**Conceptos**

Elaborar el documento (y subirlo en un archivo en el mismo repositorio) explicando dónde y para qué aplicaron los conceptos del paradigma, justificando las decisiones que tomaron en el modelado. A modo de esquema, proponemos lo siguiente:

* **Polimorfismo**: ¿Cuáles son los mensajes polimórficos? ¿Qué objetos lo implementan? ¿Qué objeto se aprovecha de ello?
* **Colecciones**: ¿Qué operaciones de colecciones se utilizan? ¿Usaron mensajes con y sin efecto? ¿Para qué?
* **Clases**: ¿Usan clases? ¿Por qué? ¿Dónde o cuándo se instancian los objetos?
* **Herencia**: ¿Entre quiénes y por qué? ¿Por qué no? ¿Qué comportamiento es común y cuál distinto?
* **Delegación de responsabilidades**: ¿Qué objetos interactúan? ¿Dónde se delega? ¿Qué tan fácil sería modificar o extender?

**Polimorfismo:** El mensaje choque (method choque() ), que lo entienden gran parte de las clases creadas y lo entienden tanto el objeto jugador como el objeto árbitro para colisionar con las paredes e interactuar con los distintos creados a partir de clases funciona de forma polimórfica, ya que puede recibir varios objetos distintos que están en movimiento y estos logran interactuar sin cambiar el mensaje o repetir lógica.

**Colecciones:** Usamos colecciones para poner los objetos que conforman parte de las distintas pantallas.Poniendo en una lista para cada clase de objetos todas las posiciones en las que queríamos hacer el add.Visual().Y finalmente cargarlas mediante un For Each.

Usamos mensajes con efecto sobre todo en el personaje para modificar los estados del personajes a lo largo del nivel como por ejemplo sumarle las estrellas o su movimiento.

**Clases:** Usamos clases para todos los visuales que se usan más de una vez por ejemplo para la creación de paredes es indispensable ya que necesitamos crear muchas para delimitar el movimiento del jugador y de no tener las clases tendríamos que hacer muchos objetos. Además todas las paredes deben tener el mismo comportamiento así el uso de clases fue clave para que estas sean todas iguales.

**Herencia:** La herencia la utilizamos para el árbitro, el cual hereda la clase tarjeta. Esto quiere decir que al colisionar con este, el interactúa como una tarjeta dependiendo la que tenga en la mano, con esto nos ahorramos repetir el código de una tarjeta suelta. Solo hubo que hacer override del método imagen para que tenga una imagen distinta a la de su clase.

**Delegación de Responsabilidades:** Los objetos que más utilizan este concepto son los que manejan la carga de niveles por ejemplo la flecha que selecciona qué nivel cargar contiene un vector con todos los objetos de los niveles que se pueden cargar y dependiendo en qué posición está selecciona un objeto,nivel, y le manda el mensaje cargarNivel() lo cual efectivamente carga el nivel.Por ende añadir nuevos niveles seria tan facil como hacer un objeto y decir donde queremos los visuales en ese nivel dentro del metodo cargarNivel().