Taller 3: El Proceso de Diseño

Nicolás Díaz Montaña
Carlos Medina Cardozo
Eliana Palacio Pinzón

Primera Versión

Parte 1: Objetos y Roles: que participan en la solución

Pacman: persigue a los fantasmas, puede ser un controller o service provider.

Fantasma: su objetivo es matar a Pacman, pueden ser un controller o service provider.

Tablero: recuadro donde se mueve el mundo, este puede cambiar dependiendo del archivo ingresado por el usuario, y es un structurer.

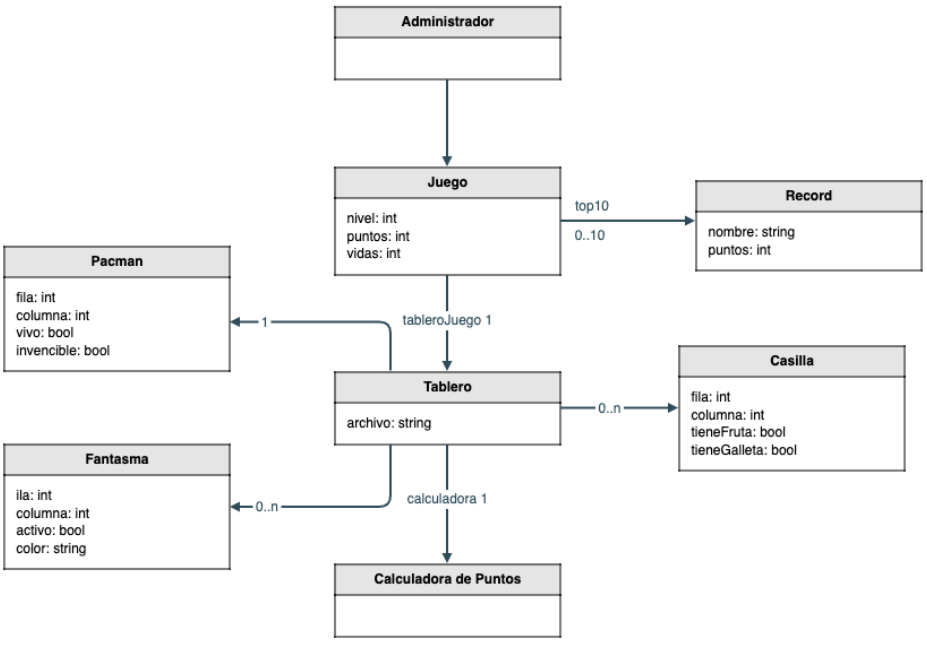
Casilla: lugar donde se ubica alguno de los objetos clave del juego (Pacman, Fantasmas, Galletas, Frutas), es un information holder.

Récord: recibe la información del puntaje del usuario y le deja guardar bajo su nombre ese puntaje, además es un information holder.

Juego: Es donde se tiene registro y control de la información clave del juego como lo son: el número de vidas, el número de puntos y el nivel que se esté cursando, además es un controller.

Administrador: se encarga del manejo del juego, es un Controller.

Calculadora de puntos: calcula los puntos, es un service provider.



Parte 2: Responsabilidades

Abrir los archivos de texto
Almacenar información clave del juego
Sumar los puntos adquiridos por Pacman
Tener la información de las casillas
Mantener la información de puntaje de un usuario
Crear nuevo juego
Abrir y cerrar juego
Matar a Pacman
Recolectar puntos

Parte 3: Colaboraciones

Se pudieron evidenciar (de momento) solo dos colaboraciones entre los componentes de la solución:

1. Cuando se genera un nuevo juego.
2. Cuando se recolecta la información de un juego en proceso.

Segunda Versión

Parte 1: Objetos y Roles: que participan en la solución

• Juego <<Structurer>>: representa el estado actual del juego y además agrupa al resto de elementos que hacen parte de un juego. La cantidad de puntos es la que haya acumulado durante un juego completo, no durante un tablero. La cantidad de vidas es la cantidad de vidas que le restan al jugador. Se puede actualizar la dificultad y las reglas del juego.

• Pacman <<Interfacer>>: es el protagonista del juego, siempre debe estar en una casilla, mientras esté vivo, si recibe “power-ups” puede ser capaz de “comerse” a los fantasmas, y ayuda al usuario a ganar puntos al “comerse” las galletas y frutas.

• Fantasma <<Interfacer>>: son los antagonistas del juego quienes 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. Si Pacman tiene un “power-up” son vulnerables hasta que sean “comidos” una vez. Se pueden crear nuevos fantasmas si el usuario lo desea.

• Tablero y Casilla <<Service Provider>>: el tablero tiene un conjunto de casillas, organizadas en filas y columnas. Cada tablero se carga de un archivo. Cada casilla puede estar vacía, puede tener una fruta o una galleta, pero no ambas al mismo tiempo. Por las casillas pacman y los fantasmas pueden realizar su recorrido.

• Record <<Information Holder>>: recibe la información del puntaje que obtuvo un usuario en su juego y le deja guardar bajo su nombre ese puntaje, y además es un elemento dentro del Top-10 (el registro de los 10 mejores puntajes en el juego) que está ubicado en la clase Juego.

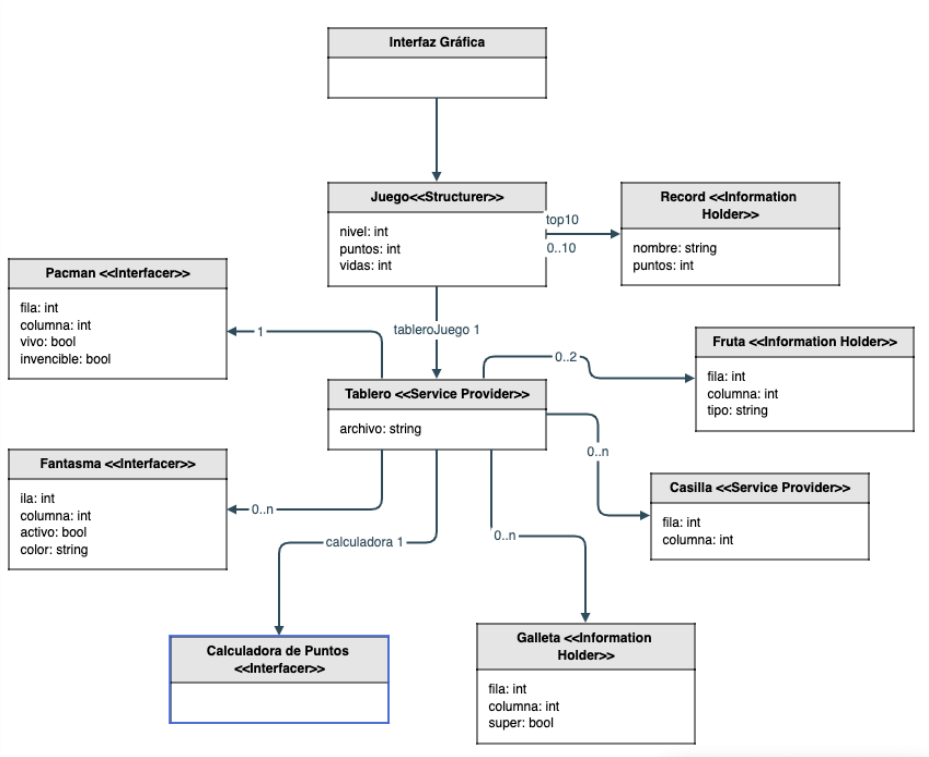
• Calculadora de puntos <<Interfacer>>: calcula los puntos dependiendo de qué objeto/alimento haya “comido” pacman. Cuando los fantasmas estén en este estado vulnerable, los puntos se otorgan según el número de ellos que pacman haya comido seguidos (200, 400, 800 y 1600 puntos, respectivamente).

• Galleta <<Information Holder/Controller>>: son la principal fuente de puntos y deben estar distribuidas a lo largo del tablero. Cada vez que Pacman se “coma” una, se suman 10 puntos al puntaje. Existe una variación, la Super-Galleta, la cual también están distribuidas alrededor del tablero, suman 50 puntos al puntaje, dan el power-up de “comerse” a los fantasmas, pero tienen las restricciones de solo ser 4 por tablero y tener un tiempo de duración determinado.

• Fruta <<Information Holder>>: es otro medio de agregar puntos al puntaje. Puede haber un máximo de dos frutas por nivel. Existen diferentes tipos, cada uno con su respectivo puntaje: cereza(100), fresa(300), naranja(500), manzana(700) y melón(1000).

• Personalidad <<Controller>>: cada fantasma tiene una personalidad diferente y esta determina qué acciones debe tomar el fantasma para atrapar a Pacman. Existen personalidades bases para los 4 fantasmas originales (Blinky, Pinky, Inky y Clyde) que son “Shadow”, “Speedy”, “Bashful” y “Pokey” respectivamente. Se pueden crear otras personalidades si el usuario lo desea para agregarle a nuevos fantasmas.

• Interfaz Gráfica<<>>: ????



Parte 2: Responsabilidades

No	Responsabilidad	Componente
1	Determinar cuándo se acaba el nivel (por medio de la cantidad de vidas)	
2	Verificar y actualizar el Top-10 de Puntajes	
3	Modificar las reglas del juego	
4	Agregar nuevos fantasmas	
5	Inicializar nuevo juego	
6	Mantener información Top-10 entre ejecuciones	
7	Almacenar información del juego actual	
8	Solicitar y cargar archivos de tipo texto	
9	Determinar cuándo se acaba el nivel (por medio de la cantidad de galletas)	
10	Determinar donde están ubicados los objetos del juego	
11	Almacenar información del estado de las casillas	
12	Calcular y sumar los puntos de los objetos que pacman “comió”	
13	Mostrar gráficamente las actualizaciones y movimiento de los objetos del tablero	
14	Determinar el patrón de movimiento de los fantasmas	
15	Establecer nuevo comportamiento de pacman y los fantasmas	
16	Cambiar de Nivel	

Parte 3: Colaboraciones

Hay 4 escenarios principales en los cuales se ve la colaboración entre los componentes de la solución:

1. Cuando los objetos entran en contacto.
2. Cuando se recolecta información.
3. Cuando se crea un nuevo fantasma.
4. Cuando se crea un nuevo juego.

Tercera / Final Versión

Parte 1: Objetos y Roles: que participan en la solución

• Juego <<Structurer>>: representa el estado actual del juego y además agrupa al resto de elementos que hacen parte de un juego. La cantidad de puntos es la que haya acumulado durante un juego completo, no durante un tablero. La cantidad de vidas es la cantidad de vidas que le restan al jugador. Se puede actualizar la dificultad y las reglas del juego. Determina cuando un juego termina.

• Pacman <<Information Holder>>: es el protagonista del juego, siempre debe estar en una casilla, mientras esté vivo, si recibe “power-ups” puede ser capaz de “comerse” a los fantasmas, y ayuda al usuario a ganar puntos al “comerse” las galletas y frutas.

• Fantasma <<Information Holder>>: son los antagonistas del juego quienes 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. Si Pacman tiene un “power-up” son vulnerables hasta que sean “comidos” una vez. Se pueden crear nuevos fantasmas si el usuario lo desea.

• Tablero <<Controller>>: el tablero tiene un conjunto de casillas, organizadas en filas y columnas. Cada tablero se carga de un archivo de texto. Y otorga la información de los objetos que se encuentran en el mismo. Verifica si los objetos entran en contacto. También tiene registro de cantidad de super-galletas dentro del tablero.

• Casilla: <<Information Holder>> Cada casilla puede estar vacía, puede tener una fruta o una galleta, pero no ambas al mismo tiempo. Por las casillas pacman y los fantasmas pueden realizar su recorrido.

• Record <<Information Holder>>: recibe la información del puntaje que obtuvo un usuario en su juego y le deja guardar bajo su nombre ese puntaje, y además es un elemento dentro del Top-10 (el registro de los 10 mejores puntajes en el juego) que está ubicado en la clase Juego.

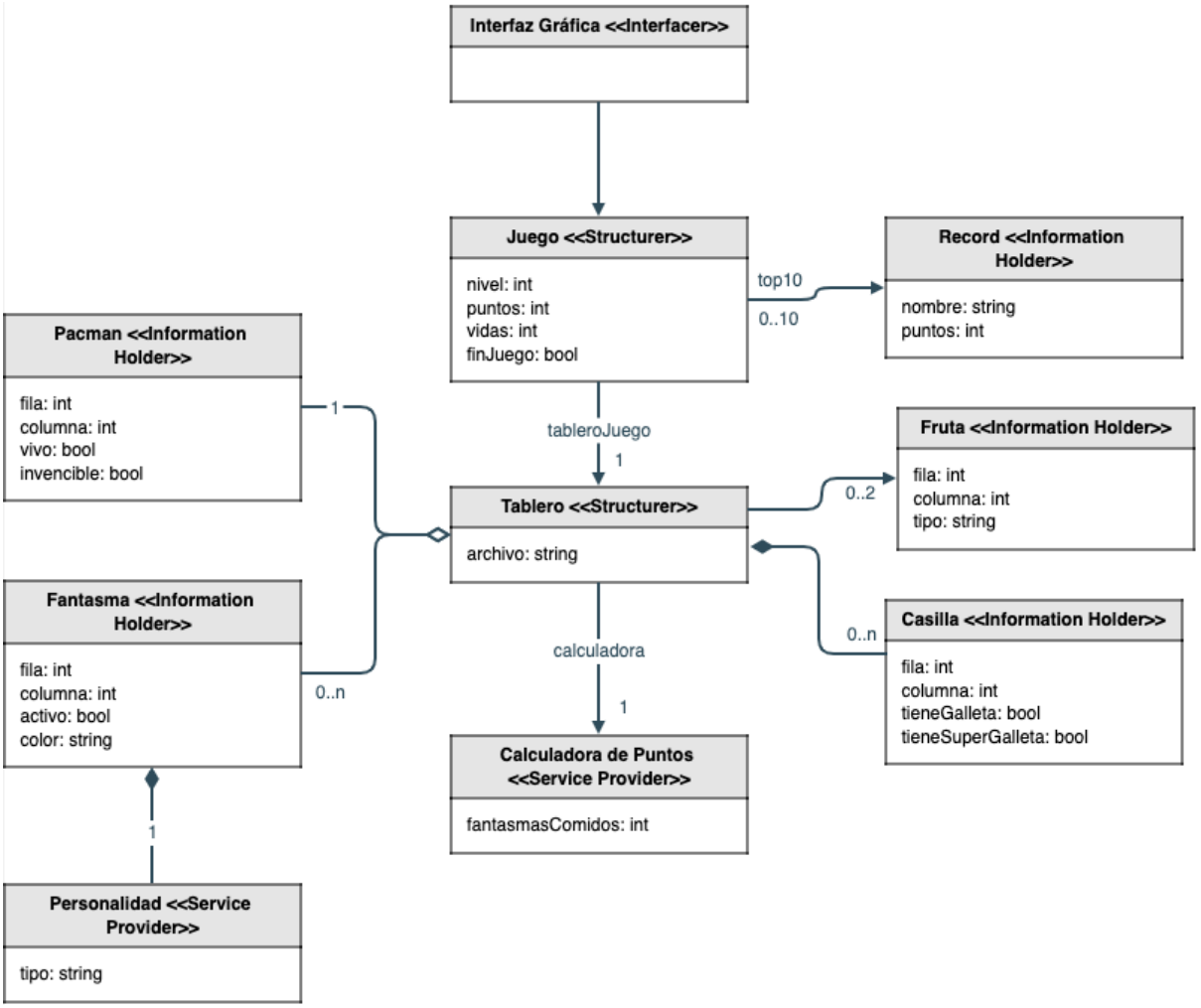
• Calculadora de puntos <<Service Provider>>: calcula los puntos dependiendo de qué objeto/alimento haya “comido” pacman. Cuando los fantasmas estén en este estado vulnerable, los puntos se otorgan según el número de ellos que pacman haya comido seguidos (200, 400, 800 y 1600 puntos, respectivamente). Mientras que las galletas suman 100 puntos y las Super-galletas 500 puntos.

• Fruta <<Information Holder>>: es otro medio de agregar puntos al puntaje. Puede haber un máximo de dos frutas por nivel. Existen diferentes tipos, cada uno con su respectivo puntaje: cereza(100), fresa(300), naranja(500), manzana(700) y melón(1000).

• Personalidad <<Service Provider>>: cada fantasma tiene una personalidad diferente y esta determina qué acciones debe tomar el fantasma para atrapar a Pacman. Existen personalidades bases para los 4 fantasmas originales (Blinky, Pinky, Inky y Clyde) que son

“Shadow”, “Speedy”, “Bashful” y “Pokey” respectivamente. Se pueden crear otras personalidades si el usuario lo desea para agregarle a nuevos fantasmas.

• Interfaz Gráfica <<Interfacer>>: Recibe los comandos registrados por el usuario, es gráfica y será la encargada de mantener el ritmo del juego. Mostrará gráficamente las actualizaciones/movimientos de los objetos.



Parte 2: Responsabilidades

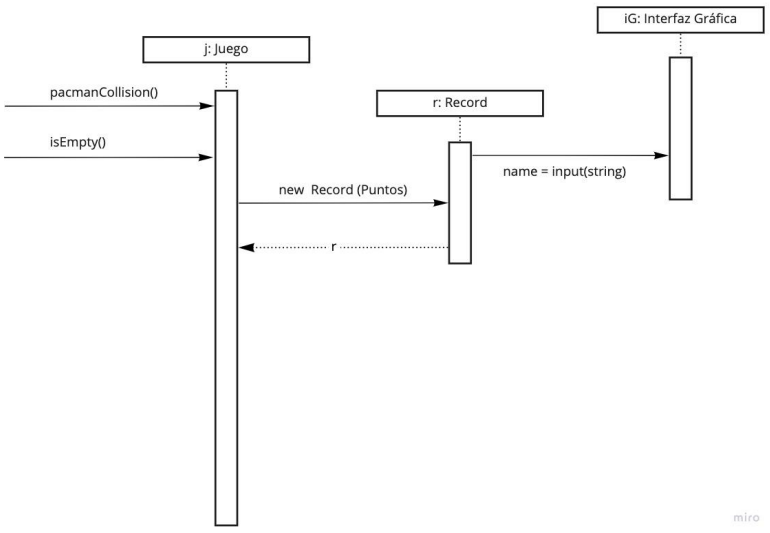
No	Responsabilidad	Componente
1	Determinar cuándo se acaba el nivel	Juego
2	Verificar y actualizar el Top-10 de Puntajes	Juego
3	Modificar las reglas del juego	Juego
4	Agregar nuevos fantasmas	Juego
5	Inicializar nuevo juego	Juego

6	Mantener información Top-10 entre ejecuciones	Juego
7	Almacenar información (vidas, puntos, nivel) del juego actual	Juego
8	Cargar archivos de tipo texto	Tablero
9	Determinar donde están ubicados los objetos del juego	Tablero
10	Almacenar información del estado de las casillas	Casillas
11	Calcular y sumar los puntos de los objetos que pacman “comió”	Calculadora de Puntos
12	Mostrar gráficamente las actualizaciones/movimiento de los objetos del tablero	Interfaz Gráfica
13	Responder a los comandos del usuario para mover a Pacman	Interfaz Gráfica
14	Determinar el patrón de movimiento de los fantasmas	Personalidad
15	Establecer nuevo comportamiento de pacman y los fantasmas	Super-Galleta
16	Cambiar de Nivel	Juego
17	Determinar cuando dos objetos colisionan	Tablero
18	Cerrar el juego	Juego

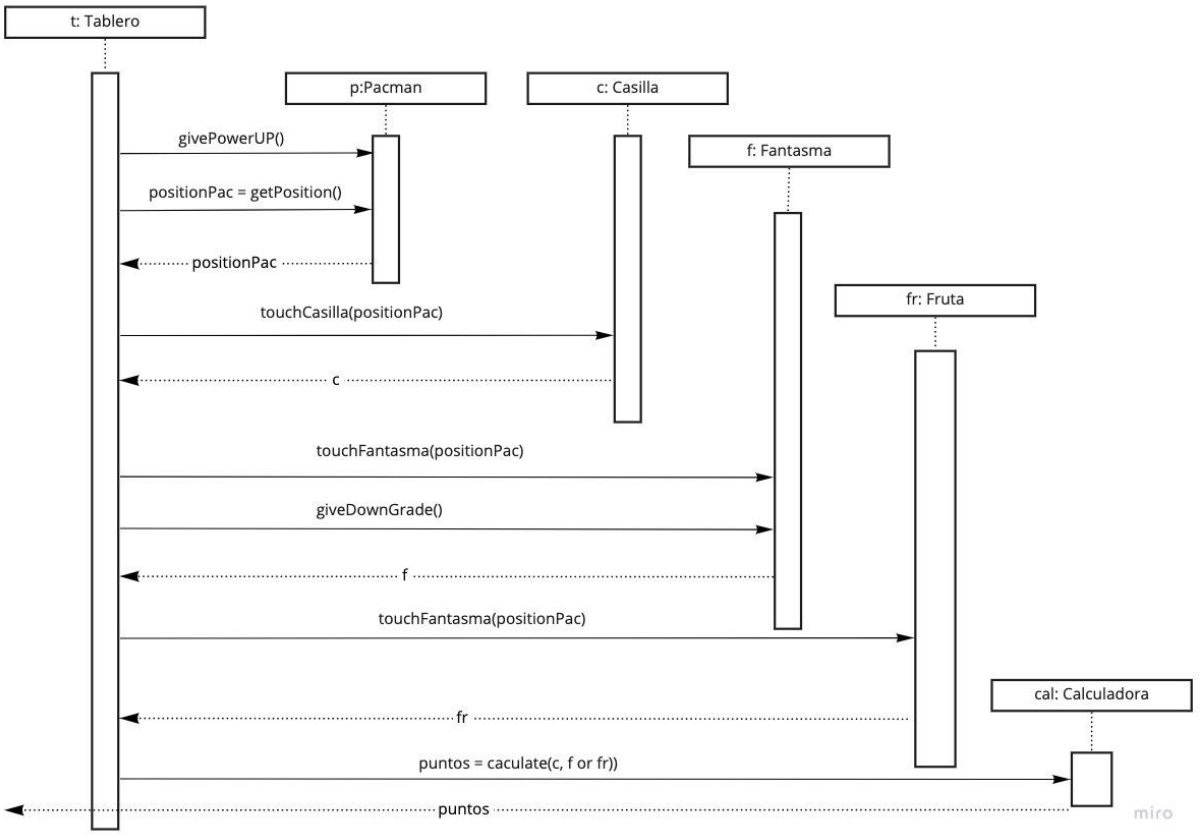
Parte 3: Colaboraciones

Hay 2 escenarios principales en los cuales se ve la colaboración entre los componentes de la solución:

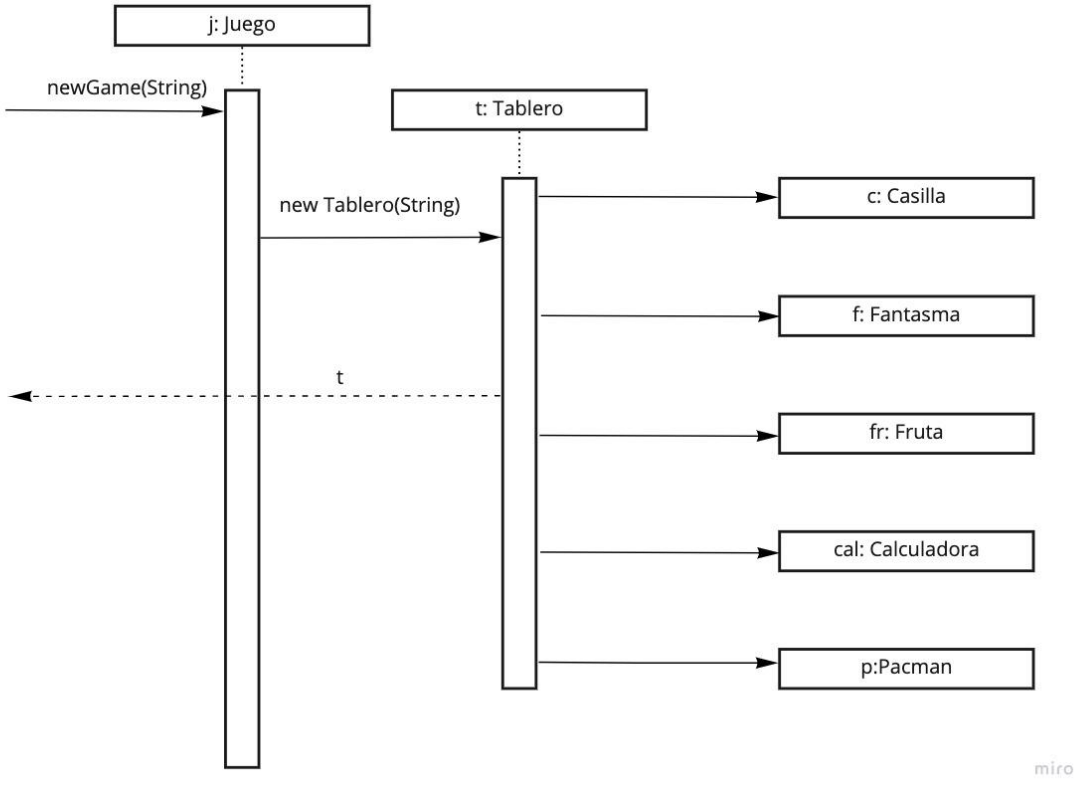
1. Cuando se recolecta información al finalizar un juego.



2. Cuando los objetos entran en contacto.



3. Cuando se crea un nuevo juego.



Parte 4 - Reflexión

La solución de diseño seleccionada no es la única posible, ni se puede decir que sea la más óptima, pero fue la que ofrecía el mejor balance entre todos los requerimientos (funcionales y no funcionales) y las restricciones que se tenían. Por una parte, dentro de las ventajas del diseño propuesto podemos afirmar que cualquier persona que desee implementarlo no tendrá problemas al hacerlo, ya que este es bastante explícito y fácil de leer en la mayoría de los aspectos. Del mismo modo, este diseño está basado en la arquitectura de software MVC (Modelo Vista Controlador), el cual permite separar los datos de su representación visual, facilita el manejo de errores y, sobre todo, permite agregar múltiples representaciones de los datos. Sin embargo, cuenta con desventajas como la cantidad de archivos que se deben mantener y su separación en capas, que aumenta la complejidad del sistema.

Con respecto al proceso de diseño, podemos concluir que fue difícil encontrar una buena solución en el primer intento, por lo cual se realizaron varias iteraciones para llegar a la mejor versión. La técnica implementada que dió mejores resultados fue “divide y vencerás” ya que, descompusimos el proyecto en partes más elementales, lo cual permitió encontrar los factores fundamentales para cumplir con los requerimientos. Para cada componente interno definimos una frontera en términos de lo que hace y cómo se ve, y definimos la interacción entre estos componentes.

Algunas ventajas del proceso iterativo que se puso en práctica fue que se definió una frontera, se descompuso y resolvió cada problema por separado, por lo que fue más sencillo. Adicionalmente fue más fácil repartir el trabajo entre los compañeros del equipo y no era necesario tener que pensar todo el tiempo en todos los detalles. Por otra parte, entre las desventajas que se tuvieron fue que se tuvo que hacer varias iteraciones para resolver todos los problemas del proyecto y eso ocupaba más tiempo. También, diseñar componentes totalmente por separado llevó a diseños diferentes. No eran del todo incompatibles, pero si eran inconsistentes.