**Informe Implementación de Patrón Memento**

**Presentado por:**

**Carlos Bermúdez Galván**

**Miguel Angel Thomas**

**Juan Jose Barrera**

**Curso:**

**Patrones de Diseño de Software**

**Universidad Sergio Arboleda**

****

**Escuela de Ciencias Exactas e Ingeniería**

**Bogotá, Colombia**

**Octubre de 2023**

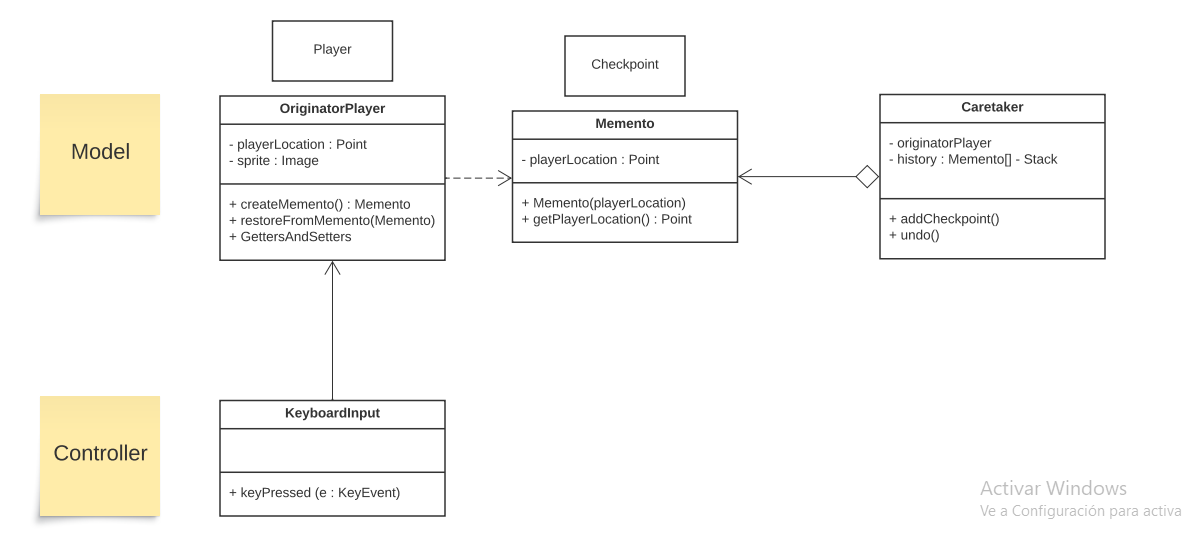
# Review de Implementación de Memento

## Creación del diagrama UML

El diagrama UML muestra la estructura del patrón Memento implementado en el proyecto.

Las tres clases principales son **OriginatorPlayer**, **Memento** y **Caretaker**.

* **OriginatorPlayer**: Esta clase representa al jugador del videojuego. Contiene atributos como la ubicación del jugador y su sprite. Los métodos createMemento y restoreFromMemento permiten crear instantáneas del estado del jugador y restaurar el estado desde un Memento.
* **Memento**: La clase Memento almacena el estado del jugador en un momento específico. En este caso, guarda la ubicación del jugador.
* **Caretaker**: El Caretaker es responsable de gestionar los checkpoints (Mementos). Puede agregar checkpoints y deshacer acciones para restaurar el estado del jugador.

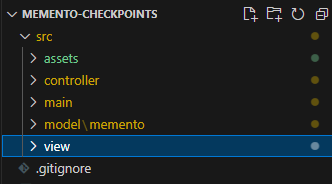


Creado en [www.lucidchart.com](http://www.lucidchart.com)

## 

## Estructura del Proyecto

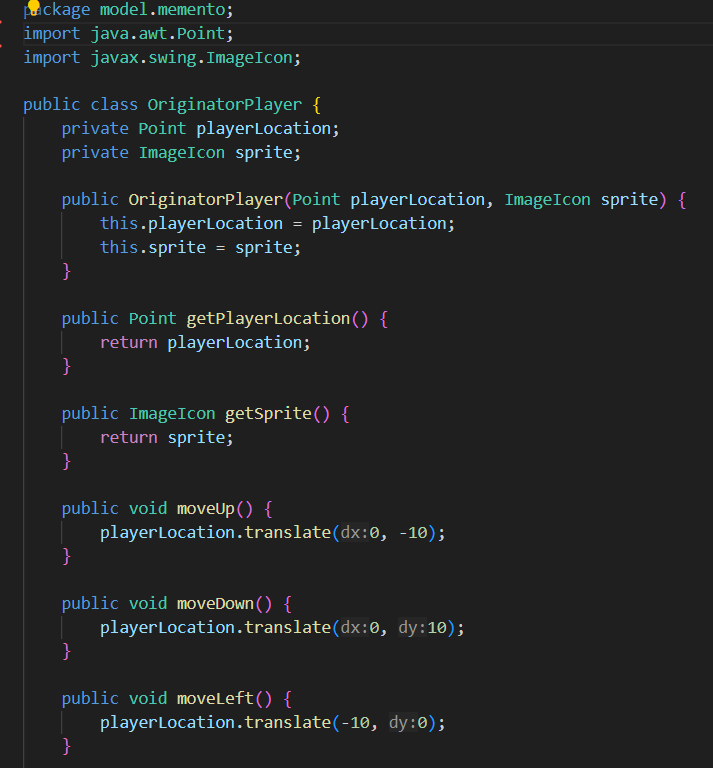
Se basa en el patrón de diseño modelo-Vista-Controlador (MVC). Este enfoque permite una gestión ordenada y eficiente de los componentes clave del juego. Además, se han agregado dos paquetes importantes para una mejor organización quedando de la siguiente manera:

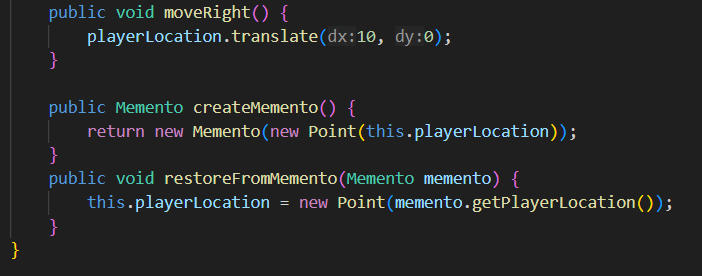


### Contenido de paquetes:

1. **Model:** En este paquete se encuentran las clases que representan el "Modelo" en el patrón MVC. Aquí reside la lógica del juego, incluyendo la implementación del patrón Memento. Las clases en este paquete son:

* OriginatorPlayer: Representa al jugador del juego y almacena su estado, como su ubicación y su sprite.

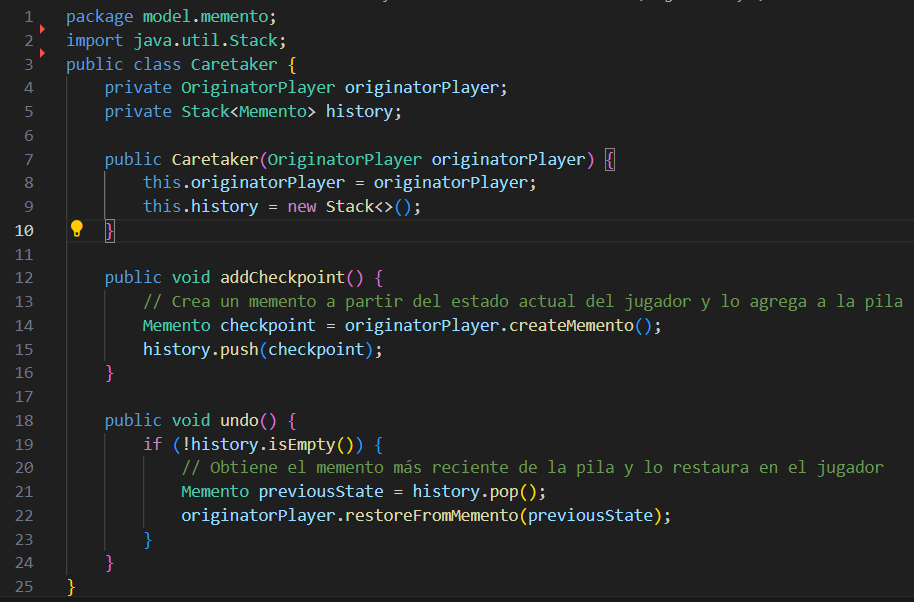




* Memento: Esta clase almacena instantáneas del estado del jugador en un momento específico.



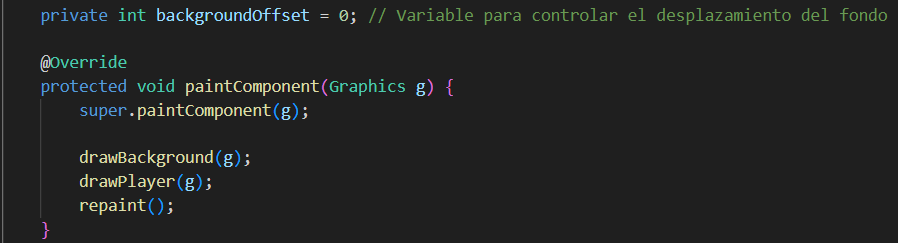
* Caretaker: Administra los checkpoints (Mementos) y permite restaurar el estado del jugador a través de estos checkpoints.

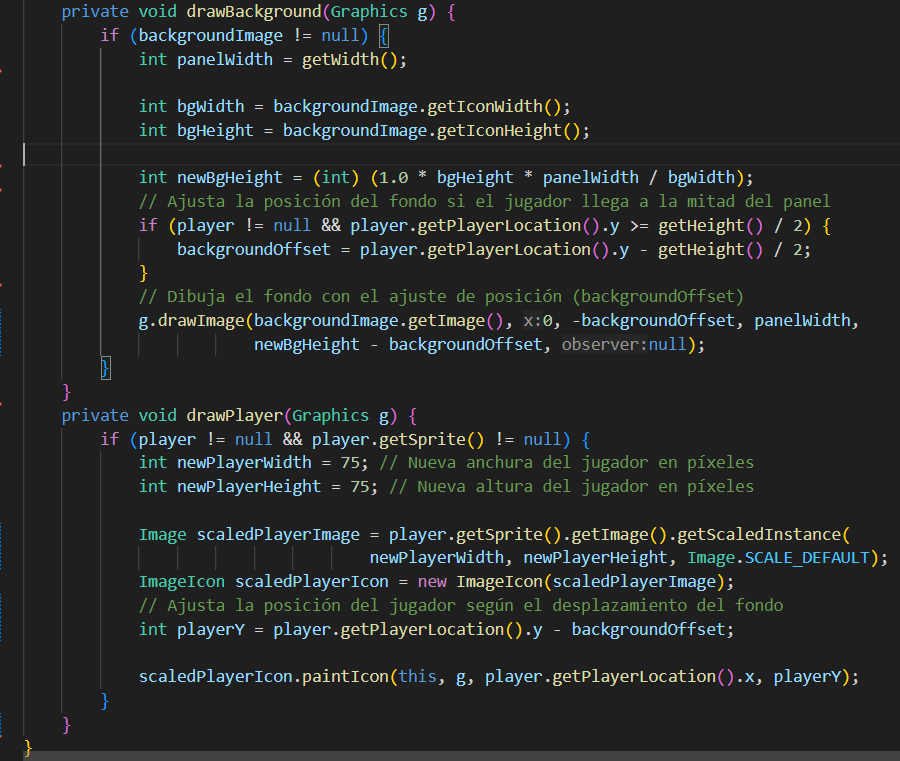


1. **View:** En este paquete se encuentran las clases relacionadas con la interfaz de usuario y la "Vista" en el patrón MVC. Las clases aquí son responsables de mostrar elementos gráficos en pantalla. Las clases en este paquete son:

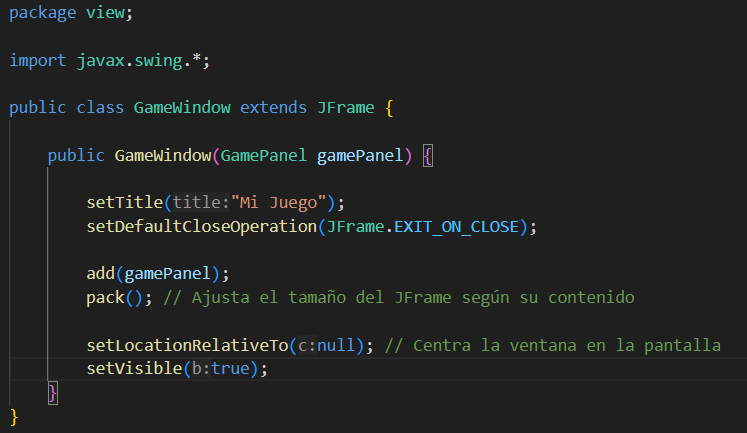
* GamePanel: Representa el panel de juego en el que se muestran elementos visuales, como el jugador y el fondo. Se ajusta al tamaño de la pantalla y gestiona la representación de los objetos en el juego.



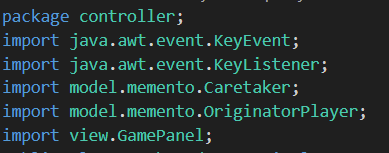


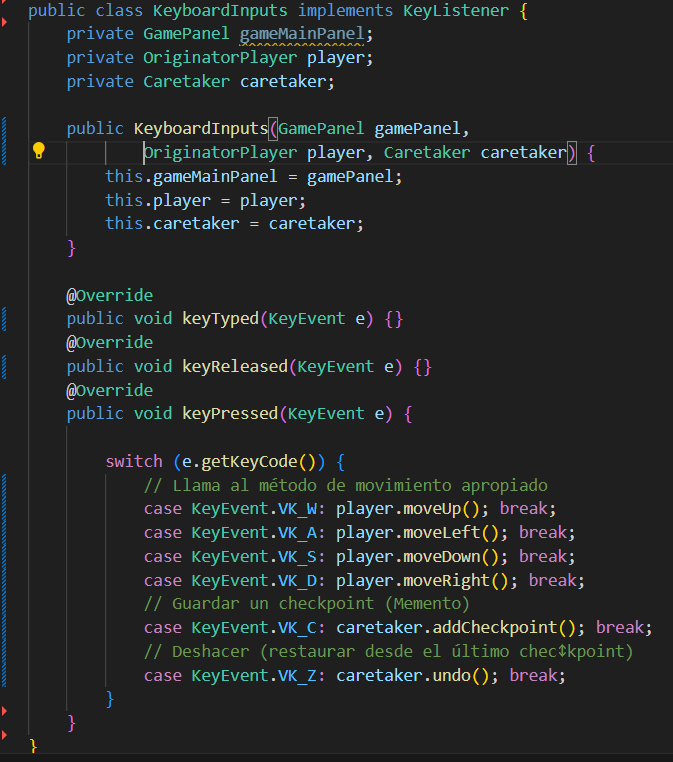


* GameWindow: La ventana del juego que contiene el GamePanel. Define la ventana principal en la que se ejecuta el juego.



1. **Controller:** Aunque no se ha incluido un paquete específico para el "Controlador", la funcionalidad de control se maneja en la clase KeyboardInputs en el paquete raíz. Esta clase se encarga de escuchar las entradas del teclado y controlar las acciones del jugador, incluyendo la creación y restauración de checkpoints.



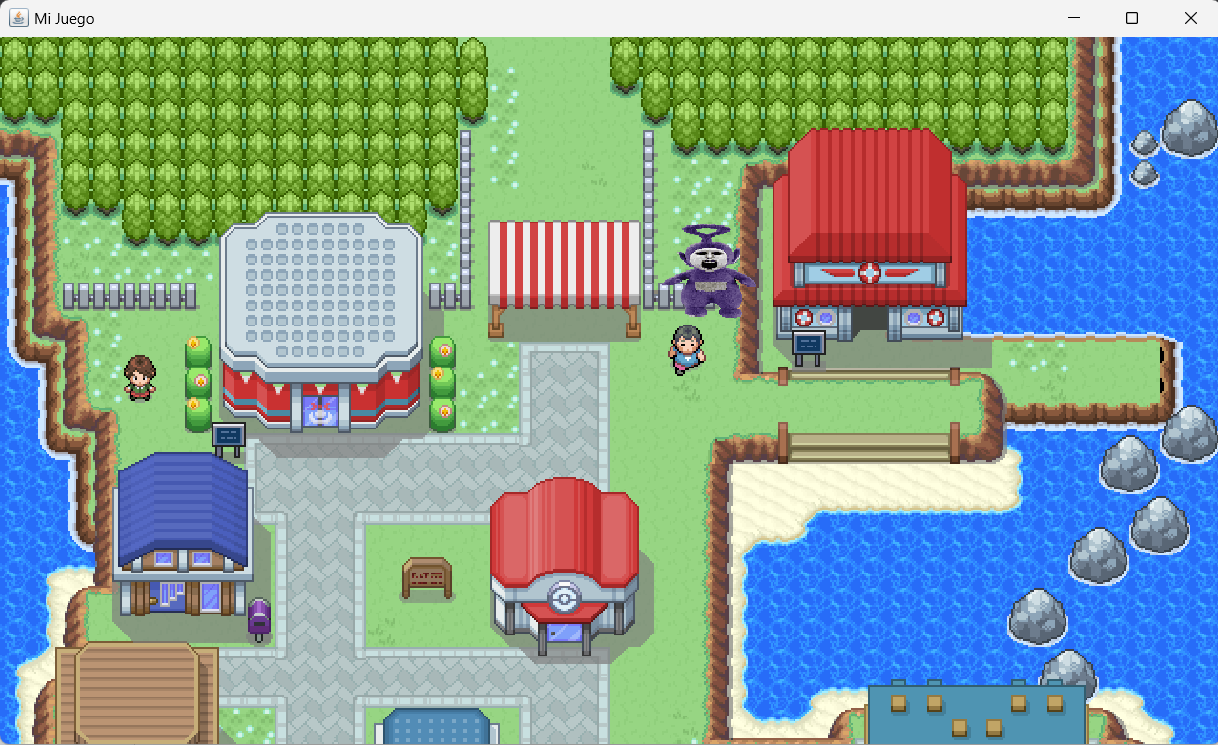


1. **Assets:** Aquí se almacenan los recursos gráficos utilizados en el juego, como imágenes en formato PNG.
2. **Main:** En este paquete se encuentra la clase Game. En esta clase, se inicializan los componentes esenciales, como el GamePanel, GameWindow, el jugador y el Caretaker.

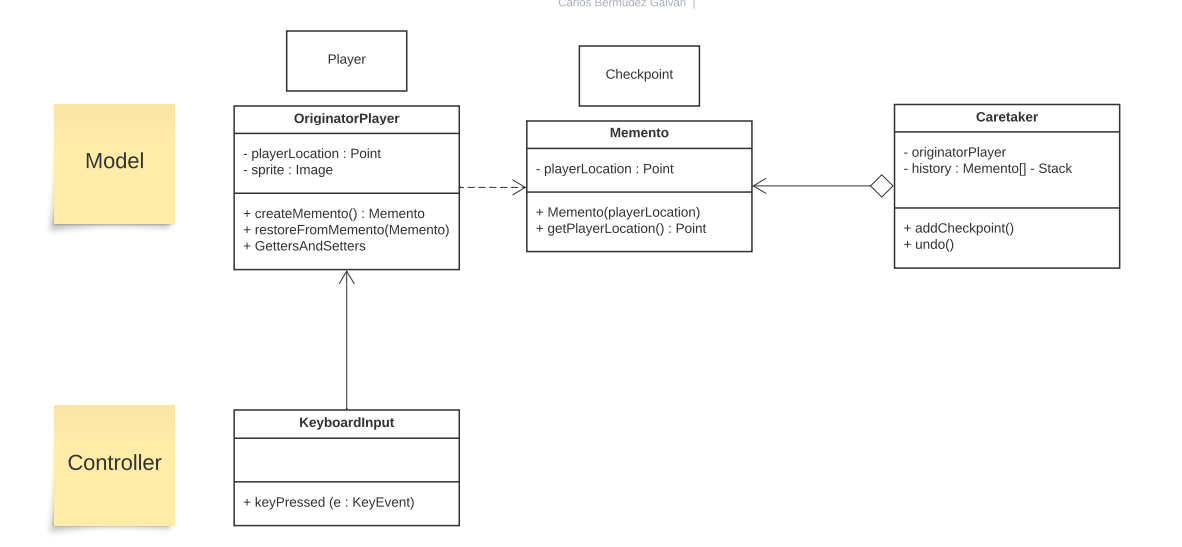


Esta estructura facilita la separación de responsabilidades y la escalabilidad del proyecto. Permite una gestión eficaz de los recursos gráficos, el flujo del juego y la interacción del jugador.

## Visualización Del Programa En Ejecución



**Diagrama UML**

****