Asignatura: Diseño y Programación Orientada a Objetos

Profesor: Germán Romero

Estudiante: Luis Ernesto Tejón **Código: 202113150**

Estudiante: Pablo Pedreros Díaz **Código: 202112491**

Estudiante: Alejandro Hernández **Código: 202111716**

Taller 3

PRIMERA ITERACIÓN:

Rol(es):

2. El único componente en la primera iteración será la aplicación como tal. Sus **responsabilidades** serán:

- La aplicación tiene la responsabilidad de hacer que el juego funcione y todos los elementos del juego están contenidos en la aplicación, por ahora el proyecto solo se descompone en la aplicación.
- La aplicación guarda el top 10 de los mejores puntajes obtenidos desde la creación del juego.
- La aplicación carga el mapa que entra como un archivo al juego que está contenido en ella.
- La aplicación se encarga de mover a Pac-man
- La aplicación se encarga de establecer los algoritmos con los que se moverán los fantasmas.
- La aplicación verifica si el juego terminó al quedarse el tablero sin galletas o cuando pac-man se quede sin vidas.
- La aplicación se encarga de correr el juego con todo lo que esto conlleva: movimiento, colisiones y sus resultados, etc.

SEGUNDA ITERACIÓN:

Elementos del dominio y roles:

- Los componentes y sus roles serán:

- **Consola:** Maneja y solicita la información al usuario sobre el tablero y sobre las jugadas del juego. **(Controller).**
- **Juego (clase):** Define el nivel en que se está jugando (el tablero), define también cuántas vidas quedan y el guarda el puntaje **(Information holder).**
- **Record (clase):** Guarda el top 10 de las mejores puntuaciones **(Information holder).**
- **Tablero (clase):** Defina la distribución de casillas en las que se mueven Pacman y los fantasmas y define la distribución de galletas y frutas en estas casillas **(Information holder).**
- **Casilla (clase):** Es el espacio mínimo de juego, puede contener frutas, fantasmas, galletas o a Pacman **(Information holder)**
- **Pacman (clase):** Se mueve por el tablero, come galletas, fantasmas o frutas y sube el puntaje del jugador al comer **(Information holder).**
- **Fantasma (clase):** Sigue a Pacman por el tablero tratando de capturarlo **(Information holder).**
- **Personalidad:** Define el método que un fantasma usa para acercarse a Pacman **(Information holder).**
- **Puntaje:** Cantidad de puntos obtenidos por el jugador al acabar el juego (Este es el único componente que fluye en el sistema, inicia como un contador del puntaje en la clase "Juego" y si está entre los 10 mejores puntajes se mueve a la clase "Récord") **(Information holder).**
- **Galletas:** Le da puntos al jugador cuando Pacman las come **(Information holder).**
- **Frutas:** Le da puntos al jugador cuando Pacman las come y le da un breve periodo de tiempo en el que Pacman puede comer a los fantasmas **(Information holder).**
- **Vida:** Si Pacman se queda sin vidas entonces el juego acaba **(Information holder).**

TERCERA ITERACIÓN:

- En la tercera iteración simplificamos varias clases que pueden tratarse simplemente como atributos dentro de otras clases y cambiamos la clase Juego para que soporte la lógica principal del programa. Además, cambiamos la clase fantasma a una clase abstracta para crear diferentes fantasmas con un método diferente para moverse y agregamos más específicamente qué funciones cumple cada componente o rol.

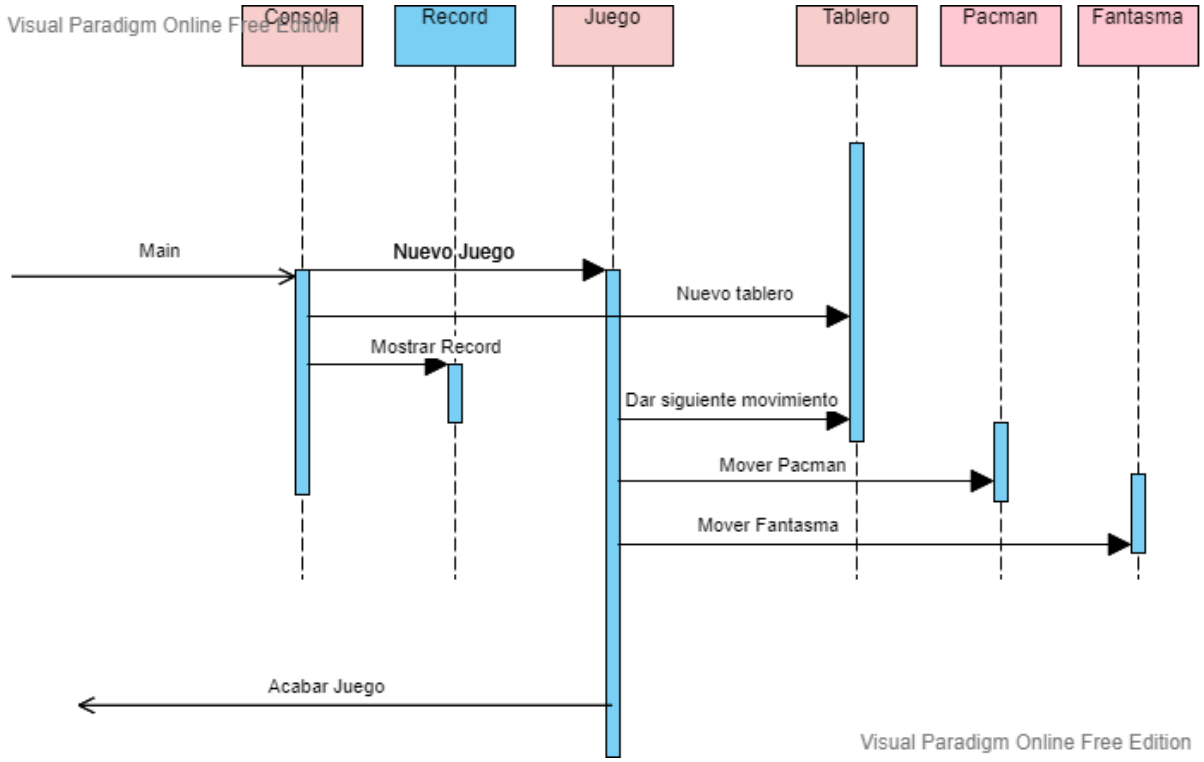
Elementos del dominio y roles:

- Los componentes y sus roles, especificando cómo colaboran entre ellos serán:

- **Consola:** Tiene el método que ejecuta la aplicación. Maneja y solicita la información al usuario sobre el tablero y sobre las jugadas que seguirán en el juego. Almacena la información de los récords de puntajes y es la responsable de crear nuevas partidas. Empezará pidiendo un tablero, y luego pedirá jugadas constantemente hasta que el juego le indique que se acabó el tablero (en ese caso, pedirá otro) o que se acabaron las vidas (en ese caso mostrará los records y preguntará si empezar un nuevo juego). **(Coordinator).**
- **Juego:** Define el nivel en que se está jugando (el tablero). Contiene información de Pacman y de todos los fantasmas en el tablero, además del puntaje del juego y de las vidas restantes. Contiene el método avanzarJuego, que modificará la información necesaria de los information holders con cada jugada (cambio en la posición de Pacman y los fantasmas, indicar qué casillas dejan de tener galletas o frutas, aumentar el puntaje o reducir vida, etc). Este será el método más importante de la aplicación, pues modificará todos los elementos del juego dependiendo de si Pacman se mueve a una casilla con fruta, con galleta, con un fantasma mientras es invencible, mientras no lo es, etc. Además, al final de cada llamado del método, comprobará si quedan vidas restantes o galletas restantes en el juego para notificar a la consola en caso de que tenga que pedir un nuevo tablero o terminar el juego. **(Controller).**
- **Record:** Guarda el top 10 de las mejores puntuaciones en un archivo de texto para que no se borre con las ejecuciones de la aplicación. Al iniciar la aplicación debe leer el archivo de texto para crear el top 10 y al acabar un juego debe actualizarlo si es necesario. **(Information holder).**
- **Tablero:** Define la distribución de casillas del juego, contendrá una matriz de casillas. **(Structurer).**
- **Casilla:** Es el espacio mínimo de juego, puede contener frutas, galletas o ninguna. **(Information holder)**
- **Pacman:** Posee información de su posición actual y la cambia según la jugada diga. También tiene información de si está en estado invencible o no. **(Information holder).**
- **Fantasma (Clase Abstracta):** Es la plantilla sobre la que se crearán fantasmas de diferentes colores. Contiene información de la posición (fila y columna) del fantasma y si está vivo o no. Contendrá un método abstracto que indicará bajo qué algoritmo se

moverá el fantasma, este será el método que cambiará dependiendo del color del fantasma. **(Information holder).**

Diagrama de secuencia de los roles:



REFLEXIONES Y TRADEOFFS:

1. El programa diseñado es una unión entre centralizado y delegado pues el programa se concentra en el controller "Juego" el cual prácticamente solo usa las demás clases para recibir información (information holders) por lo que la mayoría de la lógica se agrupará en esta clase, esto tiene como trade off que la lógica será densa y compleja aunque pero como ventaja es más fácil encontrar los puntos críticos del sistema. Adicionalmente la clase "Consola" es coordinadora esto facilita trabajar con los inputs de parte del jugador y le delega acciones necesarias al controller juego.
2. Para facilitar el agregación de más fantasmas en el juego se usó la clase fantasma como un una clase abstracta la cual no tiene definida la función de dar_siguiente_movimiento() así aunque esto implica programar varias veces la función

para cada fantasma esto facilita crear diferentes fantasmas diferentes “personalidades” al momento de perseguir a Pacman.

3. Del modo en el que está implementado el top 10 de puntuaciones en el UML inicial tenía como problema que el top 10 se perdería cada vez que se cerrará el juego, naturalmente se tuvo que modificar esto con respecto al uml inicial agregando un método de guardar la información del ranking en un archivo txt.

DIAGRAMA UML DE LA ITERACIÓN FINAL: