120481125)

[Figura 15.Codificación archivo HilosControlador. 16](#_Toc120481126)

[Figura 16. Codificación archivo principal. 18](#_Toc120481127)

[Figura 17. Resultado de la ejecución 20](#_Toc120481128)

[Figura 18. Logotipo identificativo de Netbeans 21](#_Toc120481129)

[Figura 19. Selección del tipo de aplicación a desarrollar 21](#_Toc120481130)

[Figura 20. Nombre del proyecto 22](#_Toc120481131)

[Figura 21. Estructura del proyecto generado. 22](#_Toc120481132)

[Figura 22. Creación de paquete. 23](#_Toc120481133)

[Figura 23. Selección del nombre de los paquetes del video 172 23](#_Toc120481134)

[Figura 24. Creación de clase Java 24](#_Toc120481135)

[Figura 25. Codificación archivo HilosVista 24](#_Toc120481136)

[Figura 26. Creación de clase Java 26](#_Toc120481137)

[Figura 27. Codificación archivo HilosModelo 26](#_Toc120481138)

[Figura 28. Creación de clase Java. 28](#_Toc120481139)

[Figura 29. Codificación archivo HilosVariosControlador 28](#_Toc120481140)

[Figura 30. Creación de clase Java. 30](#_Toc120481141)

[Figura 31. Codificación archivo HilosVarios2Controlador 30](#_Toc120481142)

[Figura 32. Codificación archivo principal 32](#_Toc120481143)

[Figura 33. Resultado de la ejecución. 34](#_Toc120481144)

**ÍNDICE DE TABLAS**

[Tabla 1. Terminología utilizada en Hilos Java 5](#_Toc120481057)

[Tabla 2. Código de HilosVista. 11](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481058)

[Tabla 3. Codificación del archivo HilosModelo 13](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481059)

[Tabla 4. Codificación del archivo HilosVarios 15](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481060)

[Tabla 5. Codificación del archivo HilosControlador 17](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481061)

[Tabla 6.Codificación del archivo principal del proyecto 19](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481062)

[Tabla 7. Código de HilosVista. 25](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481063)

[Tabla 8. Codificación del archivo HilosModelo 27](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481064)

[Tabla 9. Codificación del archivo Controlador 29](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481065)

[Tabla 10. Codificación del archivo HilosVarios2Controlador 31](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481066)

[Tabla 11. Codificación del archivo principal del proyecto 33](file:///C:\Users\rikye\Desktop\HilosJava_Video172_grupo6.docx#_Toc120481067)

# MARCO TEÓRICO

## OBJETIVOS

### OBJETIVO GENERAL

Comprender la funcionalidad de los hilos en Java para ejecutar multiples tareas al mismo así como también ver la manera en la que un programa realizado con hilos actúa de manera diferente a uno que no los utiliza.

### OBJETIVOS ESPECÍFICOS

* Analizar le proceso de los hilos en java.
* Conocer la forma de utilizar los hilos en el lenguaje de programación de java.
* Conocer la forma de ejecutar múltiples hilos al mismo tiempo y a su vez detenerlos.

## JAVA.

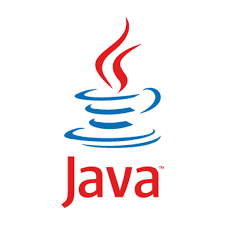


Figura 1. Logotipo identificativo de Java

Java es según la definición dada por [1] es un lenguaje de programación ampliamente utilizado para codificar aplicaciones de distintos indoles, desde aplicaciones de escritorio hasta tipo web, pasando por desarrollo de videojuegos, macrodatos, IA e IoT. Actualmente la ultima versión numero 8 y cuenta con 3 versiones disponibles de Java las cuales son Java SE, Java EE y Java ME.

La mayor fortaleza de java es su capacidad de ser un lenguaje sumamente apto para ser aplicado en el paradigma orientado a objetos, lo cual es su mayor fortaleza en este aspecto.

## APACHE NETBEANS

Apache netbeans es [2] un entorno de desarrollo integrado (IDE) creado para principalmente el lenguaje de programación de Java, anteriormente conocido simplemente como netbeans, cambio su nombre a apache netbeans y actualmente se encuentra en la versión 15.

Este IDE ofrece una serie de plugins que pueden ser instalados para facilitar el trabajo al momento de desarrollar aplicaciones y ejecutarlas, como por ejemplo poder instalar de manera rápida y fácil el servidor de payara o glassfish.

Cabe recalcar que apache netbeans es completamente gratis para su uso tanto para fines empresariales como educativos, y que se puede encontrar fácilmente para su descarga en el sitio oficial, teniendo versiones para los sistemas operativos Windows, Mac, Linux y Solaris.

## CONCURRENCIA

La concurrencia es [3] la ejecución de varios procesos a la vez, es decir, es la ejecución simultánea de múltiples tareas interactivamente. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa. Las tareas se pueden ejecutar en una sola CPU (multiprogramación), en varios procesadores, o en una red de computadores distribuidos.

## HILOS

Un hilo es [3] un flujo de control dentro de un programa. Creando varios hilos podremos realizar varias tareas simultáneamente. Cada hilo tendrá sólo un contexto de ejecución (contador de programa, pila de ejecución). Es decir, a diferencia de los procesos UNIX, no tienen su propio espacio de memoria, sino que acceden todos al mismo espacio de memoria común, por lo que será importante su sincronización cuando tengamos varios hilos accediendo a los mismos objetos.

Los hilos son útiles porque permiten que el flujo del programa sea divido en dos o más partes, cada una ocupándose de alguna tarea de forma independiente. Por ejemplo un hilo puede encargarse de la comunicación con el usuario, mientras que otros actúan en segundo plano, realizando la transmisión de un fichero, accediendo a recursos del sistema (cargar sonidos, leer ficheros ...), etc.

## HILOS PARALELOS Y CONCURRENTES

Un hilo como se indicó con anterioridad se utiliza sobre todo para dividir tareas y hacerla de manera más eficiente, además como indica [4] permitiendo que estas se ejecuten de manera simultánea en vez de terminar una para continuar otra.

De manera más formal [5] “cada hilo progresa de forma independiente al resto. Esto provoca que cada uno de los hilos pueda potencialmente viajar a una velocidad distinta, ejecutándose "concurrentemente" con el resto. Esto provoca que la ejecución de un programa sea diferente en cada da pasada y que sea independiente.”

## TERMINOLOGÍA Y METODOS DE HILOS EN JAVA.

Antes de realizar la practica presente es necesario conocer algunos términos que son necesarios para entender los códigos y algunas definiciones que se utilizaran durante la práctica.

Tabla 1. Terminología utilizada en Hilos Java

|  |  |
| --- | --- |
| TÉRMINO O METODO | DESCRIPCIÓN |
| isInterrupted() | Método que verifica si un hilo esta interrumpido o no. |
| Concurrencia | En programación significa cuando tareas se ejecutan de manera simultanea. |
| resume() | Nombre o etiqueta que clasifica a los usuarios con un conjunto de características comunes que generalmente conduce a un conjunto de permisos comunes. Los grupos pueden definirse ampliamente y reflejar los grupos de la vida real como "admin" o "manager”. |
| join() | Al utilizar este método nos aseguramos que no se ejecute otro hilo hasta que éste finalice |
| sleep(long ms) | Pone el hilo en pausa el tiempo en milisegundos que le introduzcamos |
| yield() | Pausa el hilo en ejecución para permitir la ejecución de otros |
| start() | Indica al intérprete de Java que cree un contexto del hilo del sistema y comience a ejecutarlo |
| IDE | Entorno de desarrollo integrado (IDE), es un software que permite desarrollar aplicaciones con herramientas que ayudan al programador a organizar su flujo de trabajo y agilizando todo el proceso de diseño de software, a través de una interfaz gráfica. |
| stop() | Provoca que el hilo se detenga de manera inmediata |
| resume() | Reanuda un hilo suspendido |
| setPriority() | Asigna la prioridad al hilo indicada por el valor pasado como parámetro |
| getPriority() | Devuelve la prioridad del hilo de ejecución en curso |
| setName() | Asigna un nombre para identificar a los hilos de una forma más cómoda |
| getName() | Devuelve el nombre asignado |

# PARTE PRÁCTICA

## Video 171

### 2.1.1 CREACIÓN DEL PROYECTO.

Abra el IDE de desarrollo Apache Netbeans 14.

Imagen que contiene Forma

Descripción generada automáticamente

Figura 2. Logotipo identificativo de Netbeans

Primero dentro del disco C y cree una carpeta llamada DISTRIBUIDAS allí crea otra subcarpeta llamada FORMULARIOSWEB, dentro de esta crea otra subcarpeta con el nombre de HilosJava a su vez dentro de esta última cree 2 carpetas, una con el nombre de “aplicativo” y otra con el nombre de “documentación”. Una vez haya creado las carpetas abre Apache Netbeans y se sitúa en la Pestaña superior y selecciona File->New Project -> Java with Ant -> Java Application

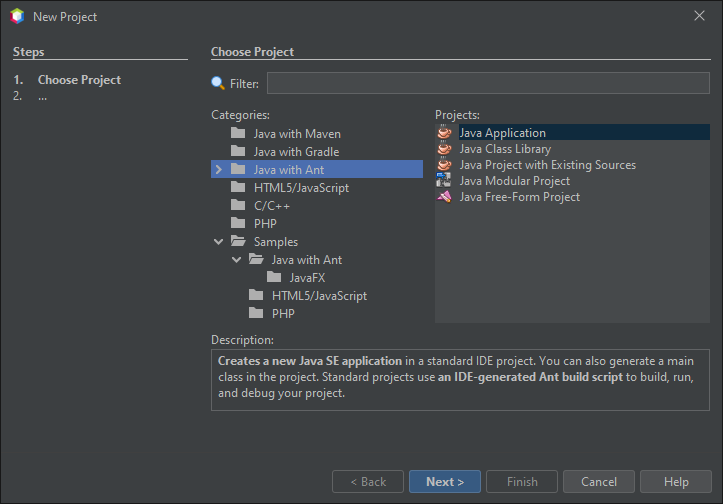


Figura 3. Selección del tipo de aplicación a desarrollar

Le da el siguiente nombre G, y en la opción que dice localización del proyecto presiona buscar y selecciona la carpeta “aplicativo” que fue creada con anterioridad, en el grupo Id “ec.edu.monster.main” para tener un elemento principal que invoque inicialmente a los demás elementos presentes en el modelo de arquitectura MVC.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Figura 4. Nombre del proyecto

De clic en finalizar y Verifique que la estructura del proyecto se encuentre de la siguiente manera.

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media**

Figura 5. Estructura del proyecto generado

### Creación de paquetes MVC.

Para aplicar la arquitectura MVC es necesario crear 3 paquetes distintos, cada paquete representara a un elemento de la arquitectura, es decir que existirá un paquete visto, uno controlador y uno de modelo, para ello siga los siguientes pasos.

De clic en sources packages->new->java package

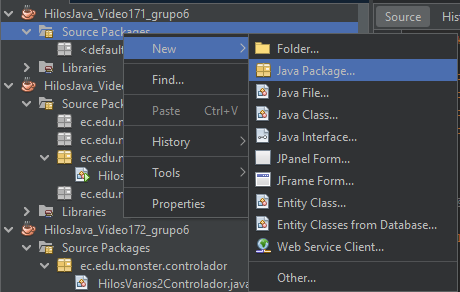


Figura 6. Creación de paquete.

En el nombre del paquete coloque ec.edu.monster.(modelo-vista-controlador) según requiera.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 7. Selección del nombre de los paquetes del video 171

Repita el proceso hasta tener los 3 paquetes creados

### CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA

De clic derecho en el paquete vista, luego nuevo y finalmente clase Java.

Captura de pantalla de computadora

Descripción generada automáticamente

Figura 8. Creación de clase Java.

Le da el nombre de “HilosVista”, y proceda a colocar el siguiente código para construir su primera clase en el paquete de vista, que en este caso solo ocuparemos la consola para mostrar los resultados.



Figura 9. Codificación archivo HilosVista.

.

Tabla 2. Código de HilosVista.

package ec.edu.monster.vista;

import javax.swing.JButton;

import javax.swing.JFrame;

import static javax.swing.JFrame.EXIT\_ON\_CLOSE;

import javax.swing.JLabel;

import javax.swing.JPanel;

public class HiloVista extends JFrame{

public HiloVista(){

this.setSize(300,300);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setLocationRelativeTo(null);

setTitle("Sicronizacion Hilos");

jpanel=new JPanel();

jpanel.setLayout(null);

this.getContentPane().add(jpanel);

jlabel=new JLabel();

jbtnIniciar=new JButton("Iniciar Hilos");

jbtnIniciar.setBounds(100,100,100,30);

jpanel.add(jbtnIniciar);

}

public JPanel jpanel;

public JLabel jlabel;

public JButton jbtnIniciar;

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO

De clic derecho en el paquete modelo, luego nuevo y finalmente clase Java.

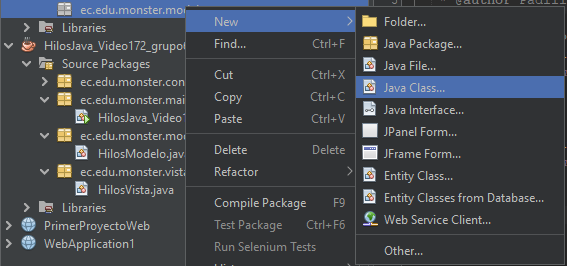


Figura 10. Creación de clase Java.

Le da el nombre de “HilosModelo”, y proceda a colocar el siguiente código para construir su primera clase en el paquete de modelo.

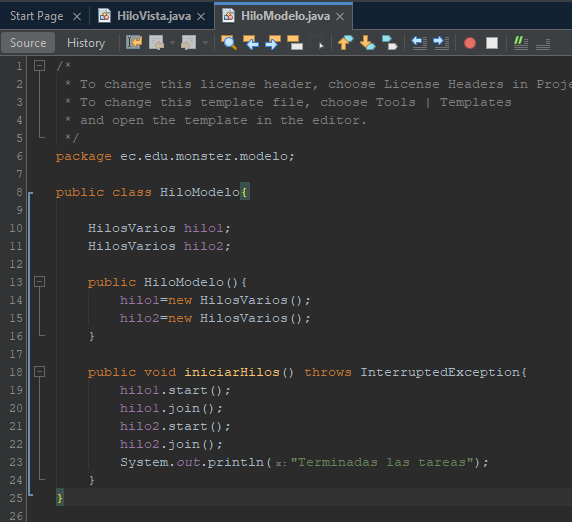


Figura 11. Codificación archivo HilosModelo.

Tabla 3. Codificación del archivo HilosModelo

package ec.edu.monster.modelo;

public class HiloModelo{

HilosVarios hilo1;

HilosVarios hilo2;

public HiloModelo(){

hilo1=new HilosVarios();

hilo2=new HilosVarios();

}

public void iniciarHilos() throws InterruptedException{

hilo1.start();

hilo1.join();

hilo2.start();

hilo2.join();

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

}

}

Cree una nueva clase en el paquete del modelo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 12. Creación de clase Java.

Le da el nombre de “HilosVarios”, y proceda a colocar el siguiente código para construir la clase en el paquete de controlador.

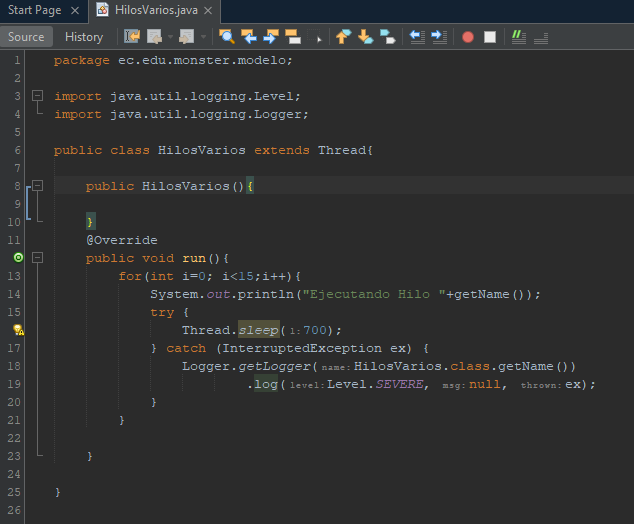


Figura 13. Codificación archivo HilosVarios

Tabla 4. Codificación del archivo HilosVarios

package ec.edu.monster.modelo;

import java.util.logging.Level;

import java.util.logging.Logger;

public class HilosVarios extends Thread{

public HilosVarios(){

}

@Override

public void run(){

for(int i=0; i<15;i++){

System.out.println("Ejecutando Hilo "+getName());

try {

Thread.sleep(700);

} catch (InterruptedException ex) {

Logger.getLogger(HilosVarios.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR

De clic derecho en el paquete modelo, luego nuevo y finalmente clase Java.

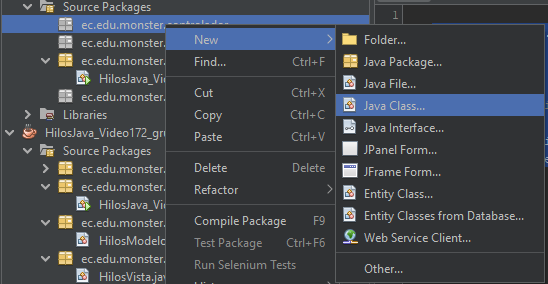


Figura 14. Creación de clase Java.

Le da el nombre de “HilosControlador”, y proceda a colocar el siguiente código para construir su primera clase en el paquete de controlador.

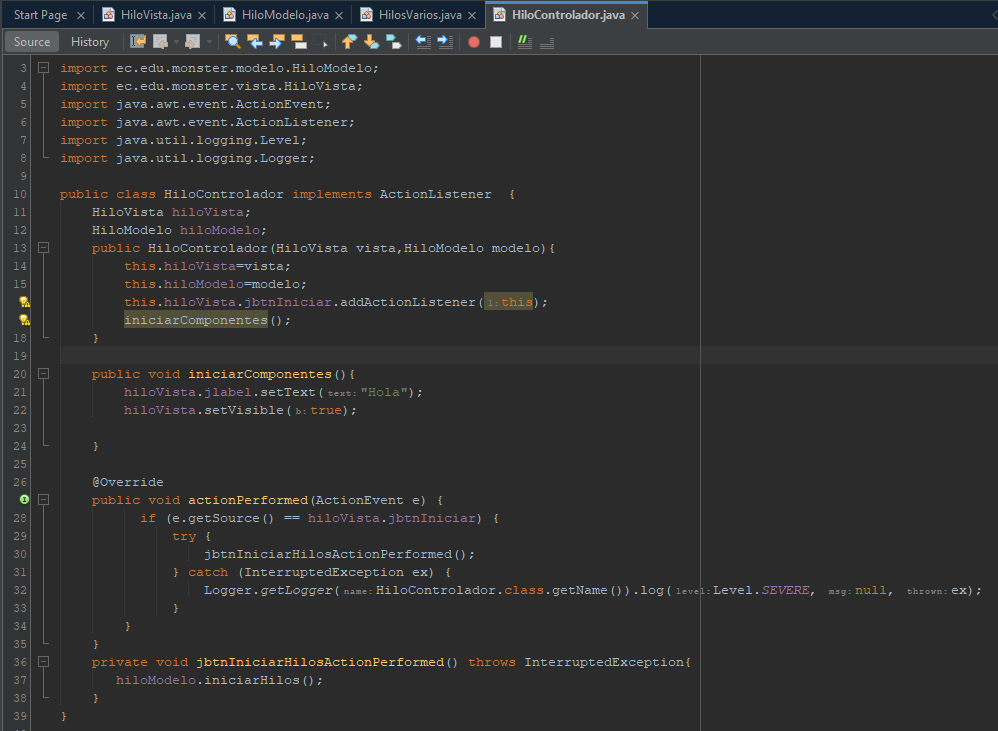


Figura 15.Codificación archivo HilosControlador.

Tabla 5. Codificación del archivo HilosControlador

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.HiloModelo;

import ec.edu.monster.vista.HiloVista;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.logging.Level;

import java.util.logging.Logger;

public class HiloControlador implements ActionListener {

HiloVista hiloVista;

HiloModelo hiloModelo;

public HiloControlador(HiloVista vista,HiloModelo modelo){

this.hiloVista=vista;

this.hiloModelo=modelo;

this.hiloVista.jbtnIniciar.addActionListener(this);

iniciarComponentes();

}

public void iniciarComponentes(){

hiloVista.jlabel.setText("Hola");

hiloVista.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == hiloVista.jbtnIniciar) {

try {

jbtnIniciarHilosActionPerformed();

} catch (InterruptedException ex) {

Logger.getLogger(HiloControlador.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

private void jbtnIniciarHilosActionPerformed() throws InterruptedException{

hiloModelo.iniciarHilos();

}

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN

Abra el archivo que fue creado por defecto dentro del paquete main y proceda a colocar el siguiente código para construir su primera clase en el paquete de controlador.

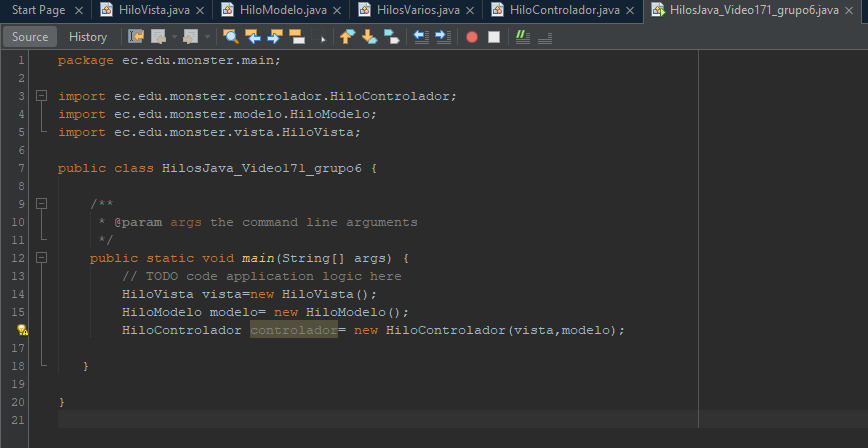


Figura 16. Codificación archivo principal.

Tabla 6.Codificación del archivo principal del proyecto

package ec.edu.monster.main;

import ec.edu.monster.controlador.HiloControlador;

import ec.edu.monster.modelo.HiloModelo;

import ec.edu.monster.vista.HiloVista;

public class HilosJava\_Video171\_grupo2 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

HiloVista vista=new HiloVista();

HiloModelo modelo= new HiloModelo();

HiloControlador controlador= new HiloControlador(vista,modelo);

}

}

### EJECUCIÓN DEL PROYECTO.

Ejecute el proyecto y podrá observar como puede lanzar varias pelotas en diferentes momentos, siendo cada pelota una representación de un hilo.

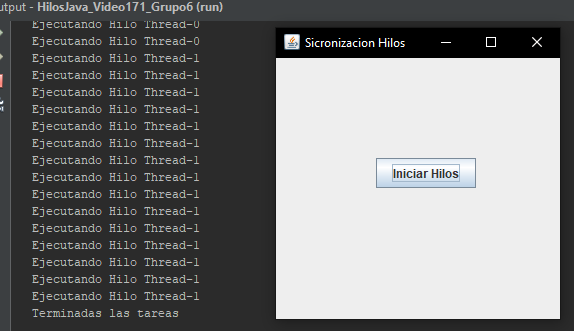


Figura 17. Resultado de la ejecución

## Video 172

### 2.1.1 CREACIÓN DEL PROYECTO.

Abra el IDE de desarrollo Apache Netbeans 14.



Figura 18. Logotipo identificativo de Netbeans

Primero dentro del disco C y cree una carpeta llamada DISTRIBUIDAS allí crea otra subcarpeta llamada FORMULARIOSWEB, dentro de esta crea otra subcarpeta con el nombre de HilosJava a su vez dentro de esta última cree 2 carpetas, una con el nombre de “aplicativo” y otra con el nombre de “documentación”. Una vez haya creado las carpetas abre Apache Netbeans y se sitúa en la Pestaña superior y selecciona File->New Project -> Java Application

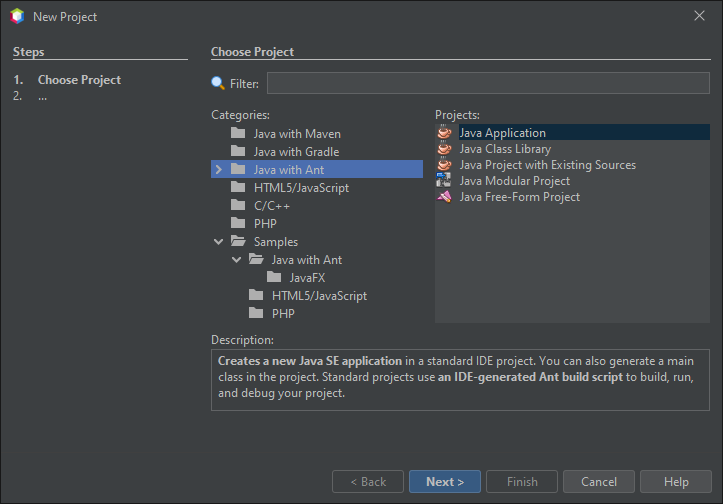


Figura 19. Selección del tipo de aplicación a desarrollar

Le da el siguiente nombre G, y en la opción que dice localización del proyecto presiona buscar y selecciona la carpeta “aplicativo” que fue creada con anterioridad, en el grupo Id “ec.edu.monster.main” para tener un elemento principal que invoque inicialmente a los demás elementos presentes en el modelo de arquitectura MVC.

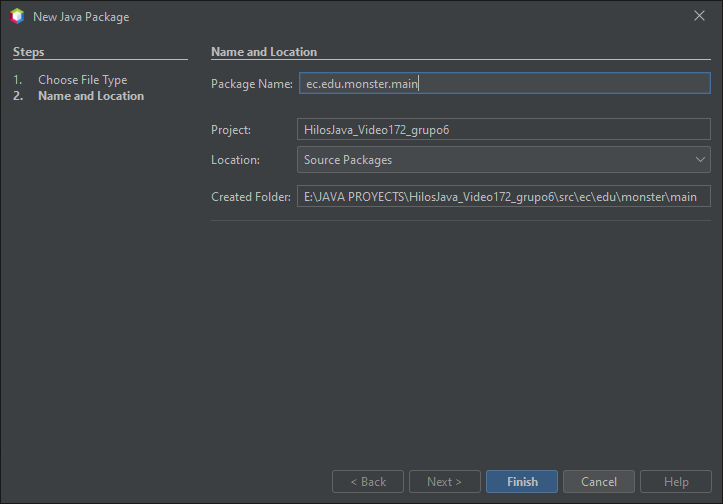
****

Figura 20. Nombre del proyecto

De clic en finalizar y Verifique que la estructura del proyecto se encuentre de la siguiente manera.

**Texto

Descripción generada automáticamente**

Figura 21. Estructura del proyecto generado.

### Creación de paquetes MVC.

Para aplicar la arquitectura MVC es necesario crear 3 paquetes distintos, cada paquete representara a un elemento de la arquitectura, es decir que existirá un paquete visto, uno controlador y uno de modelo, para ello siga los siguientes pasos.

De clic en sources packages->new->java package

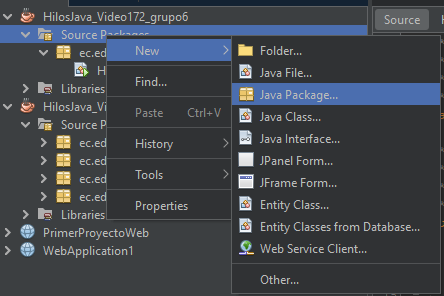


Figura 22. Creación de paquete.

En el nombre del paquete coloque ec.edu.monster.(modelo-vista-controlador) según requiera.

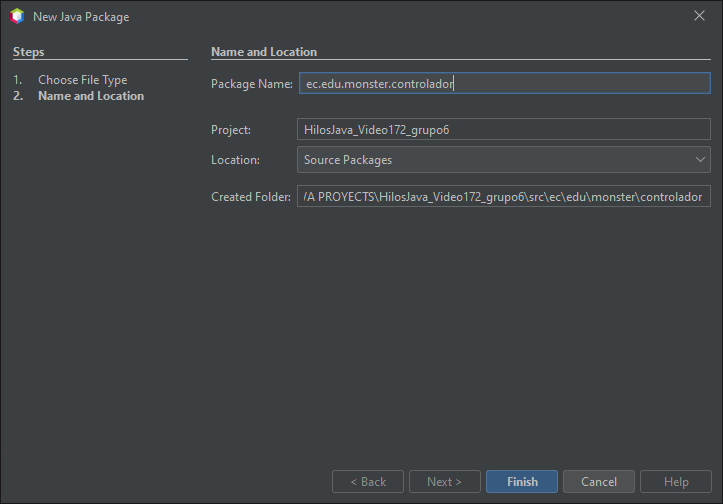


Figura 23. Selección del nombre de los paquetes del video 172

Repita el proceso hasta tener los 3 paquetes creados

### CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA

De clic derecho en el paquete vista, luego nuevo y finalmente clase Java.

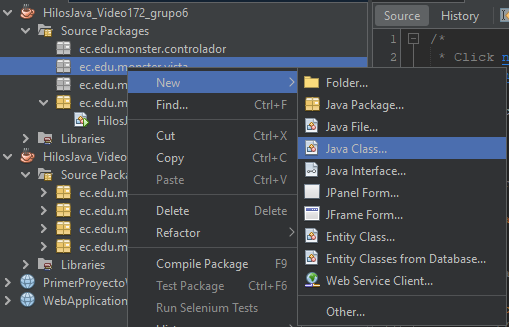


Figura 24. Creación de clase Java

Le da el nombre de “HilosVista”, y proceda a colocar el siguiente código para construir su primera clase en el paquete de vista, que en este caso solo ocuparemos la consola para mostrar los resultados.

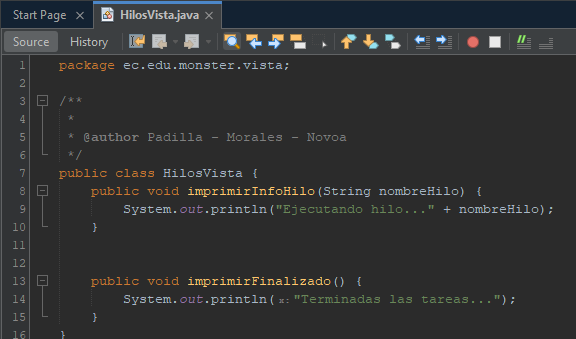


Figura 25. Codificación archivo HilosVista

Tabla 7. Código de HilosVista.

package ec.edu.monster.vista;

/\*\*

\*

\* @author Mosquera – Pallango – Sánchez

\*/

public class HilosVista {

public void imprimirInfoHilo(String nombreHilo) {

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

}

public void imprimirFinalizado() {

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

}

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO

De clic derecho en el paquete modelo, luego nuevo y finalmente clase Java.

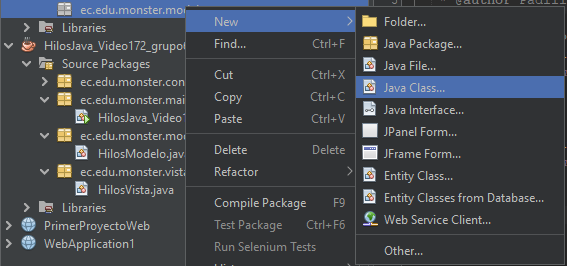


Figura 26. Creación de clase Java

Le da el nombre de “HilosModelo”, y proceda a colocar el siguiente código para construir su primera clase en el paquete de modelo.

Texto

Descripción generada automáticamente

Figura 27. Codificación archivo HilosModelo

Tabla 8. Codificación del archivo HilosModelo

package ec.edu.monster.modelo;

/\*\*

\*

\* @author Mosquera – Pallango – Sánchez

\*/

public class HilosModelo {

public Thread hilo1;

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR

De clic derecho en el paquete modelo, luego nuevo y finalmente clase Java.

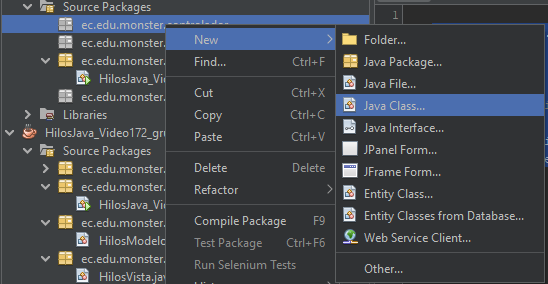


Figura 28. Creación de clase Java.

Le da el nombre de “HilosVariosControlador”, y proceda a colocar el siguiente código para construir su primera clase en el paquete de controlador.

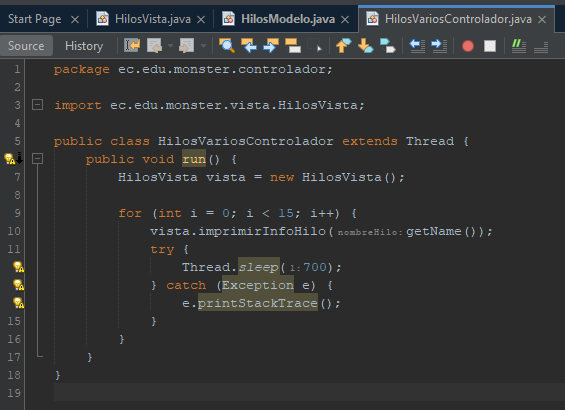


Figura 29. Codificación archivo HilosVariosControlador

Tabla 9. Codificación del archivo Controlador

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 < 15; i++) {

vista.imprimirInfoHilo(getName());

try {

Thread.sleep(700);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

Cree una nueva clase en el paquete del controlador.

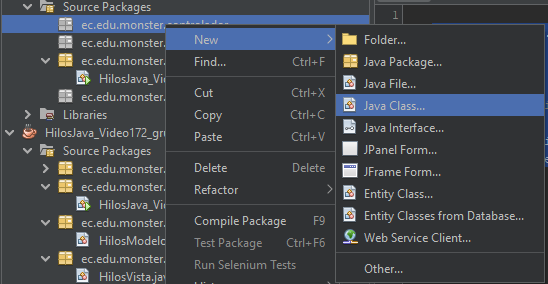


Figura 30. Creación de clase Java.

Le da el nombre de “HilosVarios2Controlador”, y proceda a colocar el siguiente código para construir la clase en el paquete de controlador.

Texto

Descripción generada automáticamente

Figura 31. Codificación archivo HilosVarios2Controlador

Tabla 10. Codificación del archivo HilosVarios2Controlador

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.HilosModelo;

import ec.edu.monster.vista.HilosVista;

public class HilosVarios2Controlador extends Thread {

private HilosModelo hiloModel;

public HilosVarios2Controlador(HilosModelo 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 < 15; i++) {

vista.imprimirInfoHilo(getName());

try {

Thread.sleep(700);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN

Abra el archivo que fue creado por defecto dentro del paquete main y proceda a colocar el siguiente código para construir su primera clase en el paquete de controlador.

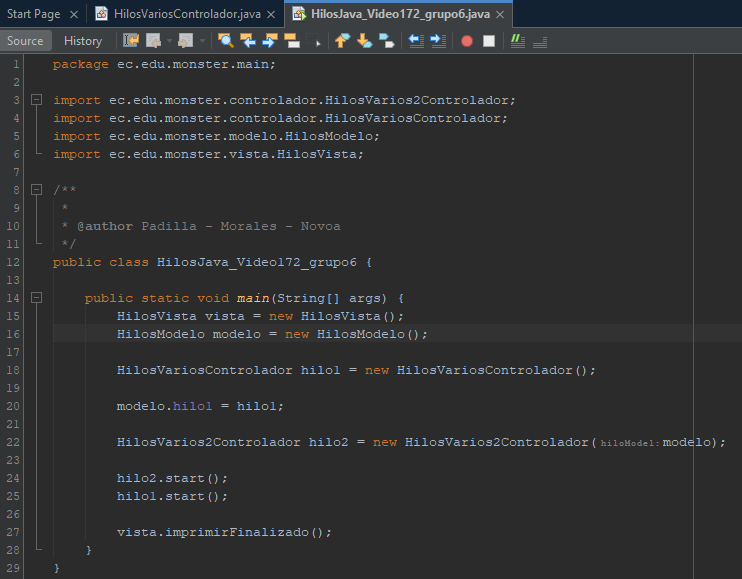


Figura 32. Codificación archivo principal

Tabla 11. Codificación del archivo principal del proyecto

/\*

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

\* Click nbfs://nbhost/SystemFileSystem/Templates/Project/Maven2/JavaApp/src/main/java/${packagePath}/${mainClassName}.java to edit this template

\*/

package ec.edu.monster.main;

import ec.edu.monster.controlador.Controlador;

import ec.edu.monster.modelo.PelotaHilos;

import ec.edu.monster.vista.Vista;

/\*\*

\*

\* @author USER

\*/

public class HilosJava\_Video168\_grupo2 {

public static void main(String[] args) {

Vista vista=new Vista();

PelotaHilos modelo=new PelotaHilos();

Controlador controlador=new Controlador(vista,modelo);

}

}

### EJECUCIÓN DEL PROYECTO.

Ejecute el proyecto y podrá observar como puede lanzar varias pelotas en diferentes momentos, siendo cada pelota una representación de un hilo.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 33. Resultado de la ejecución.

# CONCLUSIONES

* Los Hilos en java nos permite generar programas capaces de realizar múltiples tareas al mismo tiempo que pueden aprovechar todas las capacidades de procesamiento de un computador.
* Java nos permite generar proyectos con la capacidad de utilizar hilos de manera fácil y rápida ya que incluye métodos y bibliotecas propias.
* Cuando se utilizan hilos las capacidades de nuestros proyectos aumentan ya que pueden hacer más tareas a la vez.

# RECOMENDACIONES

* Se recomienda renombrar el paquete principal del proyecto ya que esto nos ayuda a tener una mejor organización de nuestro código.
* Utilizar el modelo MVC ya que de esta manera podemos realizar modificaciones de nuestros programas sin la necesidad de cambiar todo el código.

# BIBLIOGRAFIA

|  |  |
| --- | --- |
| [1] | ORACLE CORPORATION, «JAVA INTRODUCCION,» ORACLE CORPORATION, 2017. [En línea]. Available: https://aws.amazon.com/es/what-is/java/. [Último acceso: 05 11 2022]. |
| [2] | Netbeans, «Welcome to Apache NetBeans,» Netbeans, [En línea]. Available: https://netbeans.apache.org/. [Último acceso: 05 11 2022]. |
| [3] | theastrology, «theastrology,» theastrology, 2018. [En línea]. Available: https://es.theastrologypage.com/concurrency. [Último acceso: 26 11 2022]. |
| [4] | EDTeam, «EDTeam,» EDTeam, 2018. [En línea]. Available: https://ed.team/blog/como-funcionan-los-hilos-en-programacion. [Último acceso: 26 11 2022]. |
| [5] | Universidad de Alicante, «Universidad de Alicante,» Universidad de Alicante, 2012. [En línea]. Available: http://www.jtech.ua.es/dadm/restringido/java/sesion05-apuntes.html#:~:text=En%20Java%20los%20hilos%20est%C3%A1n,definir%20el%20m%C3%A9todo%20run().. [Último acceso: 26 11 2022]. |