FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2022

Nombre	Padron	Email
Corrionero, Luan Shair	102439	lcorrionero@fi.uba.ar
Meichtri, Melany	102330	mmeichtri@fi.uba.ar
Ramos Sarzuri, Claudia Lis	104596	cramos@fi.uba.ar
Francisco Sobral 108603	fsobral@fi.uba.ar	
Arballo, Facundo Nicolas	105096	farballo@fi.uba.ar

Índice

1. Supuestos

- Los vehículos de distintos jugadores pueden estar en una misma esquina.
- Intentar moverse en una cuadra con un piquete sin usar moto suma dos movimientos y se pierde el turno.
- Atravesar cuadras no consume los obstáculos o sorpresas que pueda haber en ellas.
- El mapa, su tamaño y sus elementos (obstáculos, sorpresas y la meta) son fijos.
- Primero se aplican los obstáculos y luego las sorpresas.
- Los jugadores empiezan todos en la misma posición.
- Si un jugador realiza un movimiento a una zona del mapa no disponible (fuera de los límites del mismo), se pierde el turno y se suman dos movimientos.
- El jugador que comienza jugando es el último que se agrega

2. Diagramas de clase

Un diagrama de clase es un diagrama estático en el cual se representa la estructura de un sistema compuesto por clases, reflejando así sus atributos, métodos y las relaciones con otros objetos. A continuación se presentan algunos diagramas de clase correspondientes al trabajo practico.

2.1. Diagrama de Clases General

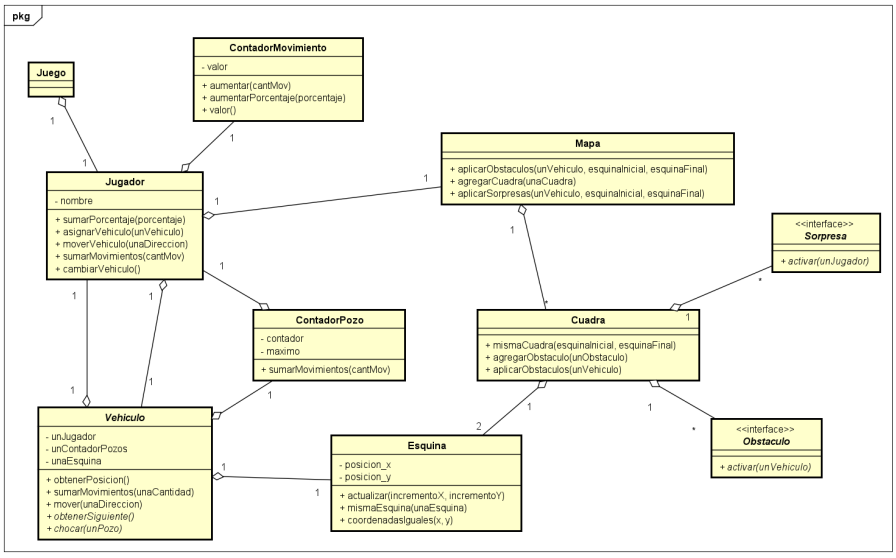

Figura 1: Diagrama de Clases de GPS Challenge.

2.2. Diagrama de Vehículos

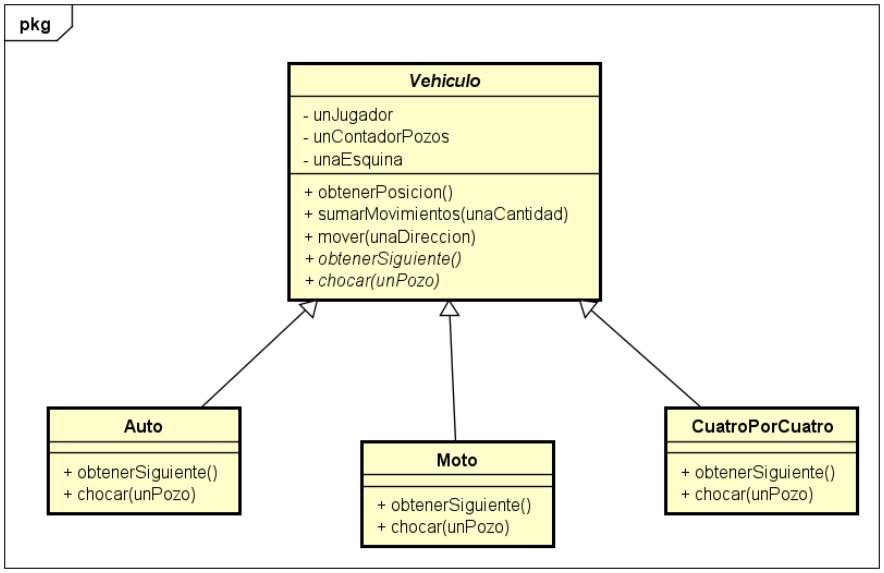

Figura 2: Diagrama de los Vehículos.

2.3. Diagrama de Direcciones

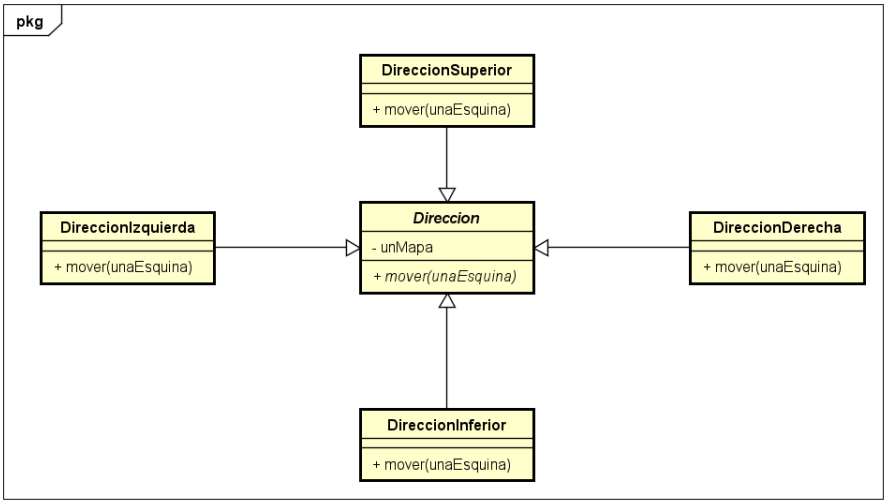

Figura 3: Diagrama de las Direcciones.

2.4. Diagrama de Obstáculos

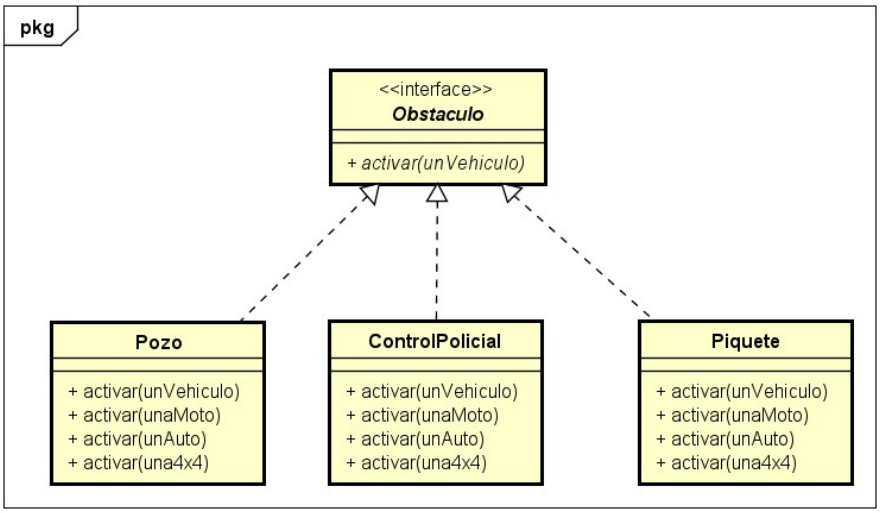

Figura 4: Diagrama de los Obstáculos.

2.5. Diagrama de Sorpresas

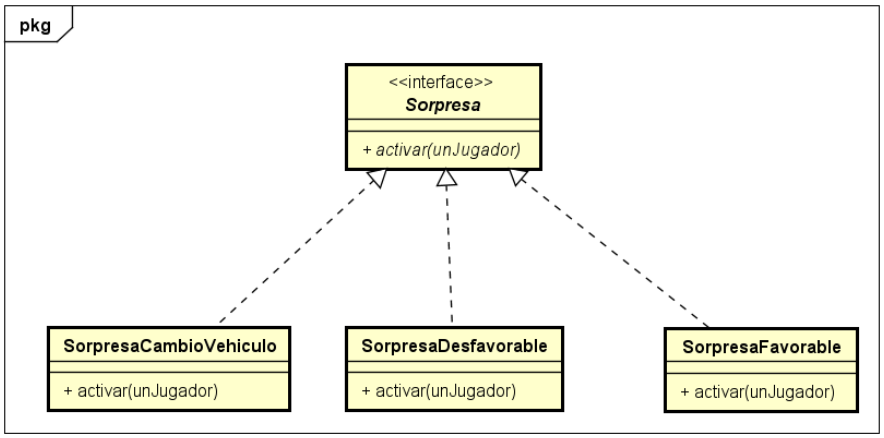

Figura 5: Diagrama de las Sorpresas.

3. Diagramas de secuencia

Un diagrama de secuencia muestra la interacción entre un conjunto de entidades en un determinado caso de uso a lo largo del tiempo. A continuación se presentaran diferentes diagramas que muestran diferentes situaciones dadas dentro del modelo de dominio

3.1. Secuencia de Movimiento General

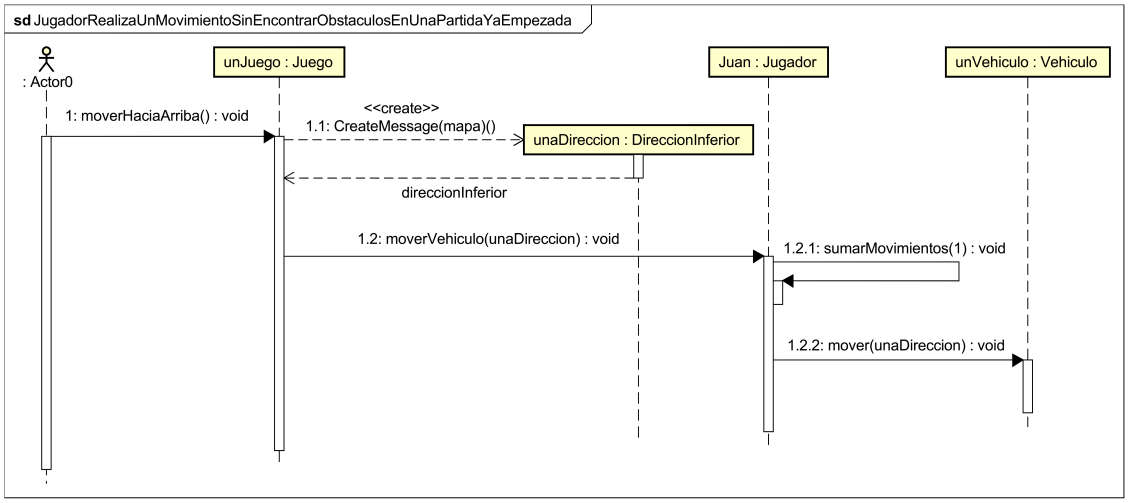

Figura 6: Diagrama de como se lleva a cabo un movimiento.

3.2. Secuencia de activación de obstáculo y sorpresa a Jugador de turno

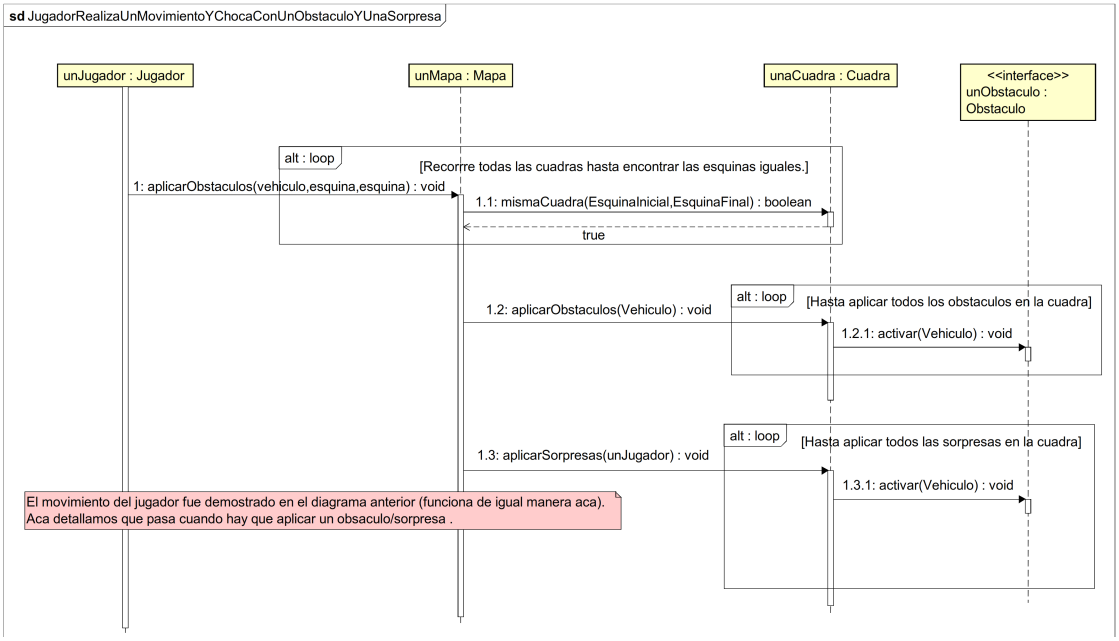

Figura 7: Diagrama de como se acciona un obstaculo/sorpresa.

3.3. Secuencia de Jugador llegando a la meta

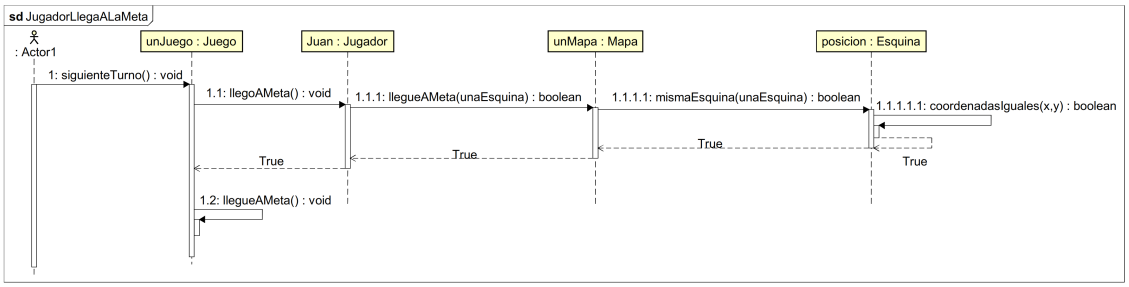

Figura 8: Diagrama de que ocurre cuando llega un jugador a la meta.

4. Diagramas de contenedores

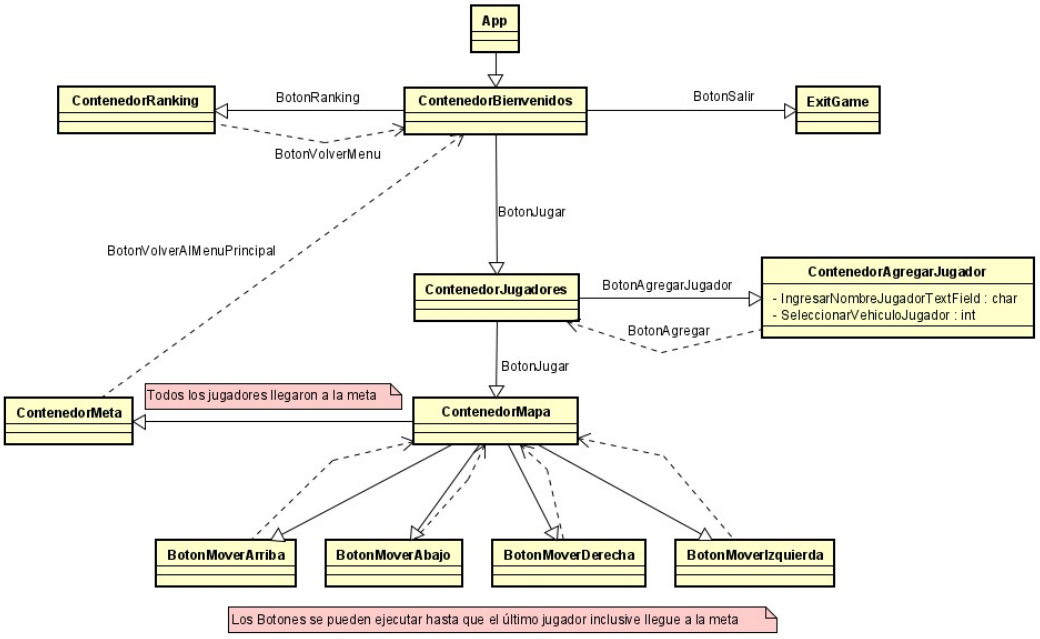

Figura 9: Diagrama de contenedores mostrando la implementación de JavaFX.

5. Diagramas de paquetes

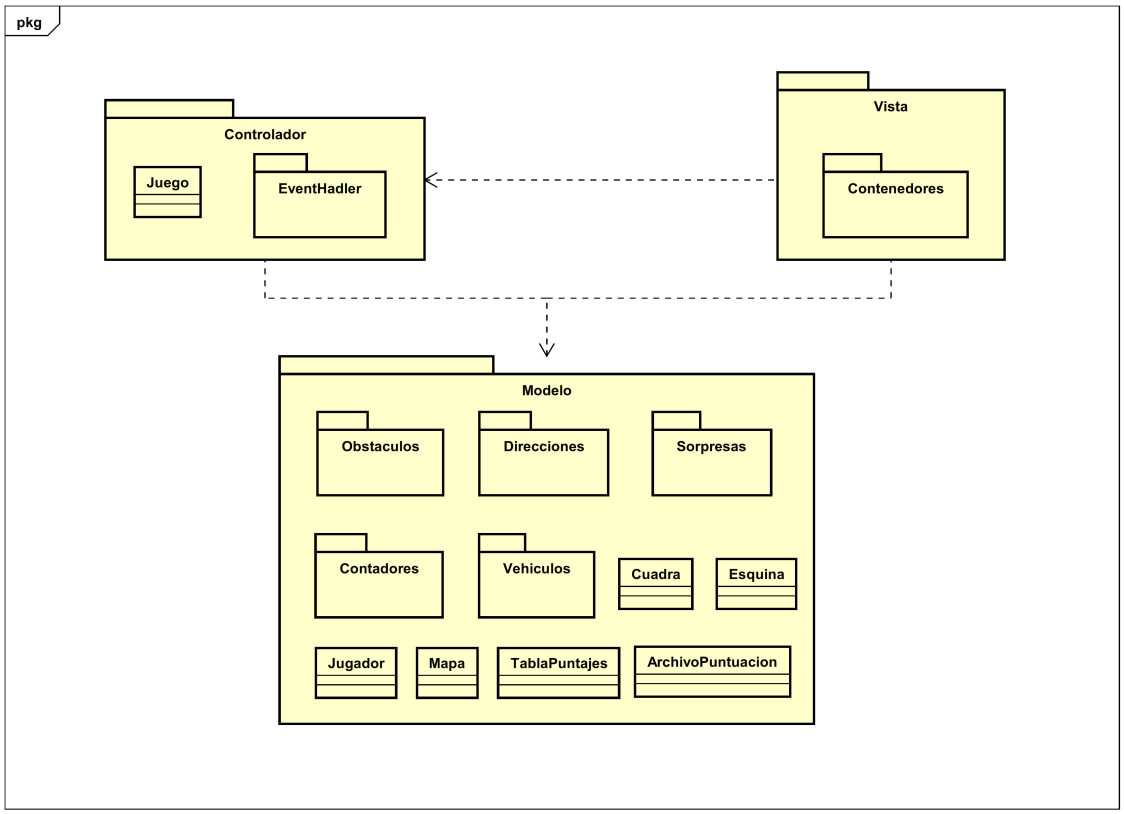

Figura 10: Diagrama de paquetes general utilizando el patrón MVC.

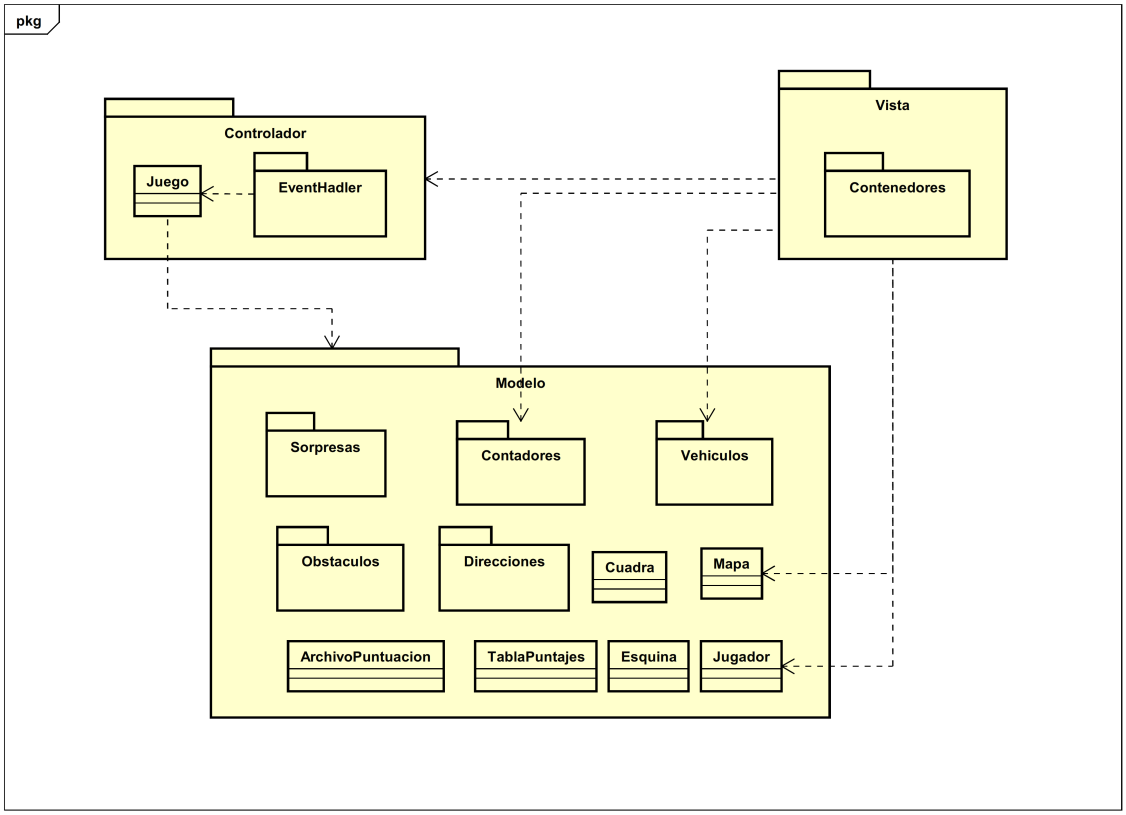

Figura 11: Diagrama de paquetes mostrando las relaciones entre las clases de los diferentes paquetes.

6. Detalles de implementación

Las clases usadas, junto con como interactúan se enuncian en la descripción de cada pilar de la programación orientada a objetos

6.1. Interacción JavaFX y modelo del Juego

Toda interacción en cuanto a código implementado, y la parte visual del presente proyecto fue realizado a través de la clase Juego. La configuración previa al juego consiste en la asignación de jugadores uno por uno, los cuales poseen característicamente un nombre (String) y un vehículo con el que van a jugar (clase abstracta Vehículo, las instancias pueden ser Moto, Auto y CuatroPorCuatro).

Una vez realizada la asignación de las personas, la interacción entre usuario y programa se realiza mediante 4 botones que aparecen en pantallas, los cuales corresponden a los movimientos posibles que puede realizar cada uno (arriba, abajo, izquierda, derecha). Una vez presionado un botón, se realiza ejecuta la acción sin pedir confirmación y se avanza al siguiente jugador de turno.

En cada momento, se muestra por pantalla el nombre del jugador de turno para saber a quien le toca mover.

6.2. Uso de double dispatch para los vehículos

Double dispatch es un patrón que permite tomar una decisión a partir de varios objetos en vez de uno solo. En este caso, nosotros decidimos utilizarlo para los vehículos, ya que según el vehículo

que tuviese una acción en el mapa, seria el calculo que se realizaría, teniendo cada uno su propia respuesta

6.3. Uso de herencia para las distintas direcciones

En el caso de las direcciones decidimos usar una clase madre abstracta denominada Direccion, la cual hereda a sus hijos DireccionDerecha, DireccionInferior, DireccionIzquierda, y DireccionSuperior. Donde además de cumplir con el contrato de 'es un', respetan el contrato de la clase madre en su totalidad

6.4. Ranking almacenado en archivo .csv

Para mantener un registro de los juegos previos, se posee un archivo csv que guarda el par clave valor (nombre de jugador y puntuación). Si se va a la sección de ranking, se pide ingresar la clave para consultar por los juegos previos con ese nombre, y luego por pantalla se muestran en orden de mejor a peor juego correspondientes a ese nombre.

6.5. Uso del patrón MVC

Utilizamos el patrón MVC para organizar mejor las dependencias entre el modelo y la interfaz. Se logró un diseño menos acoplado entre parte visual y lógica interna, utilizando como intermediario a la clase Juego. Si se desea realizar cambios tanto en JavaFx como en la estructura del programa, no repercute de manera drástica en la parte vista desde el usuario.

7. Conclusiones

A lo largo del desarrollo del presente trabajo, se hizo uso de una serie nueva de herramientas y conocimientos adquiridos a lo largo de la materia. Se abandonó el pensamiento de programación estructurada para dar a lugar a la programación orientada a objetos. Se utilizaron e implementaron los conceptos de polimorfismo, delegación, interfaces y buenas prácticas de desarrollo.

En todo momento, el código implementado fue pensado para lograr un bajo acoplamiento entre los distintos módulos del programa, con el objetivo de crear una base sólida de principio a fin para que sea sencillo el crecimiento del juego semana a semana.

Otro factor importante a destacar fue la introducción a una metodología ágil de trabajo, la cual consistió en no ver al proyecto como un todo, sino verlo como un conjunto de **sprints** o desarrollos de corta duración, sobre la cual, llegada la fecha estipulada, se concluiría con el programa completo y terminado.

Se considera que se logró cumplir con lo pedido en cuanto a implementación, se completó la entrega de un juego funcional y con las características necesarias en el tiempo estipulado, y lo más importante de todo, se realizó una puesta en práctica de los conocimientos teóricos adquiridos a lo largo de la materia.