Taller 3 Diseño y Programación Orientada a Objetos.

integrantes:

- Juanita Gil Arango - 202111556
- Santiago Forero - 202111446
- Jaime Esteban Alfonso - 202116525

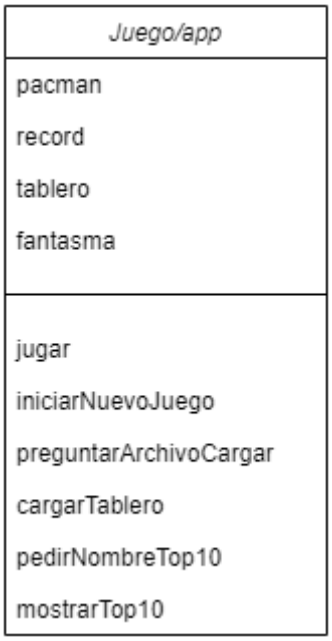
Iteración 1:

roles

- juego/app: no se le puede asignar estereotipo porque es muy general.

responsabilidades

- jugar
- iniciar un nuevo juego
- preguntar qué archivo cargar para el tablero
- cargar el tablero
- pedir nombre al usuario si entra en el top 10
- mostrar el top 10 actual



Iteración 2:

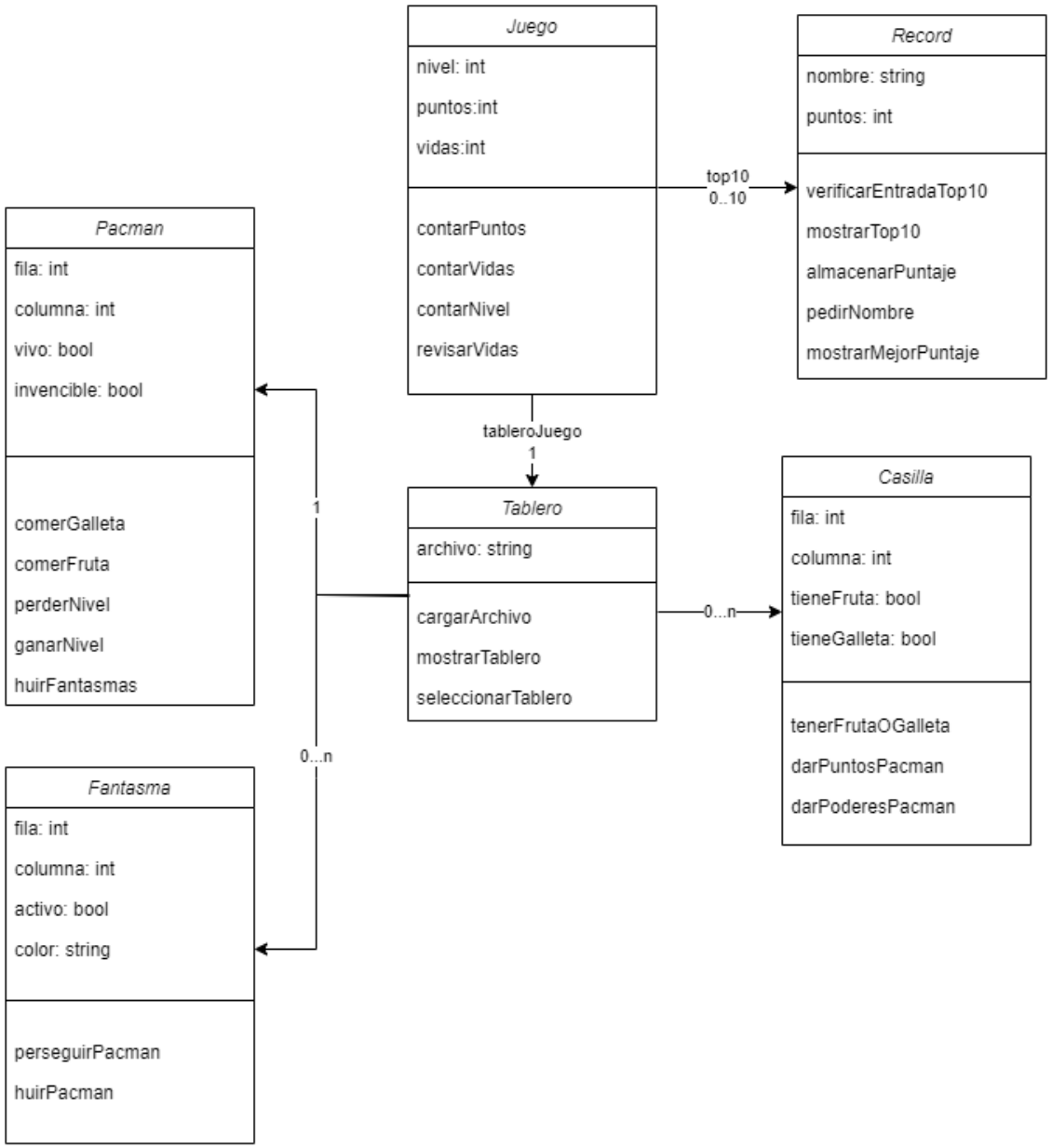
roles

- jugador -> *Controller*
  - es el usuario que está jugando el juego
- juego-> *information holder*
  - representa el estado actual del juego y además agrupa al resto de elementos que hacen parte de un juego; puntos, nivel y vidas.
- récord-> *information holder*
  - es un elemento dentro del Top-10 (el registro de los 10 mejores puntajes en el juego)
- pac-man->*information holder*
  - es el protagonista del juego y siempre debe estar en una casilla, mientras esté vivo
- tablero -> *Structurer*
  - El tablero tiene un conjunto de casillas, organizadas en filas y columnas. Cada tablero se carga de un archivo.
- casilla-> *information holder*

        - Cada casilla puede estar vacía, puede tener una fruta o una galleta, pero no ambas a tiempo.
    - fantasma-*>information holder*
        - son los antagonistas del juego que persiguen a Pacman. Cuando están vivos, se encuentran en alguna casilla del juego y cuando no, regresan al origen del tablero. Cada fantasma tiene un color diferente que permite identificar su comportamiento

responsabilidades

- Mover a Pac-Man en una de las 4 direcciones
- Comer galletas
- Huir de los fantasmas
- Comer frutas
- Ganar nivel
- Perder una vida
- Verificar vidas restantes
- Contar puntaje
- Contar nivel
- Perder nivel
- Cargar el tablero
- Mostrar el tablero
- Seleccionar tablero a jugar
- Mostrar Top 10 puntajes más altos
- Verificar si entro al Top 10
- Perseguir a Pac-Man
- Huir de Pac-Man
- Asignar comportamiento fantasma
- Agregar al Top 10
- Agregar fantasmas



Iteración 3

Roles

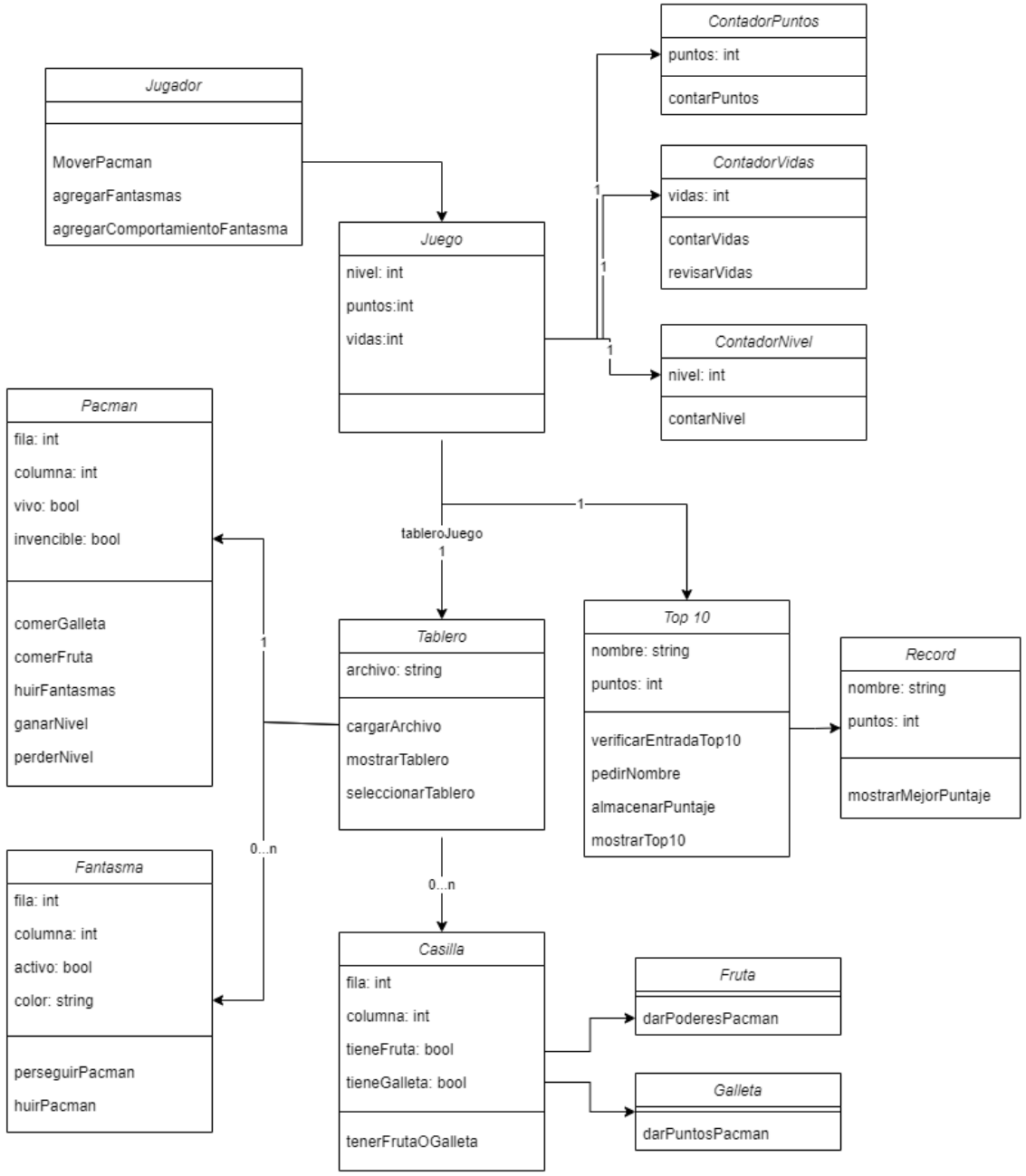
- jugador-> *controller*
  - el usuario que juega el juego
- contador puntos-> *information holder*
  - La cantidad de puntos es la que haya acumulado durante un juego completo, no durante un tablero.
- contador vidas-> *information holder*
  - la cantidad de vidas que le quedan al jugador. Máximo puede tener 3 vidas, cuando se le acaban todas, pierde el juego.
- contador nivel-> *information holder*
  - el nivel en el que está el jugador
- top 10 -> *information holder*
  - el top 10 de mejores puntajes de un tablero
- record-> *information holder*
  - el mejor puntaje dentro del top 10
- pac-man-> *information holder*

    - es el protagonista del juego y siempre debe estar en una casilla, mientras esté vivo
- tablero-> *structurer*
    - El tablero tiene un conjunto de casillas, organizadas en filas y columnas. Cada tablero se carga de un archivo.
- archivo-> *information holder*
    - el archivo que se carga en el tablero
- casilla-> *information holder*
    - Cada casilla puede estar vacía, puede tener una fruta o una galleta, pero no ambas a tiempo.
- fruta->*information holder*
    - Le dan “poderes” al Pac-Man, cuando se come una de estas frutas es invencible durante 30 segundos y puede cazar a los fantasmas, si los atrapa, vuelven a su celda.
- galleta->*information holder*
    - Son las que le dan puntos al Pac-Man, una vez las haya terminado, gana el juego.
- fantasma-> *information holder*
    - son los antagonistas del juego que persiguen a Pacman. Cuando están vivos, se encuentran en alguna casilla del juego y cuando no, regresan al origen del tablero. Cada fantasma tiene un color diferente que permite identificar su comportamiento

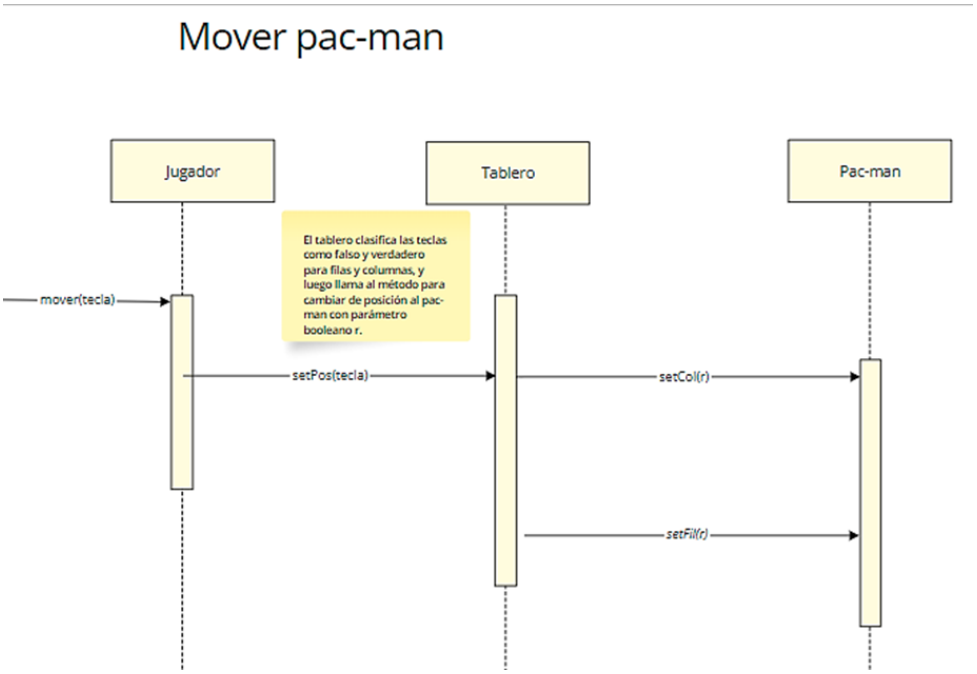
responsabilidades

#	Responsabilidad	Componente/rol
1	Mover a Pac-Man en una de las 4 direcciones	jugador
2	Contar los puntos que el jugador haya acumulado durante un juego completo.	contador puntos
3	Contar las vidas que le quedan disponibles al jugador durante un juego	contador vidas
4	Revisar que el jugador tenga disponible mínimo una vida y máximo tres.	contador vidas
5	Contar el nivel en que el jugador se encuentra.	contador nivel
6	almacenar los mejores 10 puntajes	top 10
7	mostrar los mejores 10 puntajes	top 10
8	mostrar el mejor puntaje dentro del top 10	record
9	comer galletas	pac man
10	comer frutas	pac man
11	huir de los fantasmas	pac man

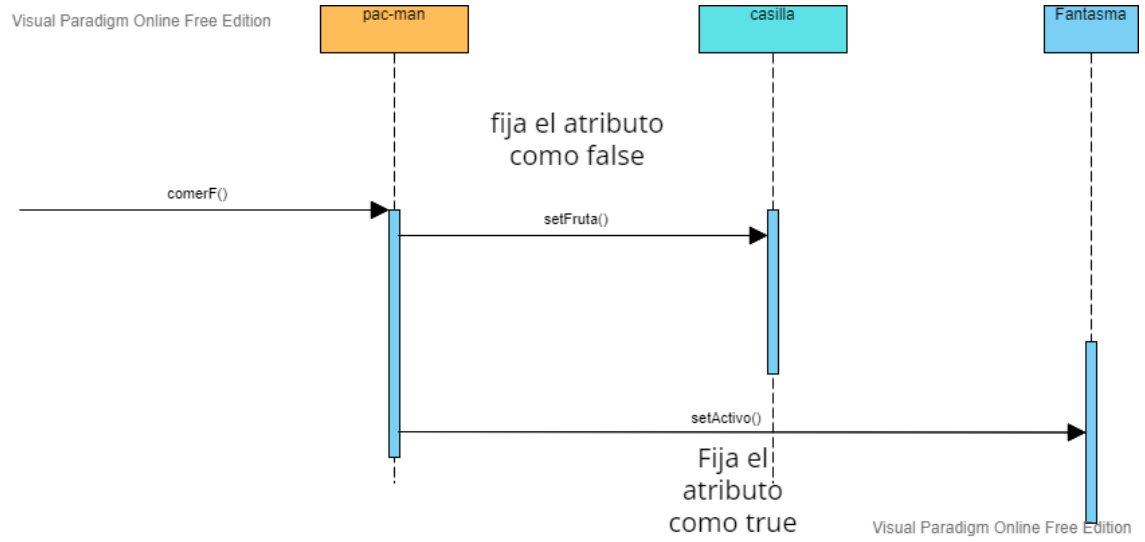
12	ganar nivel	pac man
13	perder nivel	pac man
14	cargar archivo	tablero
15	mostrar tablero	tablero
16	seleccionar en qué tablero jugar	tablero
17	verificar si entró al top 10	top 10
18	tener fruta o galleta	casilla
19	dar poderes al pac-man por 30 seg	fruta
20	dar puntos al pac-man	galletas
21	perseguir a Pac-Man	fantasma
22	huir de Pac-Man cuando esté tiene poderes	fantasma
23	agregar fantasmas	jugador
24	asignar comportamiento fantasma	jugador
25	pedir nombre a usuario en caso de que su puntaje haga parte del top 10	top 10



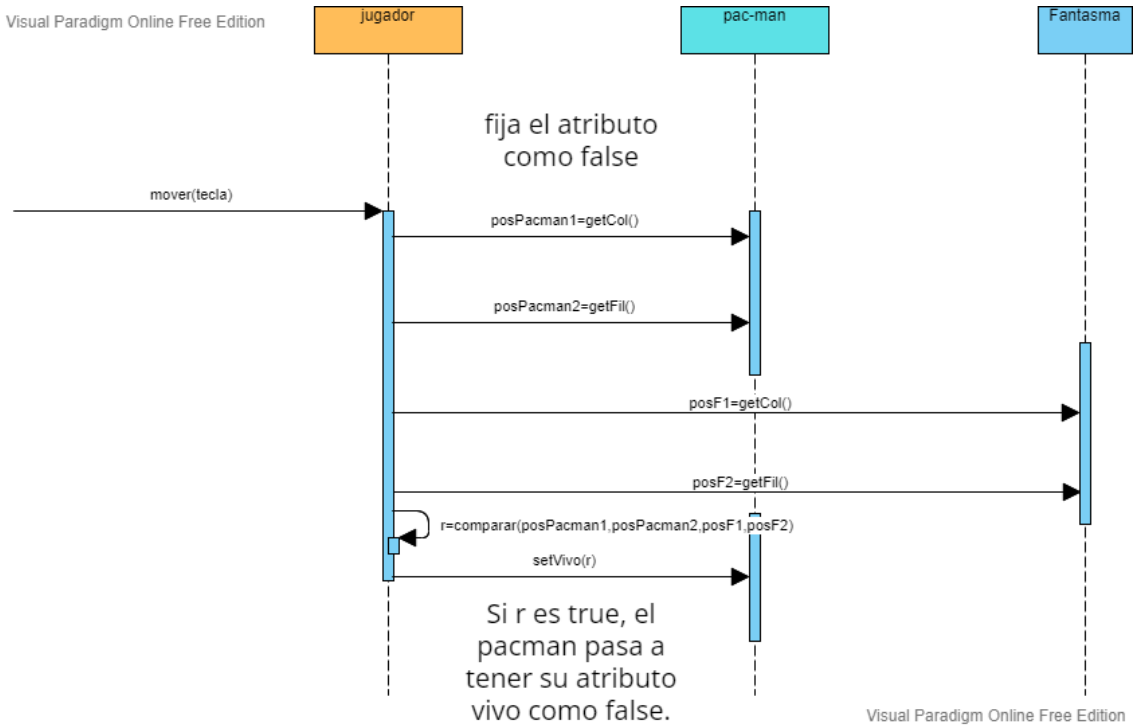
colaboraciones:



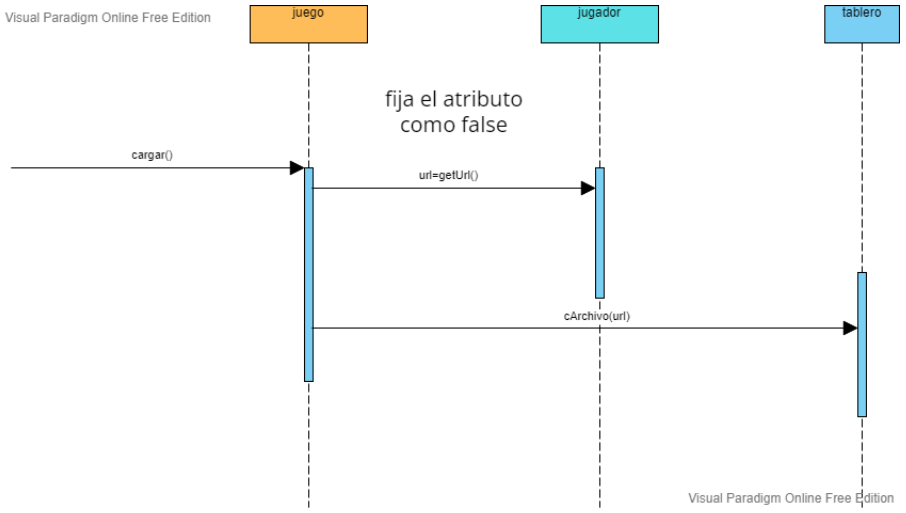
Comer frutas:



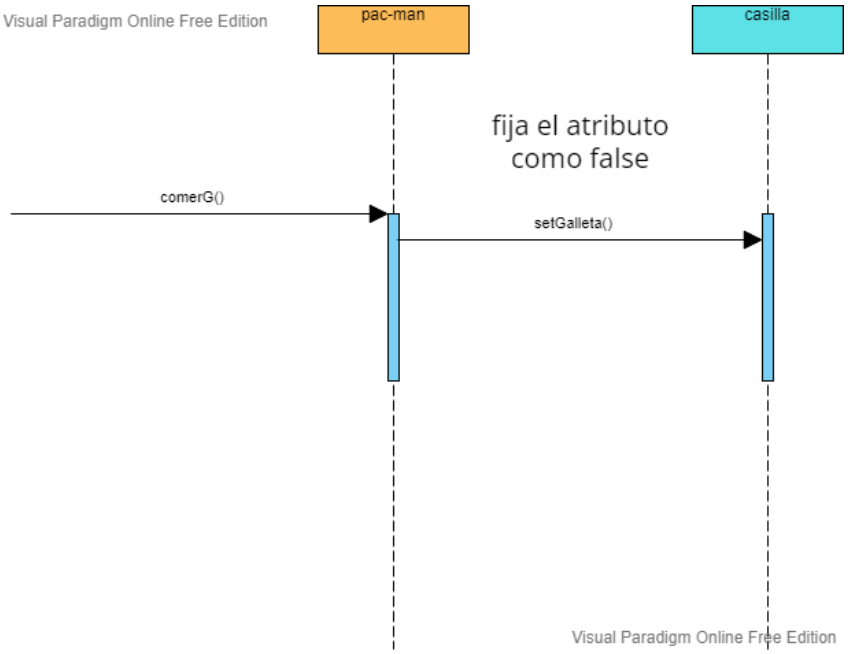
Perder Vida:



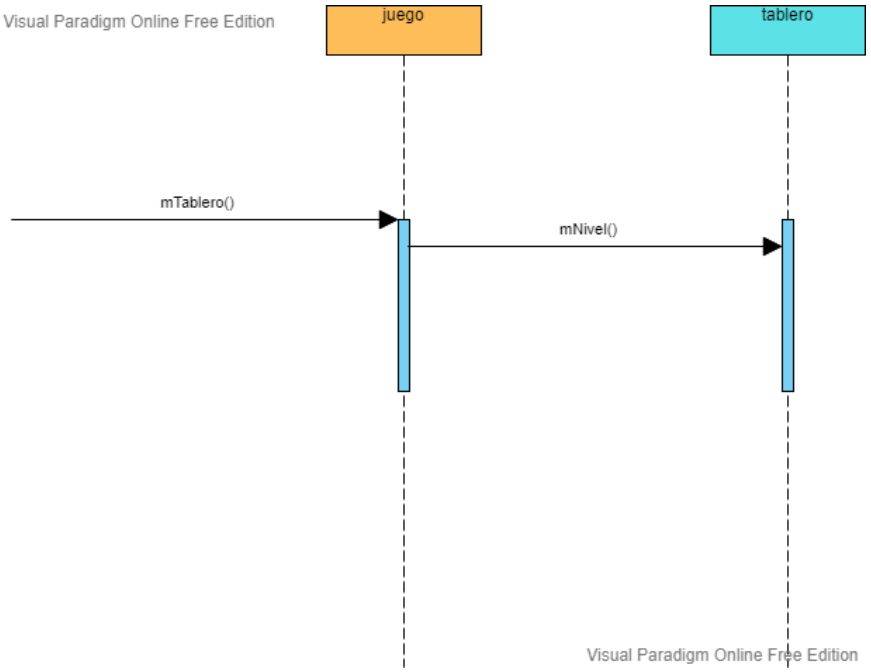
Cargar tablero:



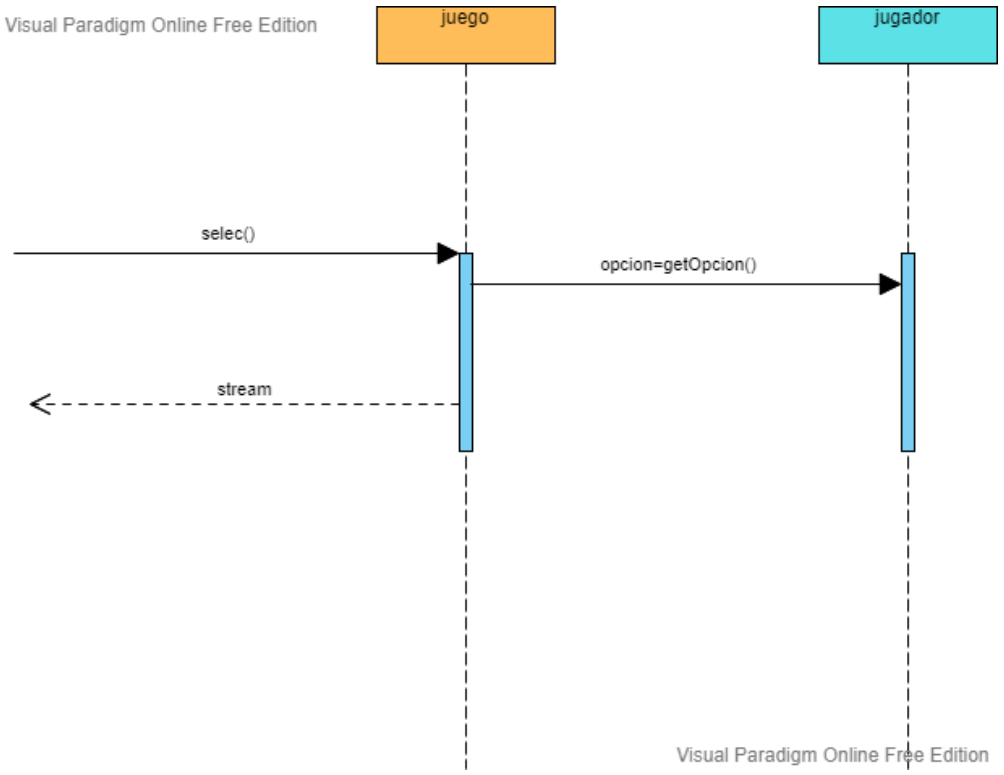
Comer galletas:



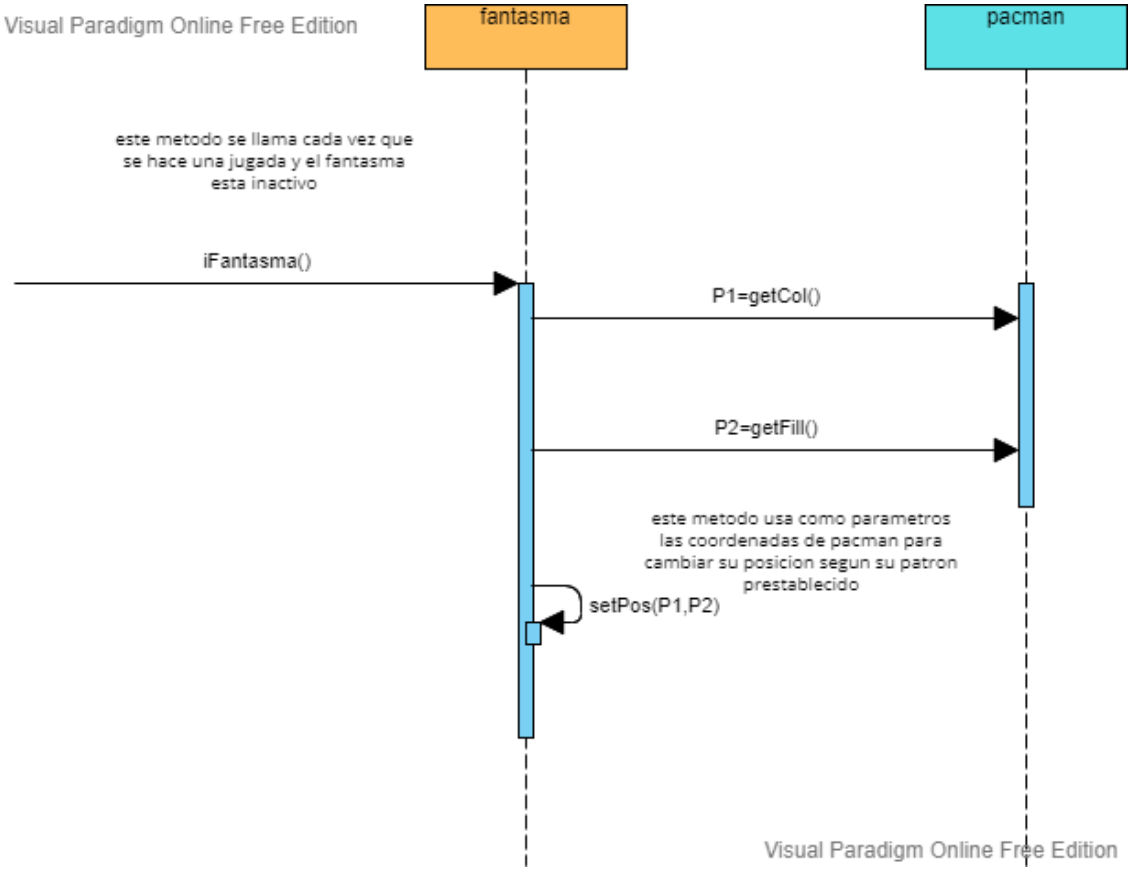
Mostrar tablero:



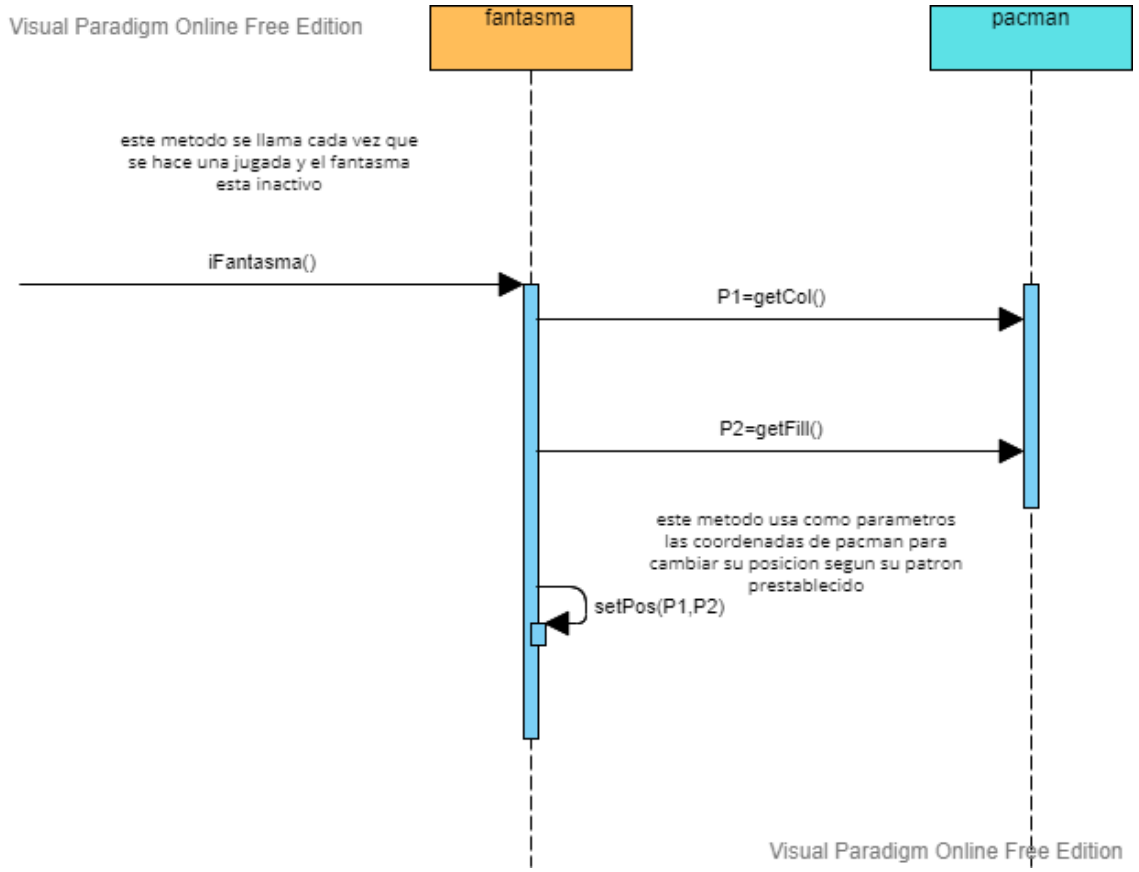
Seleccionar tablero:



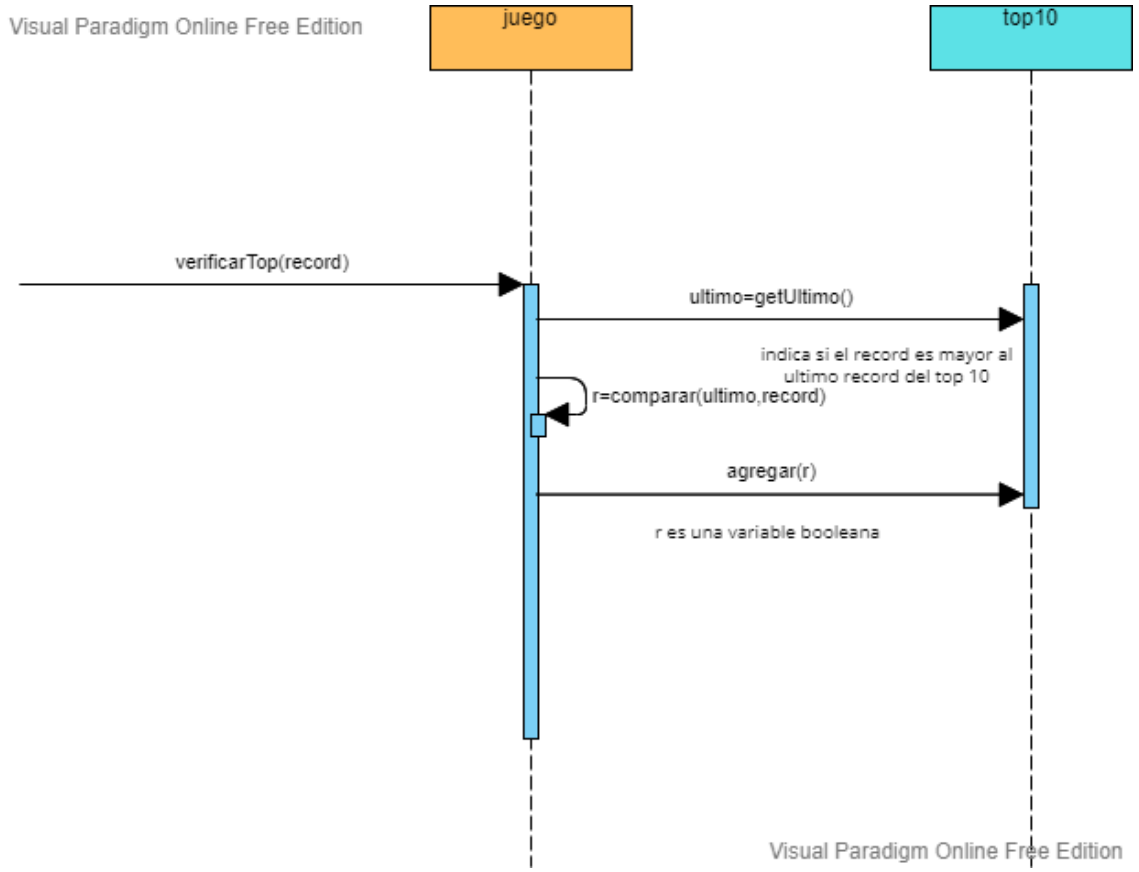
Perseguir pacman:



Huir de pacman:



Verificar record:



Reflexiones:

1. Nuestro diseño logra plasmar de forma muy cercana cómo se podría implementar una solución para realizar el videojuego pacman con las especificaciones dadas en el taller. Gracias a que separamos los componentes básicos enunciados en la etapa de análisis en componentes más pequeños, pudimos separar el problema en muchos requerimientos más pequeños que aislaban de cierta forma funcionalidades como el conteo de puntos, el movimiento de los fantasmas, etc. Aunque lo anterior signifique tener que hacer un gran número de clases y conectarlas entre ellas (lo cual puede llegar a resultar confuso en algunos casos), permite que el código sea más organizado y entendible, además, es más efectivo a la hora de identificar y corregir errores. Nuestro diseño, permitiría aumentar el número de clases sin mucho problema, sin embargo, decidimos que sería bueno dejarlo con las clases que tenemos, puesto que dividirlo más en este punto puede llegar a ser muy rebuscado y sin utilidad para lograr el objetivo deseado de manera eficiente. con respecto al top 10, decidimos hacerlo con base en una lista, ya que es más fácil ordenarla y mantenerla ordenada, aunque sea más difícil incluir los nuevos records que en un mapa. Por el contrario, para lograr agregar fácilmente nuevos fantasmas con sus propias características (como se pedía en el enunciado) decidimos usar un mapa de los fantasmas, permitiendo acceder a cada uno de ellos fácilmente. También se podría disminuir el número de colaboraciones, sin embargo, nos pareció mejor mantenerlas para que si se quisiera añadir una nueva funcionalidad, se pueda hacer sin necesidad de agregar nuevos componentes en exceso, permitiendo reutilizar de distintas formas los componentes ya existentes.

2. El proceso de diseño fue algo tedioso, ya que cada vez que creíamos que podíamos dejarlo como estaba y ya funcionaria, surgían nuevos problemas e inconsistencias que nos obligaban a replantear ciertas cosas, como en el caso de los récords individuales que hacen parte de un top 10 con los mejores puntajes. La parte de las colaboraciones también fue problemática ya que al hacerlas antes de haber dividido completamente los componentes, tocaba replantearse cada vez que se dividía un componente, lo cual complicó mucho el proceso. Por último, resultó algo confuso la identificación de cuando un componente no tenia que dividirse mas ya que a veces los dividiamos tanto a algunos que había algunas clases que carecían de utilidad y ocupamos mucho tiempo intentando encontrarles una responsabilidad funcional para después darnos cuenta que era mejor dejarlas como atributos de una clase.