

**Tema**

HILOS JAVA (168-169-170)

**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 5](#_Toc120383853)

[1.1. OBJETIVOS 5](#_Toc120383854)

[1.2.1. OBJETIVO GENERAL 5](#_Toc120383855)

[1.2.2. OBJETIVOS ESPECÍFICOS 5](#_Toc120383856)

[1.2. JAVA. 5](#_Toc120383857)

[1.3. APACHE NETBEANS 5](#_Toc120383858)

[1.4. CONCURRENCIA 6](#_Toc120383859)

[1.5. HILOS 6](#_Toc120383860)

[1.6. HILOS PARALELOS Y CONCURRENTES 6](#_Toc120383861)

[1.7. TERMINOLOGÍA Y METODOS DE HILOS EN JAVA. 6](#_Toc120383862)

[2. PARTE PRÁCTICA 8](#_Toc120383863)

[2.1. Video 168 8](#_Toc120383864)

[2.1.1 CREACIÓN DEL PROYECTO. 8](#_Toc120383865)

[2.1.2 Creación de paquetes MVC. 10](#_Toc120383866)

[2.1.3 CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA 11](#_Toc120383867)

[2.1.4. CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO 18](#_Toc120383868)

[2.1.5. CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR 24](#_Toc120383869)

[2.1.6. CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN 27](#_Toc120383870)

[2.1.7. EJECUCIÓN DEL PROYECTO. 29](#_Toc120383871)

[2.2. Video 169 30](#_Toc120383872)

[2.2.1 CREACIÓN DEL PROYECTO. 30](#_Toc120383873)

[2.2.2 Creación de paquetes MVC. 32](#_Toc120383874)

[2.2.3 CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA 33](#_Toc120383875)

[2.2.4 CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO 40](#_Toc120383876)

[2.2.5 CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR 46](#_Toc120383877)

[2.2.6 CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN 49](#_Toc120383878)

[2.2.7 EJECUCIÓN DEL PROYECTO. 51](#_Toc120383879)

[2.3. Video 170 52](#_Toc120383880)

[2.3.1 CREACIÓN DEL PROYECTO. 52](#_Toc120383881)

[2.3.2 Creación de paquetes MVC. 54](#_Toc120383882)

[2.3.3 CODIFICACIÓN DE LOS ARCHIVOS DE LA VISTA 55](#_Toc120383883)

[2.3.4 CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO 62](#_Toc120383884)

[2.3.5 CODIFICACIÓN DE LOS ARCHIVOS DEL CONTROLADOR 68](#_Toc120383885)

[2.3.6 CODIFICACIÓN DE LOS ARCHIVOS DEL PAQUETE MAIN 71](#_Toc120383886)

[2.3.7 EJECUCIÓN DEL PROYECTO. 73](#_Toc120383887)

[3. CONCLUSIONES 74](#_Toc120383888)

[4. RECOMENDACIONES 74](#_Toc120383889)

[5. BIBLIOGRAFIA 74](#_Toc120383890)

**INDICE DE IMÁGENES**

[Figura 1. Logotipo identificativo de Java 4](#_Toc120377586)

[Figura 2. Logotipo identificativo de Netbeans 7](#_Toc120377587)

[Figura 3. Selección del tipo de aplicación a desarrollar 7](#_Toc120377588)

[Figura 4. Nombre del proyecto 8](#_Toc120377589)

[Figura 5. Estructura del proyecto generado. 8](#_Toc120377590)

[Figura 6. Renombre del paquete principal. 9](#_Toc120377591)

[Figura 7. Modificacion del archivo pom.xml. 9](#_Toc120377592)

[Figura 8. Creación de paquete. 10](#_Toc120377593)

[Figura 9. Selección del nombre de los paquetes del video 168 10](#_Toc120377594)

[Figura 10. Creación de clase Java. 11](#_Toc120377595)

[Figura 11. Codificación archivo Vista. 11](#_Toc120377596)

[Figura 13. Creación de clase java. 14](#_Toc120377597)

[Figura 14. Codificación archivo LaminaPelotaVista. 14](#_Toc120377598)

[Figura 15. Creación de clase Java. 17](#_Toc120377599)

[Figura 16. Codificación archivo Pelota. 17](#_Toc120377600)

[Figura 17. Creación de clase Java. 20](#_Toc120377601)

[Figura 18. Codificación archivo PelotaHilos 20](#_Toc120377602)

[Figura 19. Creación de clase Java. 23](#_Toc120377603)

[Figura 20. Codificación archivo Controlador. 23](#_Toc120377604)

[Figura 21. Codificación archivo principal. 26](#_Toc120377605)

[Figura 22. Resultado de la ejecución 28](#_Toc120377606)

[Figura 23. Logotipo identificativo de Netbeans 29](#_Toc120377607)

[Figura 24. Selección del tipo de aplicación a desarrollar 29](#_Toc120377608)

[Figura 25. Nombre del proyecto 30](#_Toc120377609)

[Figura 26. Estructura del proyecto generado. 30](#_Toc120377610)

[Figura 27. Renombre del paquete principal del proyecto 169 31](#_Toc120377611)

[Figura 28. Modificación del paquete pom.xml. 31](#_Toc120377612)

[Figura 29. Creación de clase Java. 32](#_Toc120377613)

[Figura 30. Creación de paquetes proyecto 169. 32](#_Toc120377614)

[Figura 31. Creación de clase Java. 33](#_Toc120377615)

[Figura 32. Codificación archivo Vista. 33](#_Toc120377616)

[Figura 33. Creación de clase Java. 36](#_Toc120377617)

[Figura 34. Codificación archivo Lamina Pelota Vista. 36](#_Toc120377618)

[Figura 35. Creación de clase java. 39](#_Toc120377619)

[Figura 36. Codificación archivo Pelota. 39](#_Toc120377620)

[Figura 37. Creación de clase Java. 42](#_Toc120377621)

[Figura 38. Codificación archivo Pelota Hilos. 42](#_Toc120377622)

[Figura 39. Creación de clase Java. 45](#_Toc120377623)

[Figura 40. Codificación archivo controlador. 45](#_Toc120377624)

[Figura 41. Codificación archivo principal del video 169. 48](#_Toc120377625)

[Figura 42. Resultado del video 169. 50](#_Toc120377626)

[Figura 43. Logotipo identificativo de Apache NetBeans 51](#_Toc120377627)

[Figura 42. Selección del tipo de aplicación a desarrollar 51](#_Toc120377628)

[Figura 43. Nombre del proyecto 52](#_Toc120377629)

[Figura 44. Estructura del proyecto generado. 52](#_Toc120377630)

[Figura 45. Eliminación del archivo index.html del proyecto. 53](#_Toc120377631)

[Figura 46. Modificación del archivo pom.xml video 170. 53](#_Toc120377632)

[Figura 47. Creación de clase Java. 54](#_Toc120377633)

[Figura 48. Creación de paquete 170. 54](#_Toc120377634)

[Figura 50. Creación de clase Java. 55](#_Toc120377635)

[Figura 51. Codificación archivo Vista. 55](#_Toc120377636)

[Figura 52. Creación de clase Java. 58](#_Toc120377637)

[Figura 53. Codificación archivo Lamina Pelota Vista. 58](#_Toc120377638)

[Figura 58. Creación de clase Java. 61](#_Toc120377639)

[Figura 59. Codificación archivo Pelota. 61](#_Toc120377640)

[Figura 60. Creación de clase Java. 64](#_Toc120377641)

[Figura 61. Codificación archivo Pelota Hilos. 64](#_Toc120377642)

[Figura 62. Creación de clase Java. 67](#_Toc120377643)

[Figura 63. Codificación archivo controlador. 67](#_Toc120377644)

[Figura 64. Codificación archivo principal del proyecto 170. 70](#_Toc120377645)

[Figura 65. Resultado del proyecto 170. 72](#_Toc120377646)

**ÍNDICE DE TABLAS**

[Tabla 1. Terminología utilizada en Hilos Java 6](#_Toc120377776)

[Tabla 2. Código de Vista. 13](#_Toc120377777)

[Tabla 3. Codificación del archivo Lamina Pelota Vista. 16](#_Toc120377778)

[Tabla 4. Codificación del archivo Pelota 19](#_Toc120377779)

[Tabla 6. Codificación del archivo Controlador 25](#_Toc120377780)

[Tabla 7. Codificación del archivo principal del proyecto 28](#_Toc120377781)

[Tabla 8. Codificación del archivo Vista. 35](#_Toc120377782)

[Tabla 9. Codificación del archivo Lamina Pelota Vista 38](#_Toc120377783)

[Tabla 10. Codificación del archivo Pelota 41](#_Toc120377784)

[Tabla 12. Codificación del archivo Controlador 47](#_Toc120377785)

[Tabla 13. Codificación del archivo principal del proyecto. 50](#_Toc120377786)

[Tabla 14. Codificación del archivo Vista 57](#_Toc120377787)

[Tabla 15. Codificación del archivo Lamina Pelota Hilos. 60](#_Toc120377788)

[Tabla 16. Codificación del archivo Pelota 63](#_Toc120377789)

[Tabla 17. Codificación del archivo PelotaHilos 66](#_Toc120377790)

[Tabla 18. Codificación del archivo Controlador 69](#_Toc120377791)

[Tabla 19. Codificación del archivo Principal. 72](#_Toc120377792)

# MARCO TEÓRICO

## OBJETIVOS

### OBJETIVO GENERAL

## El objetivo general de este estudio es comprender la funcionalidad de los hilos en Java y su aplicación en la ejecución de múltiples tareas de manera concurrente. Para lograrlo, nos proponemos alcanzar los siguientes objetivos específicos:.

### 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.

## INTRODUCCION.

## El desarrollo de software se ha convertido en una parte esencial de nuestra sociedad, y el lenguaje de programación Java ha sido uno de los más utilizados para crear una amplia variedad de aplicaciones. Uno de los aspectos clave de Java es su capacidad para ejecutar múltiples tareas simultáneamente mediante el uso de hilos.

## En esta introducción, exploraremos el tema de los hilos en Java y su funcionalidad para ejecutar tareas de manera concurrente. Comprenderemos cómo los hilos pueden mejorar la eficiencia y la capacidad de respuesta de un programa al permitir la ejecución simultánea de múltiples procesos. También analizaremos la forma en que un programa diseñado con hilos actúa de manera diferente a uno que no los utiliza.

## Java es un lenguaje de programación ampliamente utilizado en diversas aplicaciones, desde programas de escritorio hasta desarrollo web, videojuegos, inteligencia artificial e internet de las cosas (IoT). Su capacidad para aplicar el paradigma orientado a objetos es una de sus fortalezas más destacadas. Java cuenta con varias versiones disponibles, siendo la versión 8 la más reciente en el momento de redacción de este texto, y se divide en tres ediciones principales: Java SE, Java EE y Java ME.

## En el ámbito del desarrollo de software, contar con un entorno de desarrollo integrado (IDE) eficiente y poderoso es fundamental. En este contexto, Apache NetBeans es una opción popular. Se trata de un IDE creado principalmente para Java, que ofrece una amplia gama de plugins y herramientas para facilitar el desarrollo y la ejecución de aplicaciones. NetBeans es gratuito y compatible con múltiples sistemas operativos, lo que lo convierte en una opción accesible para desarrolladores tanto en entornos empresariales como educativos.

## La concurrencia es un concepto clave cuando se trabaja con hilos en Java. Se refiere a la capacidad de ejecutar múltiples procesos simultáneamente. Estos procesos pueden ser conjuntos de tareas o hilos de ejecución creados por un programa. La concurrencia puede tener lugar en una única CPU, en varios procesadores o incluso en una red de computadoras distribuidas.

## Un hilo en Java es un flujo de control dentro de un programa. Al crear múltiples hilos, es posible realizar varias tareas de forma simultánea. Cada hilo tiene su propio contexto de ejecución, como un contador de programa y una pila de ejecución, pero comparten el mismo espacio de memoria. Esto implica que se debe prestar atención a la sincronización de los hilos cuando varios de ellos acceden a los mismos objetos. Los hilos son útiles para dividir el flujo de un programa en partes independientes, lo que permite que cada hilo se ocupe de una tarea específica. Por ejemplo, un hilo puede encargarse de la comunicación con el usuario mientras otros hilos realizan tareas en segundo plano o acceden a recursos del sistema.

## Es importante distinguir entre los hilos paralelos y los hilos concurrentes en Java. Los hilos paralelos permiten que las tareas se ejecuten verdaderamente de forma simultánea, lo que puede mejorar significativamente la eficiencia del programa. Por otro lado, los hilos concurrentes permiten que los hilos avancen independientemente uno del otro, a diferentes velocidades. Esto puede resultar en ejecuciones diferentes cada vez que se corre el programa.

## En la parte práctica de este estudio, nos centraremos en el video 168, donde se abordarán diferentes aspectos relacionados con la creación y utilización de hilos en Java. El video presenta un ejemplo concreto en el que se desarrolla un proyecto utilizando el IDE Apache NetBeans. A lo largo de la práctica, se crean paquetes que representan los elementos de la arquitectura MVC (Modelo-Vista-Controlador) y se codifican clases en cada uno de ellos.

## JAVA.

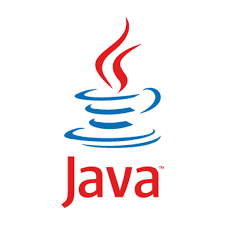


Figura 1. Logotipo identificativo de Java

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 168

### 2.1.1 CREACIÓN DEL PROYECTO.

Abra el IDE de desarrollo Apache Netbeans 14.



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 Mave -> Web Application

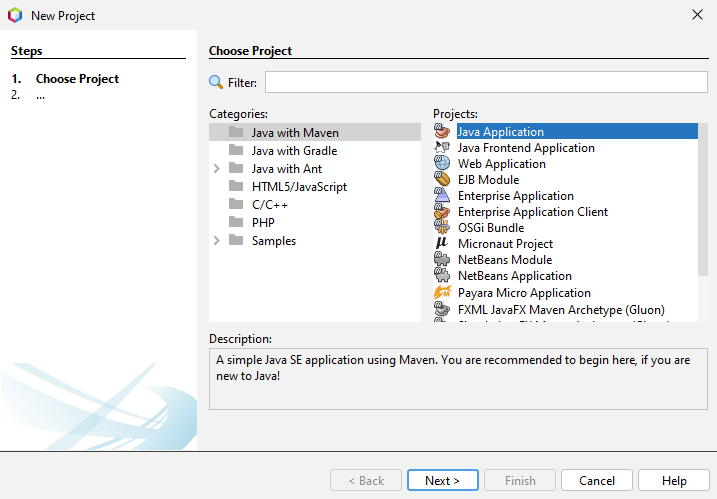


Figura 3. Selección del tipo de aplicación a desarrollar

Le da el siguiente nombre HilosJava\_Video168\_grupo#, 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.

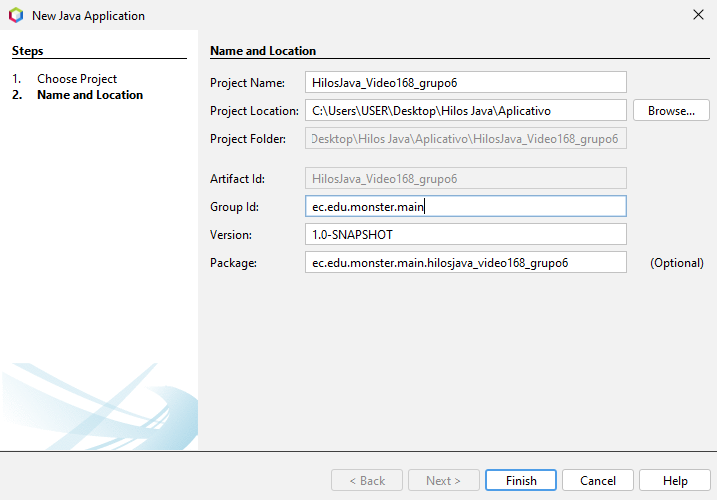


Figura 4. Nombre del proyecto

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

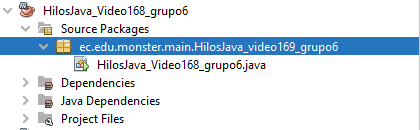


Figura 5. Estructura del proyecto generado.

Si desea para mas orden renombre el paquete creado inicialmente (ec.edu.monster.main.HilosJava\_video169\_grupo#) por “ec.edu.monster.main” Seleccione el paquete, luego->refactor->Rename.

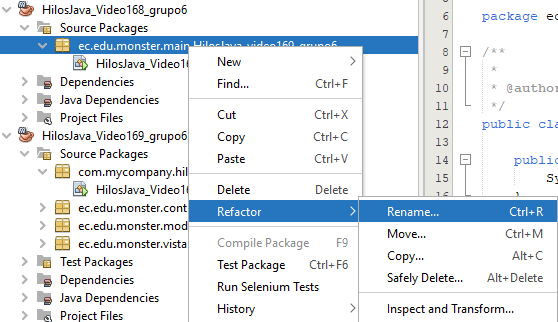


Figura 6. Renombre del paquete principal.

Abra la carpeta Project Files y selecciones el archivo pom.xml y en la etiqueta <group Id> coloque “ec.edu.monster.main”, mientras que en la etiqueta <exec.mainClass> “ec.edu.monster.main.HilosJava\_Video168\_grupo#”.

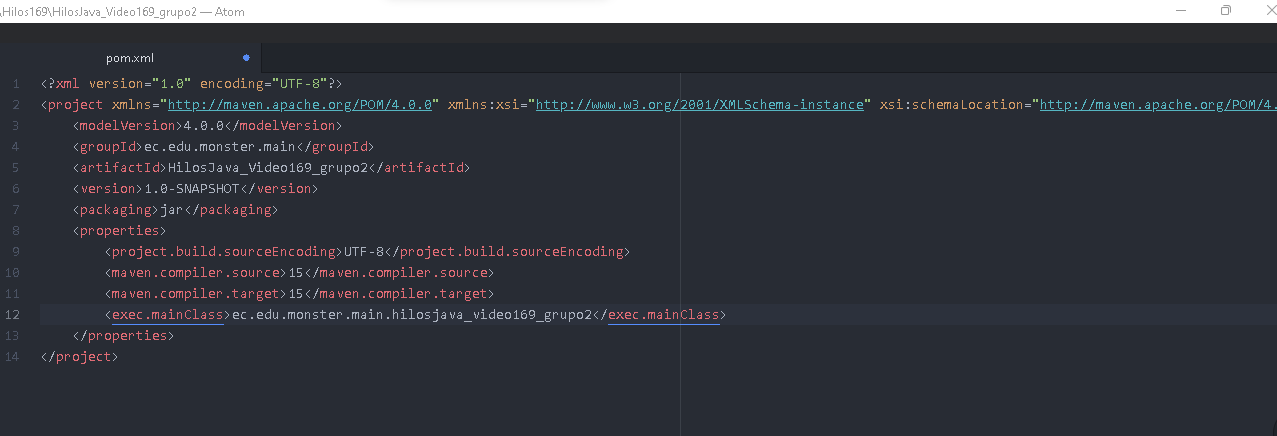


Figura 7. Modificacion del archivo pom.xml.

### 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

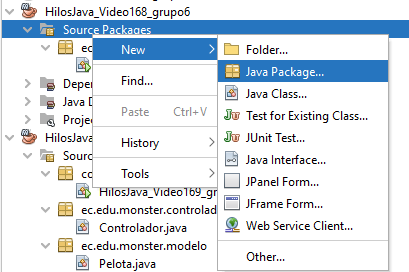


Figura 8. Creación de paquete.

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

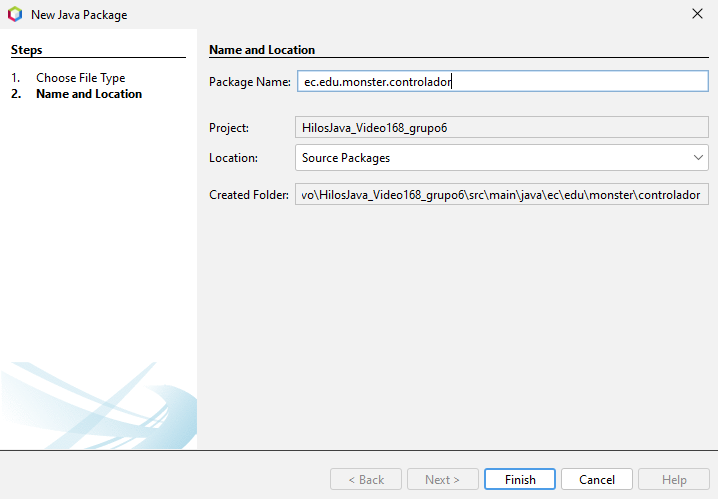


Figura 9. Selección del nombre de los paquetes del video 168

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.

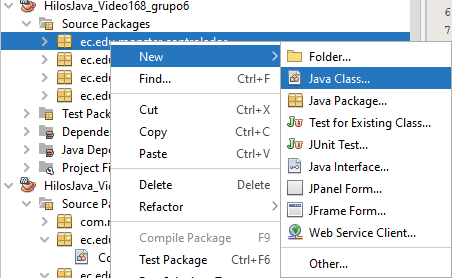


Figura 10. Creación de clase Java.

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

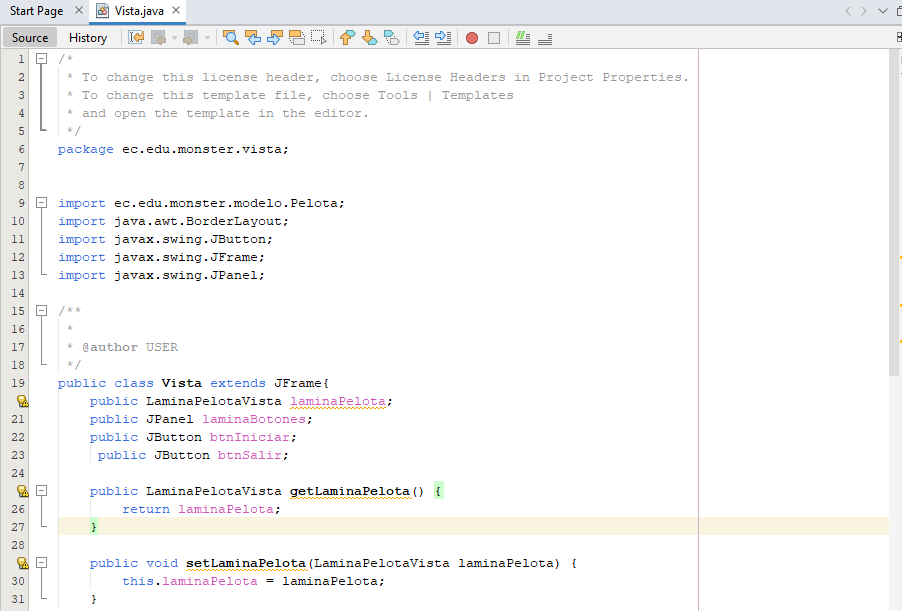


Figura 11. Codificación archivo Vista.

Tabla 2. Código de Vista.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.vista;

import ec.edu.monster.modelo.Pelota;

import java.awt.BorderLayout;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

/\*\*

\*

\* @author USER

\*/

public class Vista extends JFrame{

public LaminaPelotaVista laminaPelota;

public JPanel laminaBotones;

public JButton btnIniciar;

public JButton btnSalir;

public LaminaPelotaVista getLaminaPelota() {

return laminaPelota;

}

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Pelotas en movimiento");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

} \*/

public class Vista extends JFrame{

public LaminaPelotaVista laminaPelota;

public JPanel laminaBotones;

public JButton btnIniciar;

public JButton btnSalir;

public LaminaPelotaVista getLaminaPelota() {

return laminaPelota;

}

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Pelotas en movimiento");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

}

</tr>

<tr>

<td>PHP</td>

<td><input type="checkbox" name="chkLeng" value="PHP" /></td>

</tr>

<tr>

<td>Ruby</td>

<td><input type="checkbox" name="chkLeng" value="Ruby" /></td>

</tr>

<tr>

<td>Python</td>

<td><input type="checkbox" name="chkLeng" value="Python" /></td>

</tr>

<tr>

<td></td>

<td><input type="submit" value="Enviar" /></td>

</tr>

</tbody>

</table>

</form>

</body>

</html>

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Pelotas en movimiento");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

}

Cree otra clase Java.

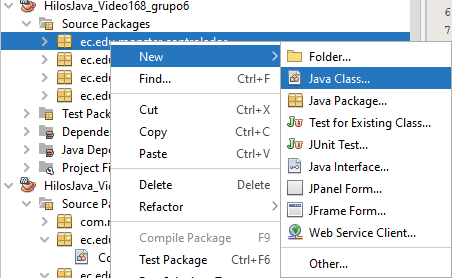


Figura 13. Creación de clase java.

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

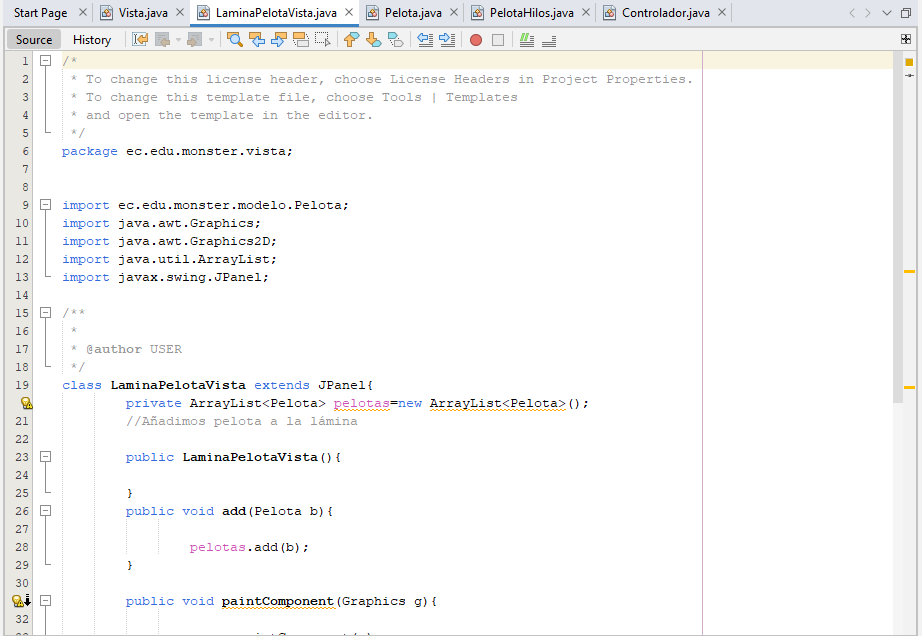


Figura 14. Codificación archivo LaminaPelotaVista.

Tabla . Codificación del archivo Lamina Pelota Vista.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.vista;

import ec.edu.monster.modelo.Pelota;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.util.ArrayList;

import javax.swing.JPanel;

/\*\*

\*

\* @author USER

\*/

class LaminaPelotaVista extends JPanel{

private ArrayList<Pelota> pelotas=new ArrayList<Pelota>();

//Añadimos pelota a la lámina

public LaminaPelotaVista(){

}

public void add(Pelota b){

pelotas.add(b);

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2=(Graphics2D)g;

for(Pelota b: pelotas){

g2.fill(b.getShape());

}

}

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2=(Graphics2D)g;

for(Pelota b: pelotas){

g2.fill(b.getShape());

}

}

}

### CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO

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

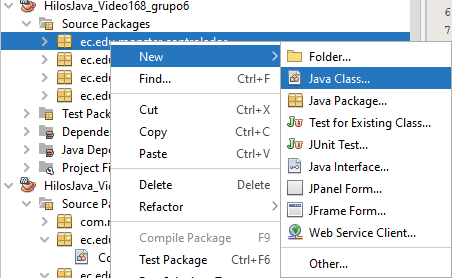


Figura 15. Creación de clase Java.

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

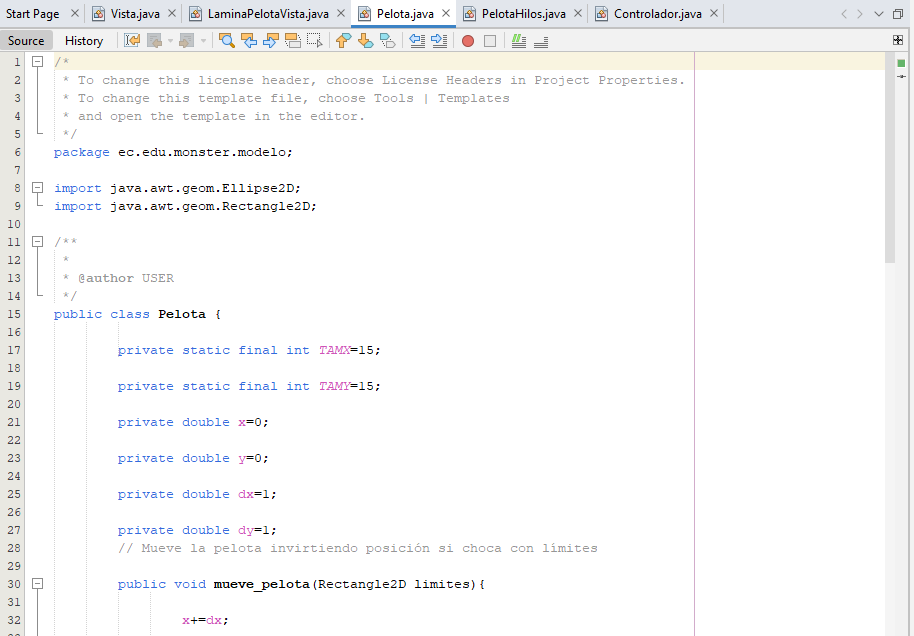


Figura 16. Codificación archivo Pelota.

Tabla . Codificación del archivo Pelota

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.modelo;

import java.awt.geom.Ellipse2D;

import java.awt.geom.Rectangle2D;

/\*\*

\*

\* @author USER

\*/

public class Pelota {

private static final int TAMX=15;

private static final int TAMY=15;

private double x=0;

private double y=0;

private double dx=1;

private double dy=1;

// Mueve la pelota invirtiendo posición si choca con límites

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

}

Cree una nueva clase en el paquete del modelo.

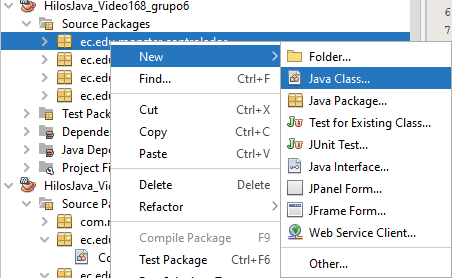


Figura 17. Creación de clase Java.

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

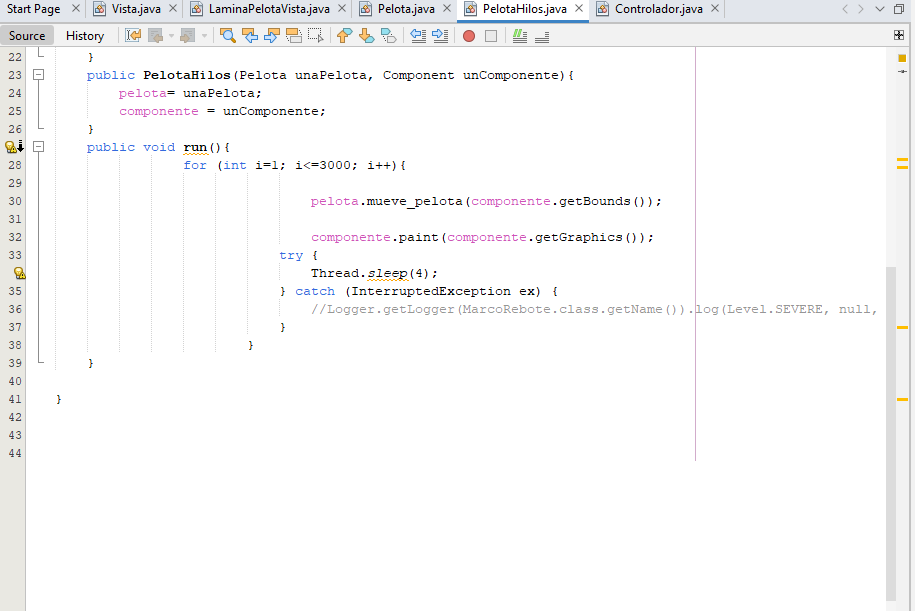


Figura 18. Codificación archivo PelotaHilos

*Tabla 5.* Codificación del archivo PelotaHilos Vista.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.modelo;

import java.awt.Component;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author USER

\*/

public class PelotaHilos implements Runnable {

private Pelota pelota;

private Component componente;

public PelotaHilos(){

}

public PelotaHilos(Pelota unaPelota, Component unComponente){

pelota= unaPelota;

componente = unComponente;

}

public void run(){

for (int i=1; i<=3000; i++){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

try {

Thread.sleep(4);

} catch (InterruptedException ex) {

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

}

}

}

}

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

*aaaa*

public void run(){

for (int i=1; i<=3000; i++){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

try {

Thread.sleep(4);

} catch (InterruptedException ex) {

//Logger.getLogger(MarcoRebote.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.

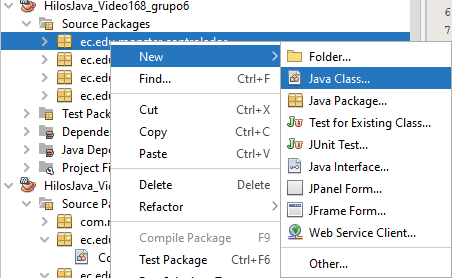


Figura 19. Creación de clase Java.

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

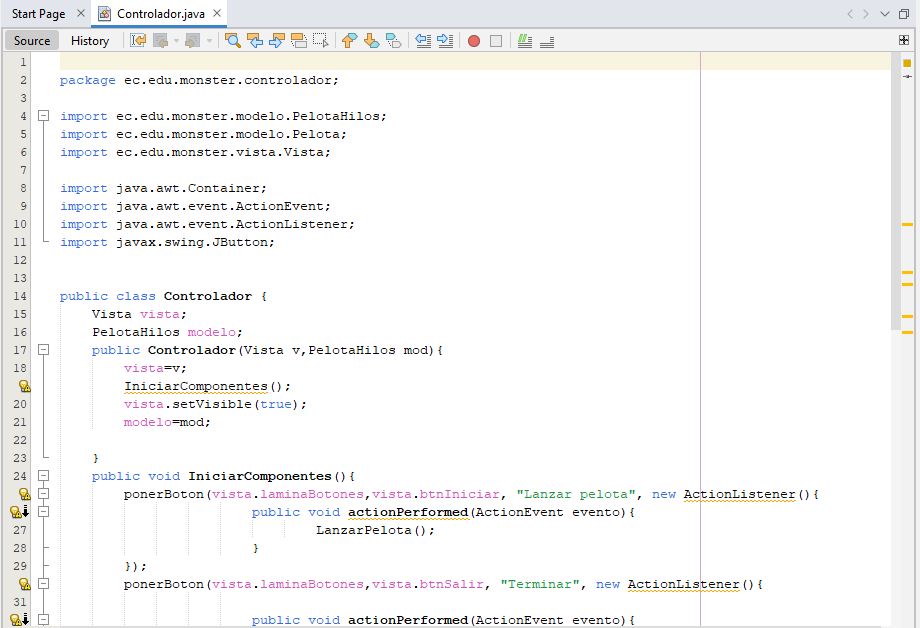


Figura 20. Codificación archivo Controlador.

Tabla 6. Codificación del archivo Controlador

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.PelotaHilos;

import ec.edu.monster.modelo.Pelota;

import ec.edu.monster.vista.Vista;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

public class Controlador {

Vista vista;

PelotaHilos modelo;

public Controlador(Vista v,PelotaHilos mod){

vista=v;

IniciarComponentes();

vista.setVisible(true);

modelo=mod;

}

public void IniciarComponentes(){

ponerBoton(vista.laminaBotones,vista.btnIniciar, "Lanzar pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

LanzarPelota();

}

});

ponerBoton(vista.laminaBotones,vista.btnSalir, "Terminar", new ActionListener(){

public void actionPerformed(ActionEvent evento){

System.exit(0);

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener accion){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(accion);

}

public void LanzarPelota(){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

Thread t = new Thread(r);

t.start();

}

}

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

ponerBoton(vista.laminaBotones,vista.btnSalir, "Terminar", new ActionListener(){

public void actionPerformed(ActionEvent evento){

System.exit(0);

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener accion){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(accion);

}

public void LanzarPelota(){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

Thread t = new Thread(r);

t.start();

}

}

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

### 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.

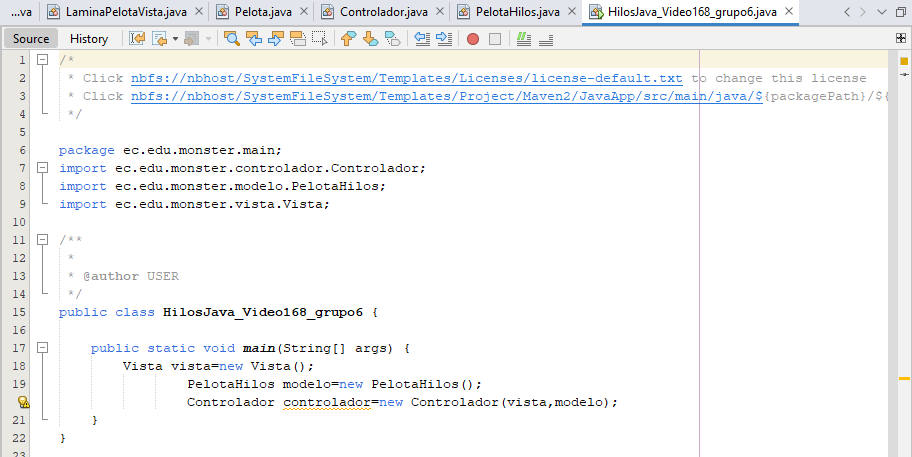


Figura 21. Codificación archivo principal.

Tabla 7. 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\_grupo6 {

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.



Figura 22. Resultado de la ejecución

## Video 169

### 2.2.1 CREACIÓN DEL PROYECTO.

Abra el IDE de desarrollo Apache Netbeans 14.



Figura 23. Logotipo identificativo de Netbeans

Abra Apache Netbeans y se sitúa en la Pestaña superior y selecciona File->New Project -> Java with Mave -> Web Application

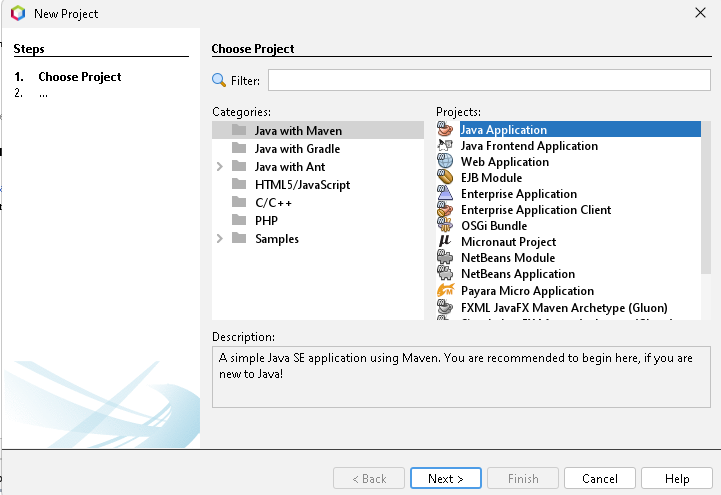


Figura 24. Selección del tipo de aplicación a desarrollar

Le da el siguiente nombre HilosJava\_Video169\_grupo#, 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.

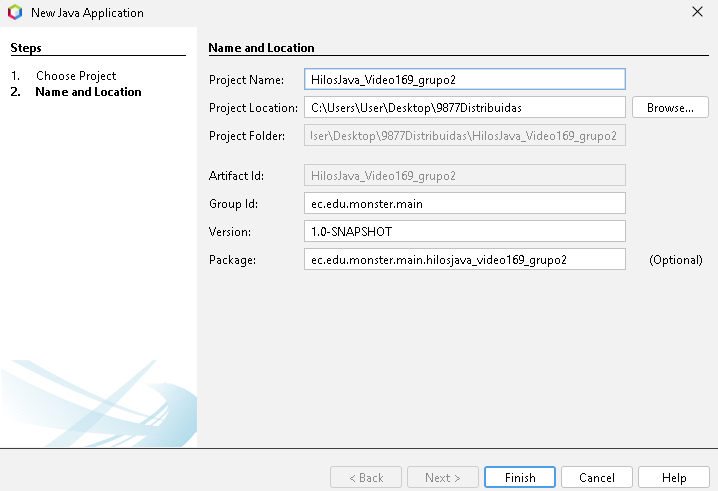
****

Figura 25. Nombre del proyecto

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

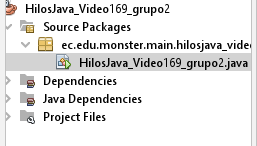
****

Figura 26. Estructura del proyecto generado.

Si desea para mas orden renombre el paquete creado inicialmente (ec.edu.monster.main.HilosJava\_video169\_grupo#) por “ec.edu.monster.main” Seleccione el paquete, luego->refactor->Rename.

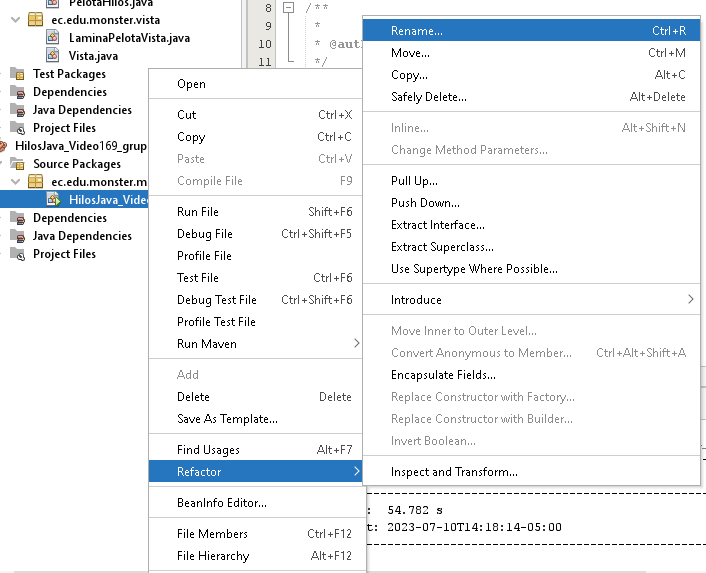


Figura 27. Renombre del paquete principal del proyecto 169

Abra la carpeta Project Files y selecciones el archivo pom.xml y en la etiqueta <group Id> coloque “ec.edu.monster.main”, mientra que en la etiqueta <exec.mainClass> “ec.edu.monster.main.HilosJava\_Video169\_grupo#”.

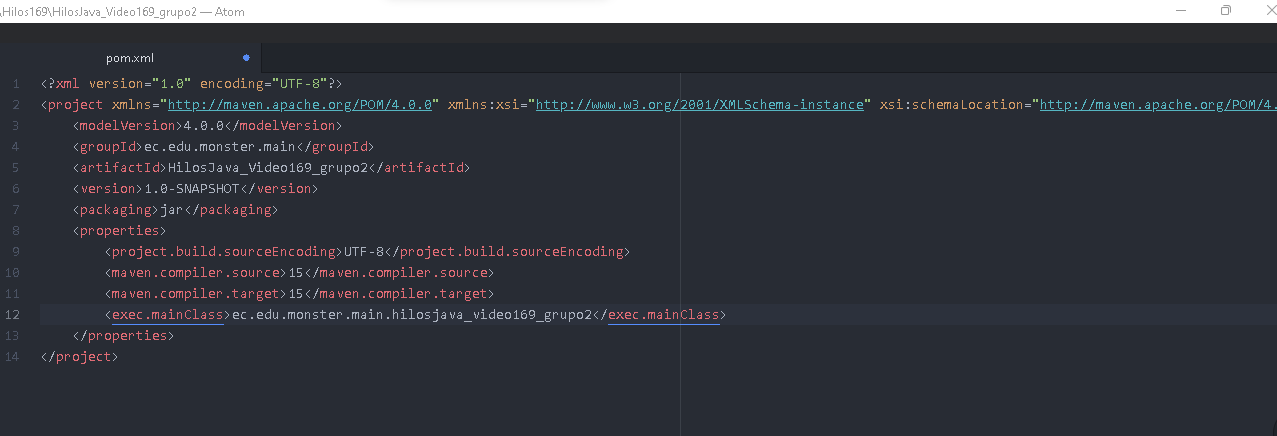


Figura 28. Modificación del paquete pom.xml.

### 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

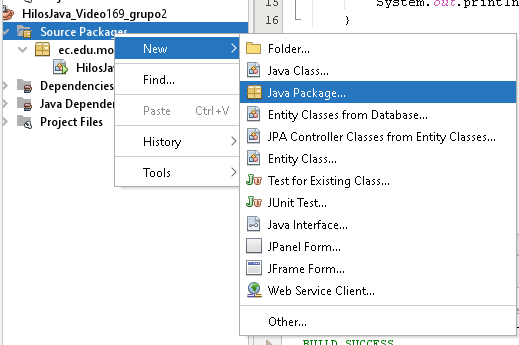


Figura 29. Creación de clase Java.

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

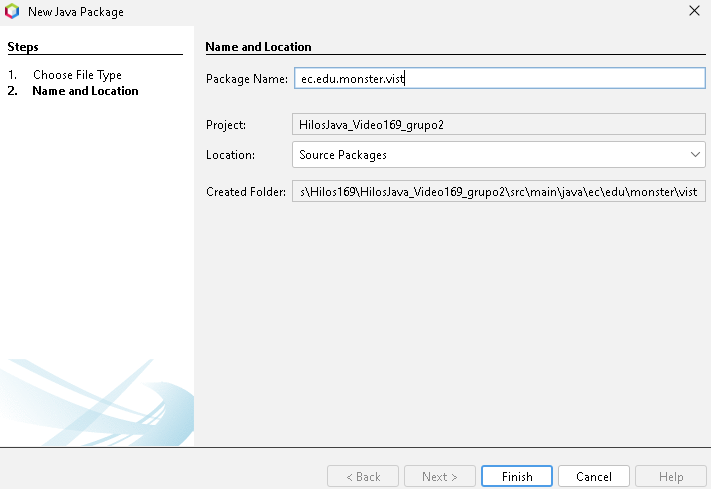


Figura 30. Creación de paquetes proyecto 169.

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.

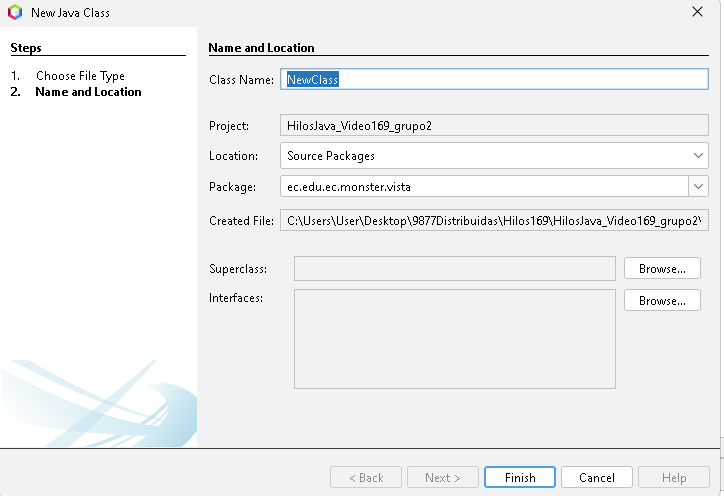


Figura 31. Creación de clase Java.

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

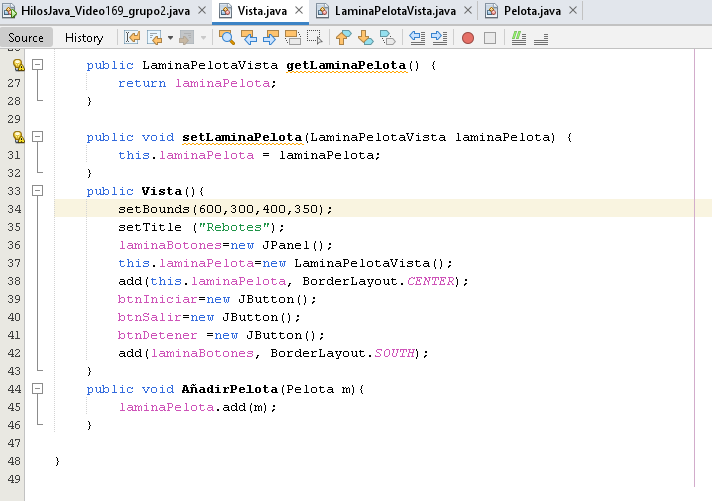


Figura 32. Codificación archivo Vista.

Tabla 8. Codificación del archivo Vista.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.vista;

import ec.edu.monster.modelo.Pelota;

import java.awt.BorderLayout;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

/\*\*

\*

\* @author USER

\*/

public class Vista extends JFrame{

public LaminaPelotaVista laminaPelota;

public JPanel laminaBotones;

public JButton btnIniciar;

public JButton btnSalir;

public JButton btnDetener;

public LaminaPelotaVista getLaminaPelota() {

return laminaPelota;

}

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Rebotes");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

btnDetener =new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

}

public LaminaPelotaVista getLaminaPelota() {

return laminaPelota;

}

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Rebotes");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

btnDetener =new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

}

Cree otra clase Java.

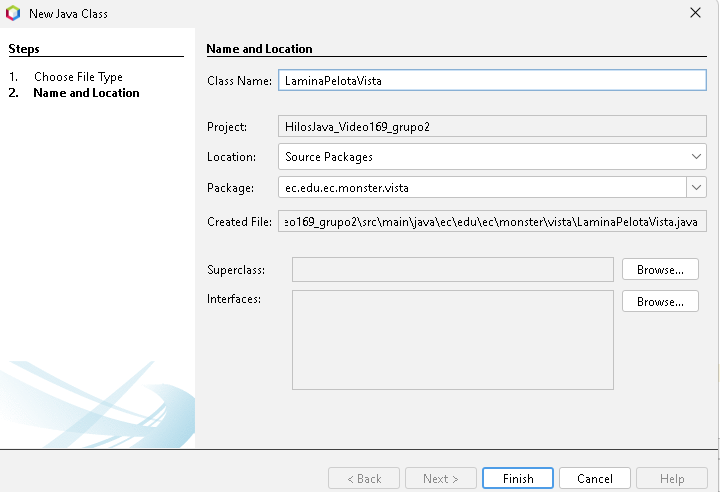


Figura 33. Creación de clase Java.

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

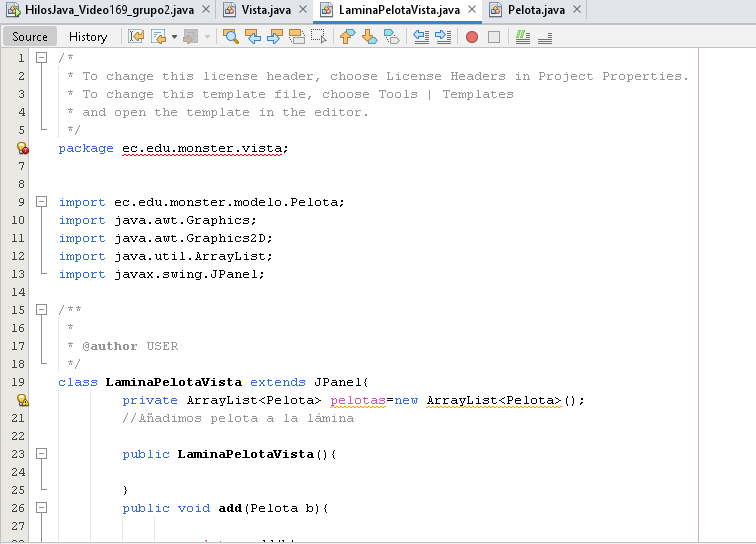


Figura 34. Codificación archivo Lamina Pelota Vista.

Tabla 9. Codificación del archivo Lamina Pelota Vista

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.vista;

import ec.edu.monster.modelo.Pelota;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.util.ArrayList;

import javax.swing.JPanel;

/\*\*

\*

\* @author USER

\*/

class LaminaPelotaVista extends JPanel{

private ArrayList<Pelota> pelotas=new ArrayList<Pelota>();

//Añadimos pelota a la lámina

public LaminaPelotaVista(){

}

public void add(Pelota b){

pelotas.add(b);

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2=(Graphics2D)g;

for(Pelota b: pelotas){

g2.fill(b.getShape());

}

}

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2=(Graphics2D)g;

for(Pelota b: pelotas){

g2.fill(b.getShape());

}

}

}

}}

### CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO

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

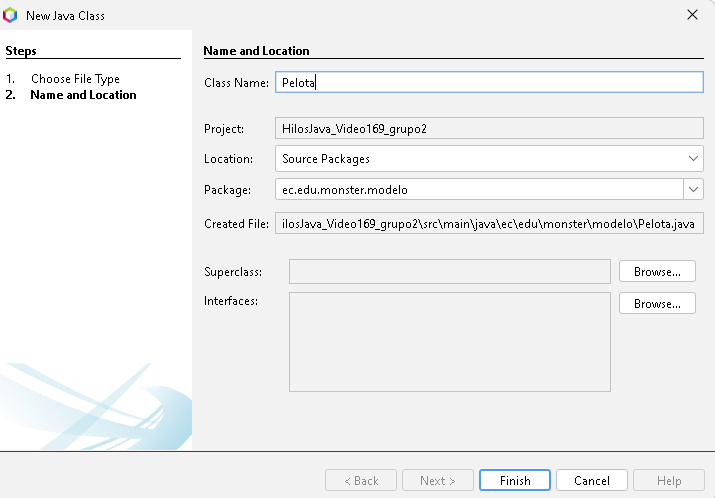


Figura 35. Creación de clase java.

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

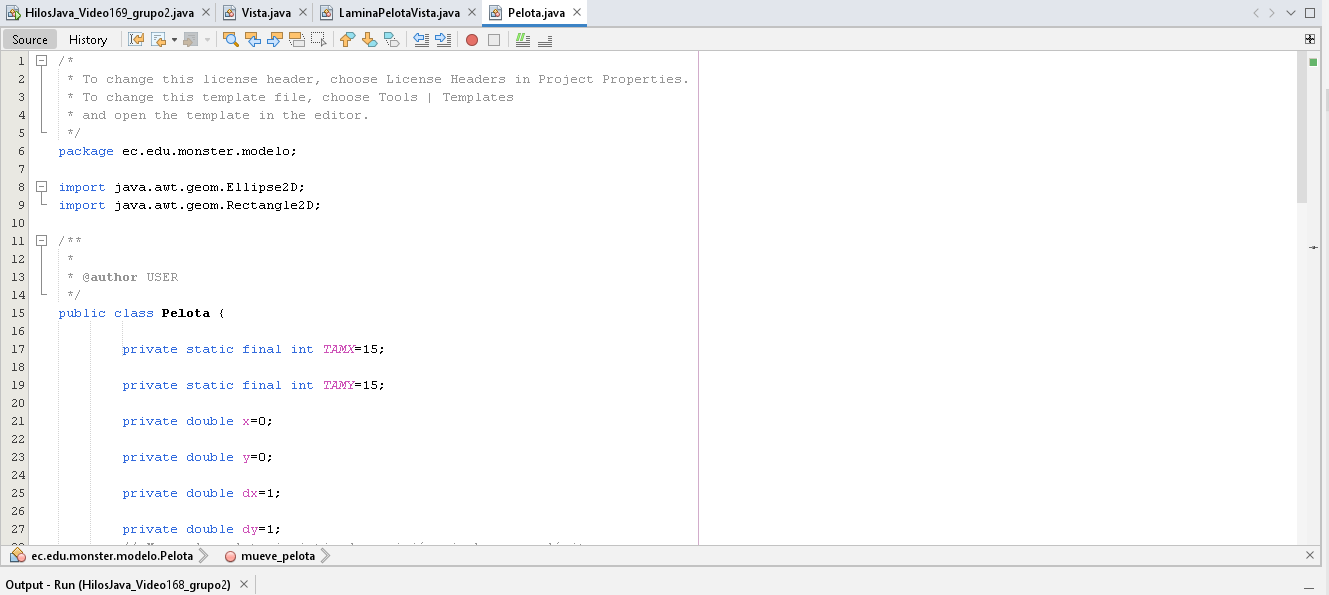


Figura 36. Codificación archivo Pelota.

Tabla 10. Codificación del archivo Pelota

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.modelo;

import java.awt.geom.Ellipse2D;

import java.awt.geom.Rectangle2D;

/\*\*

\*

\* @author USER

\*/

public class Pelota {

private static final int TAMX=15;

private static final int TAMY=15;

private double x=0;

private double y=0;

private double dx=1;

private double dy=1;

// Mueve la pelota invirtiendo posición si choca con límites

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

}

Cree una nueva clase en el paquete del modelo.



Figura 37. Creación de clase Java.

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

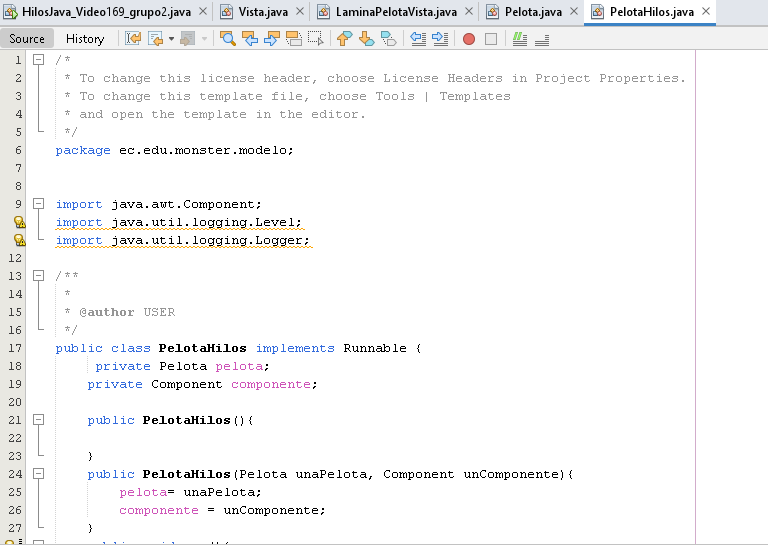


Figura 38. Codificación archivo Pelota Hilos.

*Tabla 11.* Codificación del archivo Pelota Hilos

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.modelo;

import java.awt.Component;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author USER

\*/

public class PelotaHilos implements Runnable {

private Pelota pelota;

private Component componente;

public PelotaHilos(){

}

public PelotaHilos(Pelota unaPelota, Component unComponente){

pelota= unaPelota;

componente = unComponente;

}

public void run(){

while (!Thread.currentThread().isInterrupted()){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*for (int i=1; i<=3000; i++){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*try {

Thread.sleep(4);

} catch (InterruptedException ex) {

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

}

}\*/

}

}

}

*aaaa*

while (!Thread.currentThread().isInterrupted()){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*for (int i=1; i<=3000; i++){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*try {

Thread.sleep(4);

} catch (InterruptedException ex) {

//Logger.getLogger(MarcoRebote.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.

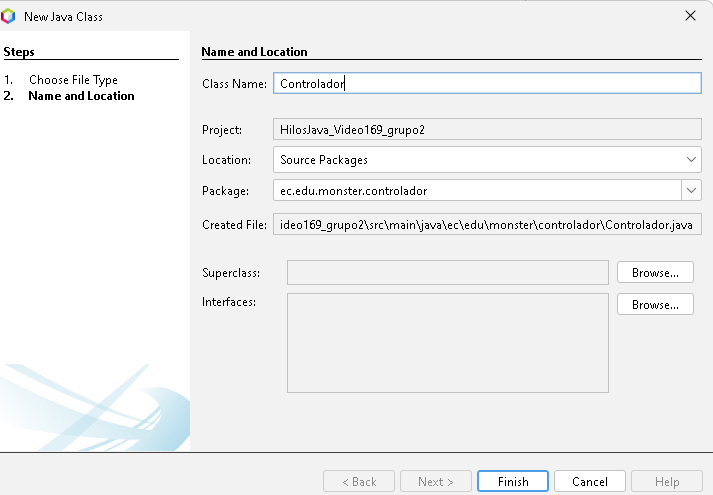


Figura 39. Creación de clase Java.

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

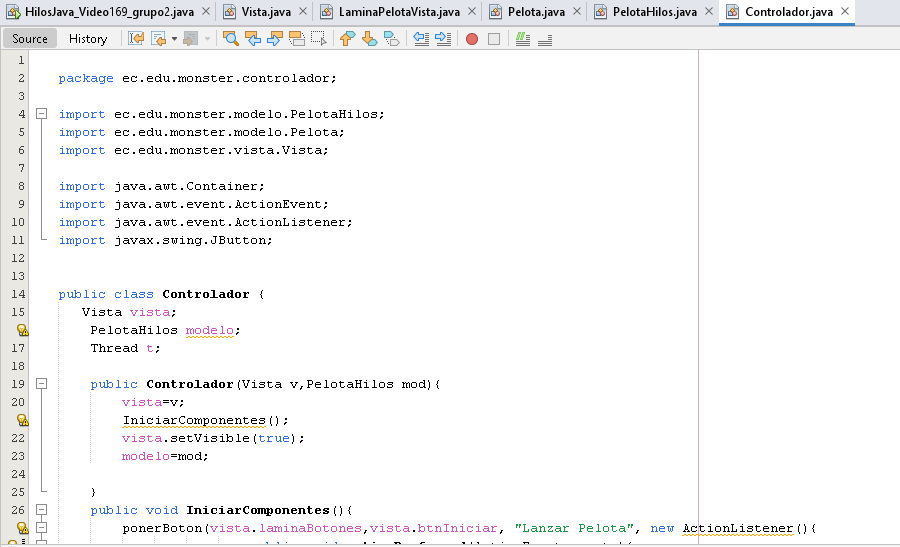


Figura 40. Codificación archivo controlador.

Tabla 12. Codificación del archivo Controlador

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.PelotaHilos;

import ec.edu.monster.modelo.Pelota;

import ec.edu.monster.vista.Vista;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

public class Controlador {

Vista vista;

PelotaHilos modelo;

Thread t;

public Controlador(Vista v,PelotaHilos mod){

vista=v;

IniciarComponentes();

vista.setVisible(true);

modelo=mod;

}

public void IniciarComponentes(){

ponerBoton(vista.laminaBotones,vista.btnIniciar, "Lanzar Pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

LanzarPelota();

}

});

ponerBoton(vista.laminaBotones,vista.btnSalir, "Salir", new ActionListener(){

public void actionPerformed(ActionEvent evento){

System.exit(0);

}

});

ponerBoton(vista.laminaBotones,vista.btnDetener, "Parar pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

detener();

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener oyente){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(oyente);

}

public void LanzarPelota (){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

t = new Thread(r);

t.start();

}

public void detener(){

//t.stop();

t.interrupt();

}

}

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener accion){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(accion);

}

public void LanzarPelota(){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

Thread t = new Thread(r);

t.start();

}

}

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

LanzarPelota();

}

});

ponerBoton(vista.laminaBotones,vista.btnSalir, "Salir", new ActionListener(){

public void actionPerformed(ActionEvent evento){

System.exit(0);

}

});

ponerBoton(vista.laminaBotones,vista.btnDetener, "Parar pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

detener();

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener oyente){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(oyente);

}

public void LanzarPelota (){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

t = new Thread(r);

t.start();

}

public void detener(){

t.interrupt(); }

}

### 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.

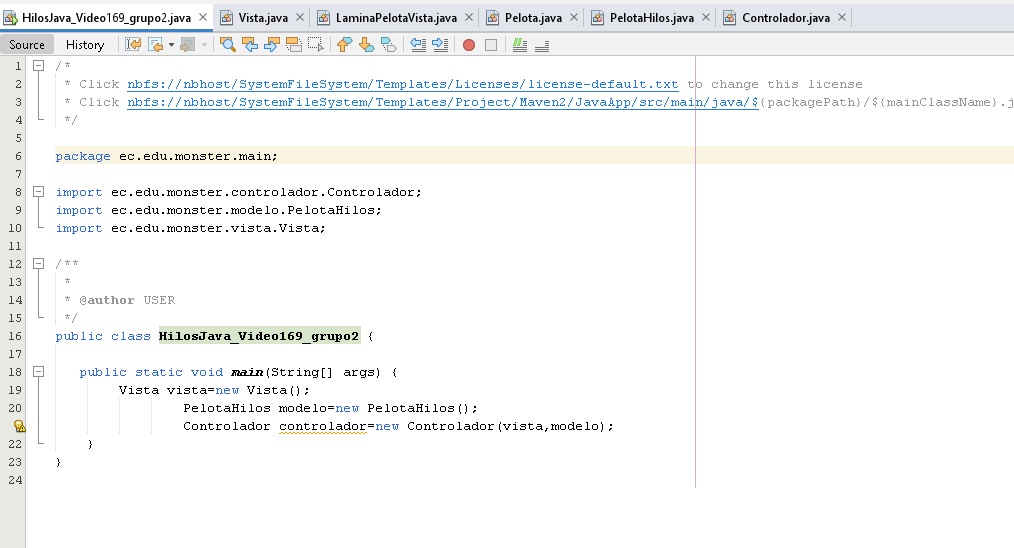


Figura 41. Codificación archivo principal del video 169.

Tabla 13. 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\_Video169\_grupo6 {

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 una pelota, además de tener la capacidad de detenerla según se desee, esto se logra con las modificaciones realizadas en comparación al proyecto anterior.

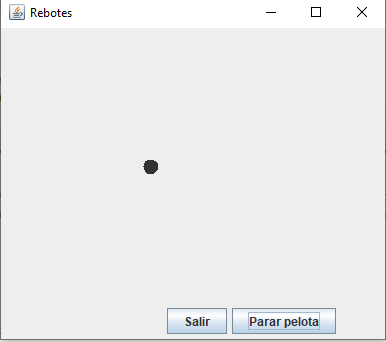


Figura 42. Resultado del video 169.

## Video 170

### 2.3.1 CREACIÓN DEL PROYECTO.

Abra el IDE de desarrollo Apache Netbeans 14.



Figura 43. Logotipo identificativo de Apache NetBeans

Abra Apache Netbeans y se sitúa en la Pestaña superior y selecciona File->New Project -> Java with Mave -> Web Application

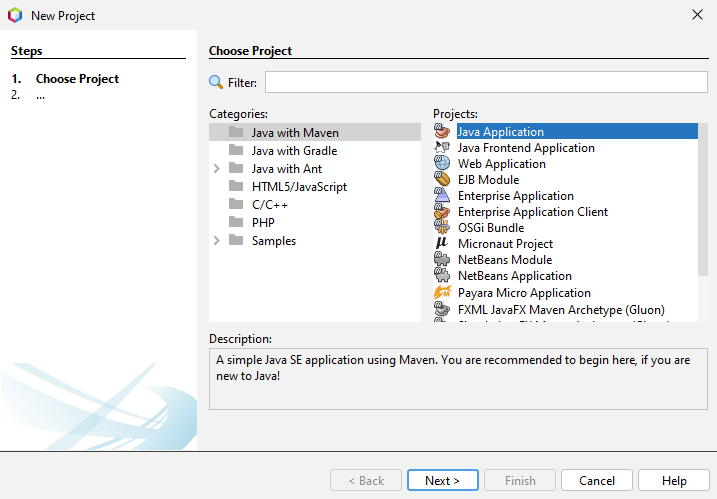


Figura 42. Selección del tipo de aplicación a desarrollar

Le da el siguiente nombre HilosJava\_Video170\_grupo#, 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.

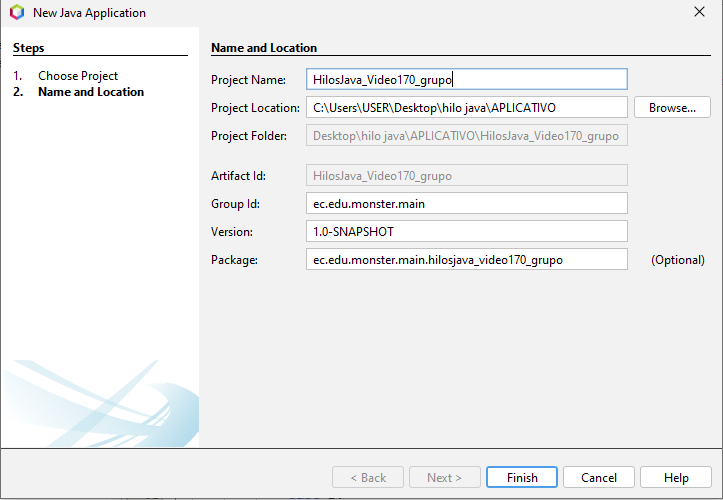


Figura 43. Nombre del proyecto

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

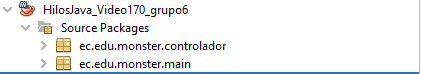


Figura 44. Estructura del proyecto generado.

Si desea para mas orden renombre el paquete creado inicialmente (ec.edu.monster.main.HilosJava\_video169\_grupo#) por “ec.edu.monster.main” Seleccione el paquete, luego->refactor->Rename.

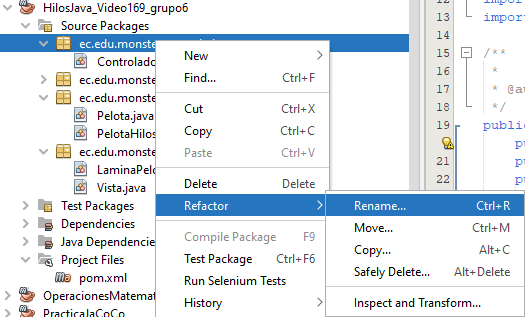


Figura 45. Eliminación del archivo index.html del proyecto.

Abra la carpeta Project Files y selecciones el archivo pom.xml y en la etiqueta <group Id> coloque “ec.edu.monster.main”, mientra que en la etiqueta <exec.mainClass> “ec.edu.monster.main.HilosJava\_Video168\_grupo#”.

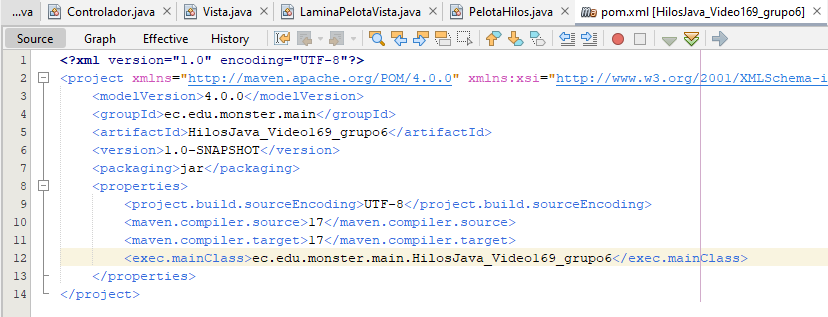


Figura 46. Modificación del archivo pom.xml video 170.

### 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 47. Creación de clase Java.

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

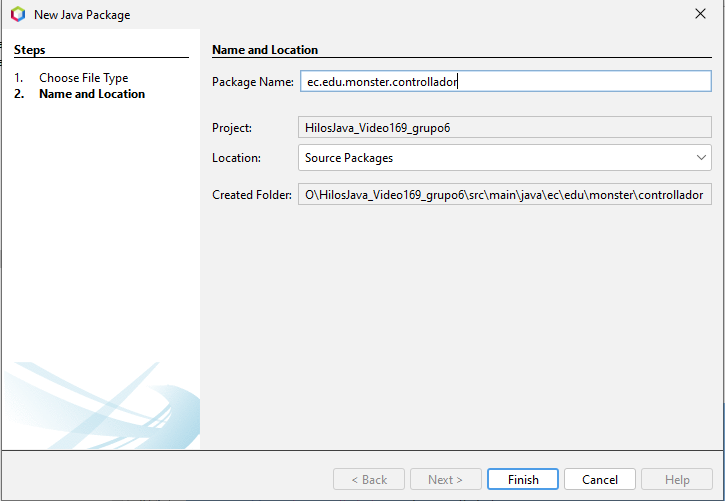


Figura 48. Creación de paquete 170.

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 50. Creación de clase Java.

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

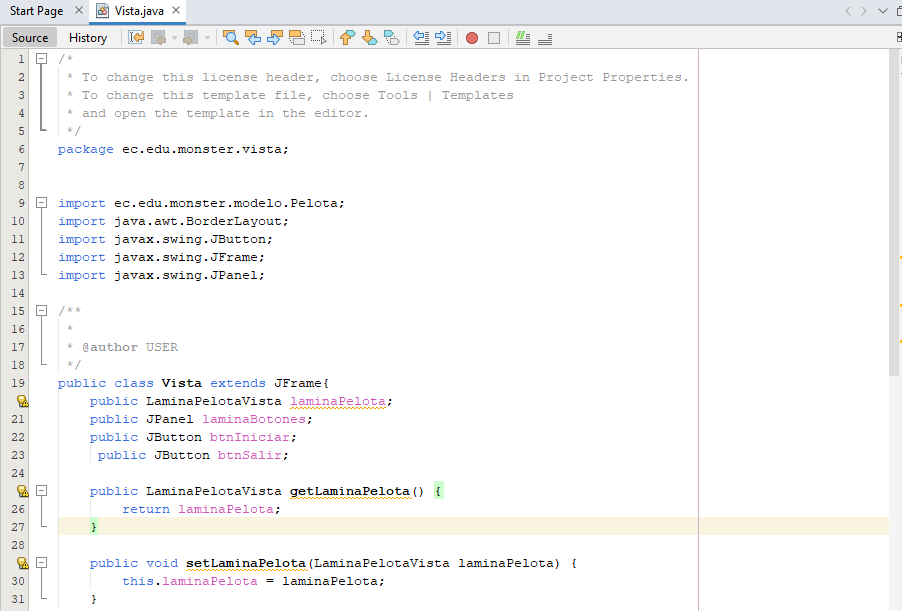


Figura 51. Codificación archivo Vista.

Tabla 14. Codificación del archivo Vista

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.vista;

import ec.edu.monster.modelo.Pelota;

import java.awt.BorderLayout;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

/\*\*

\*

\* @author USER

\*/

public class Vista extends JFrame{

public LaminaPelotaVista laminaPelota;

public JPanel laminaBotones;

public JButton btnIniciar;

public JButton btnSalir;

public JButton btnDetener;

public LaminaPelotaVista getLaminaPelota() {

return laminaPelota;

}

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Rebotes");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

btnDetener =new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

}

public LaminaPelotaVista getLaminaPelota() {

return laminaPelota;

}

public void setLaminaPelota(LaminaPelotaVista laminaPelota) {

this.laminaPelota = laminaPelota;

}

public Vista(){

setBounds(600,300,400,350);

setTitle ("Rebotes");

laminaBotones=new JPanel();

this.laminaPelota=new LaminaPelotaVista();

add(this.laminaPelota, BorderLayout.CENTER);

btnIniciar=new JButton();

btnSalir=new JButton();

btnDetener =new JButton();

add(laminaBotones, BorderLayout.SOUTH);

}

public void AñadirPelota(Pelota m){

laminaPelota.add(m);

}

}

Cree otra clase Java.



Figura 52. Creación de clase Java.

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

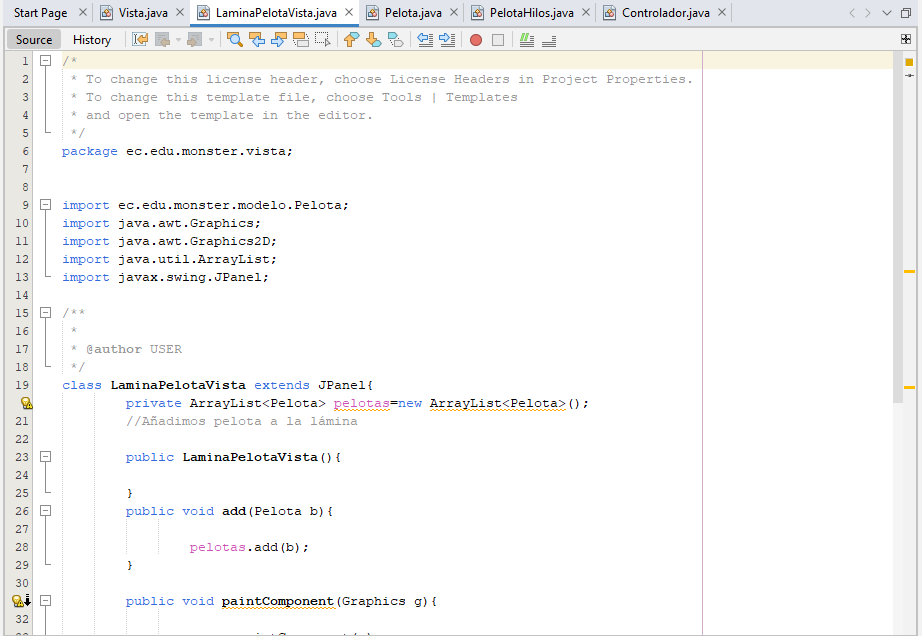


Figura 53. Codificación archivo Lamina Pelota Vista.

Tabla 15. Codificación del archivo Lamina Pelota Hilos.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.vista;

import ec.edu.monster.modelo.Pelota;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.util.ArrayList;

import javax.swing.JPanel;

/\*\*

\*

\* @author USER

\*/

class LaminaPelotaVista extends JPanel{

private ArrayList<Pelota> pelotas=new ArrayList<Pelota>();

//Añadimos pelota a la lámina

public LaminaPelotaVista(){

}

public void add(Pelota b){

pelotas.add(b);

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2=(Graphics2D)g;

for(Pelota b: pelotas){

g2.fill(b.getShape());

}

}

}

public void paintComponent(Graphics g){

super.paintComponent(g);

Graphics2D g2=(Graphics2D)g;

for(Pelota b: pelotas){

g2.fill(b.getShape());

}

}

}

}}

### CODIFICACIÓN DE LOS ARCHIVOS DEL MODELO

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



Figura 58. Creación de clase Java.

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

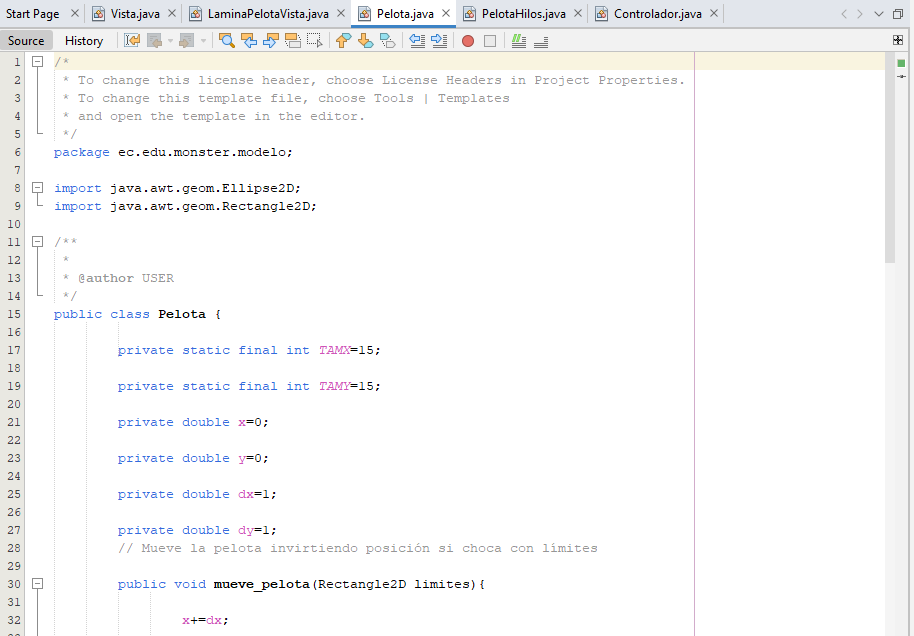


Figura 59. Codificación archivo Pelota.

Tabla 16. Codificación del archivo Pelota

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.modelo;

import java.awt.geom.Ellipse2D;

import java.awt.geom.Rectangle2D;

/\*\*

\*

\* @author USER

\*/

public class Pelota {

private static final int TAMX=15;

private static final int TAMY=15;

private double x=0;

private double y=0;

private double dx=1;

private double dy=1;

// Mueve la pelota invirtiendo posición si choca con límites

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

}

Cree una nueva clase en el paquete del modelo.



Figura 60. Creación de clase Java.

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

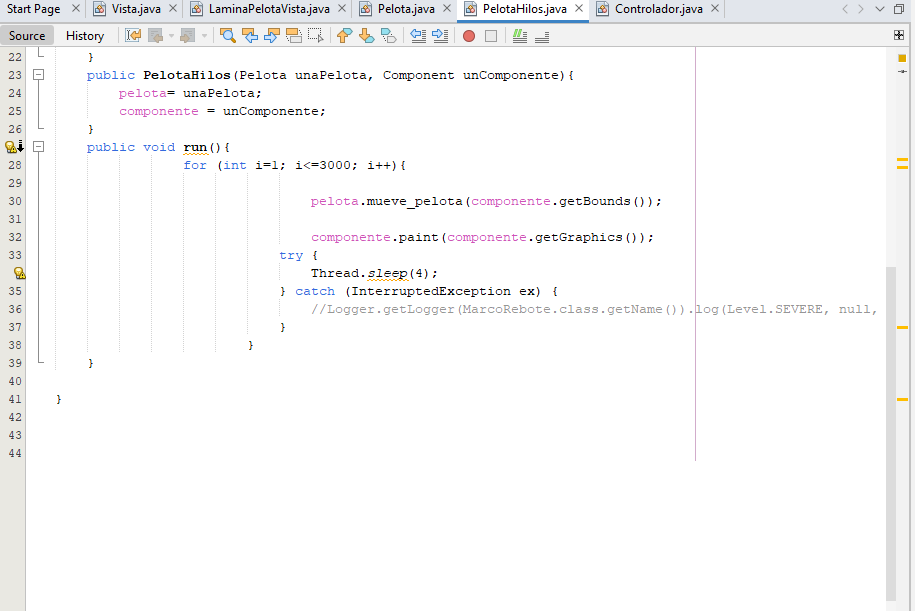


Figura 61. Codificación archivo Pelota Hilos.

Tabla 17. Codificación del archivo PelotaHilos

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.edu.monster.modelo;

import java.awt.Component;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author USER

\*/

public class PelotaHilos implements Runnable {

private Pelota pelota;

private Component componente;

public PelotaHilos(){

}

public PelotaHilos(Pelota unaPelota, Component unComponente){

pelota= unaPelota;

componente = unComponente;

}

public void run(){

while (!Thread.currentThread().isInterrupted()){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*for (int i=1; i<=3000; i++){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*try {

Thread.sleep(4);

} catch (InterruptedException ex) {

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

}

}\*/

}

}

}

while (!Thread.currentThread().isInterrupted()){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*for (int i=1; i<=3000; i++){

pelota.mueve\_pelota(componente.getBounds());

componente.paint(componente.getGraphics());

/\*try {

Thread.sleep(4);

} catch (InterruptedException ex) {

//Logger.getLogger(MarcoRebote.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 62. Creación de clase Java.

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

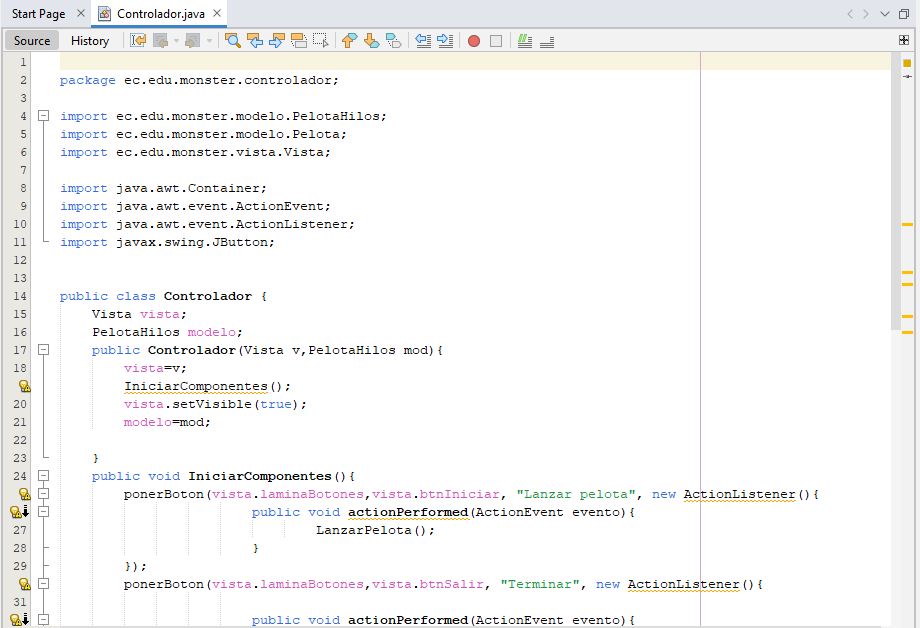


Figura 63. Codificación archivo controlador.

Tabla 18. Codificación del archivo Controlador

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.PelotaHilos;

import ec.edu.monster.modelo.Pelota;

import ec.edu.monster.vista.Vista;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

public class Controlador {

Vista vista;

PelotaHilos modelo;

Thread t;

public Controlador(Vista v,PelotaHilos mod){

vista=v;

IniciarComponentes();

vista.setVisible(true);

modelo=mod;

}

public void IniciarComponentes(){

ponerBoton(vista.laminaBotones,vista.btnIniciar, "Lanzar Pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

LanzarPelota();

}

});

ponerBoton(vista.laminaBotones,vista.btnSalir, "Salir", new ActionListener(){

public void actionPerformed(ActionEvent evento){

System.exit(0);

}

});

ponerBoton(vista.laminaBotones,vista.btnDetener, "Parar pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

detener();

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener oyente){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(oyente);

}

public void LanzarPelota (){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

t = new Thread(r);

t.start();

}

public void detener(){

//t.stop();

t.interrupt();

}

}

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener accion){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(accion);

}

public void LanzarPelota(){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

Thread t = new Thread(r);

t.start();

}

}

public void mueve\_pelota(Rectangle2D limites){

x+=dx;

y+=dy;

if(x<limites.getMinX()){

x=limites.getMinX();

dx=-dx;

}

if(x + TAMX>=limites.getMaxX()){

x=limites.getMaxX() - TAMX;

dx=-dx;

}

if(y<limites.getMinY()){

y=limites.getMinY();

dy=-dy;

}

if(y + TAMY>=limites.getMaxY()){

y=limites.getMaxY()-TAMY;

dy=-dy;

}

}

//Forma de la pelota en su posición inicial

public Ellipse2D getShape(){

return new Ellipse2D.Double(x,y,TAMX,TAMY);

}

} <%= lenguajesSeleccionados[i]%>

</li>

<%}

}

%>

</ul>

</body>

</html>

LanzarPelota();

}

});

ponerBoton(vista.laminaBotones,vista.btnSalir, "Salir", new ActionListener(){

public void actionPerformed(ActionEvent evento){

System.exit(0);

}

});

ponerBoton(vista.laminaBotones,vista.btnDetener, "Parar pelota", new ActionListener(){

public void actionPerformed(ActionEvent evento){

detener();

}

});

}

public void ponerBoton(Container c,JButton btn, String titulo, ActionListener oyente){

btn.setText(titulo);

c.add(btn);

btn.addActionListener(oyente);

}

public void LanzarPelota (){

Pelota pelota=new Pelota();

vista.AñadirPelota(pelota);

Runnable r = new PelotaHilos(pelota, vista.laminaPelota);

t = new Thread(r);

t.start();

}

public void detener(){

t.interrupt(); }

}

### 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.

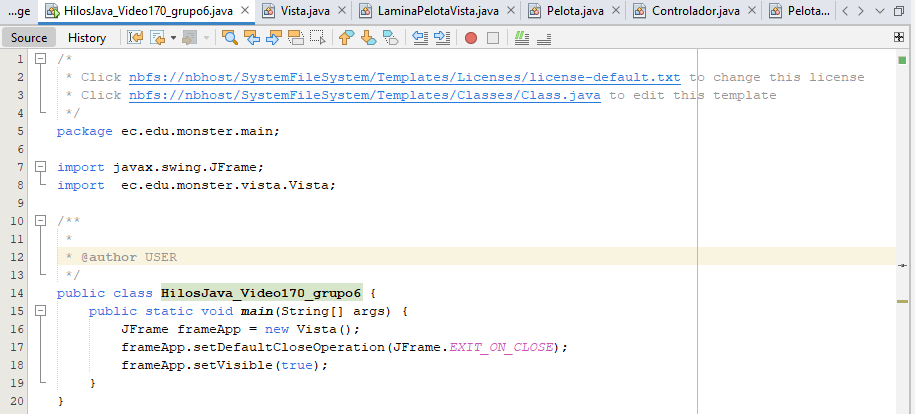


Figura 64. Codificación archivo principal del proyecto 170.

Tabla 19. Codificación del archivo Principal.

/\*

\* 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\_Video169\_grupo6 {

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, además de tener la capacidad de detener la pelota que se desee, esto se logra con las modificaciones realizadas en comparación al proyecto anterior.

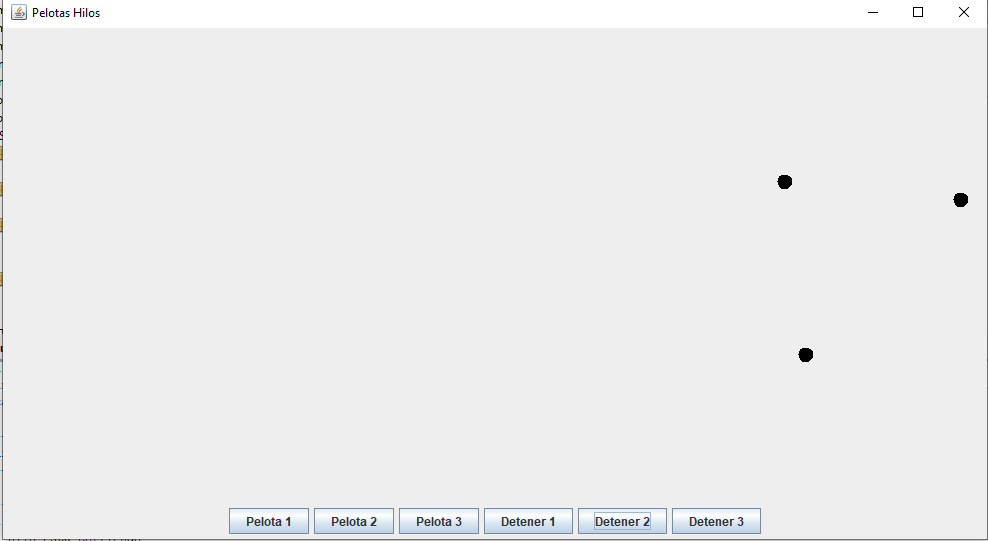


Figura 65. Resultado del proyecto 170.

# 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]. |