FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

[7507/9502] Algoritmos y Programación III
Primer cuatrimestre de 2024

Grupo:	07
Correctora:	Ing. Maia Naftali
Repositorio:	https://shorturl.at/rAThe

Índice

1. Introducción

El presente informe reune la documentación correspondiente a la resolución del segundo trabajo práctico de la materia Algoritmos y Programación III, realizado por los estudiantes: Mateo Castaño, Juan Manuel Gonzalez Segura, Milagros Thome, Manuel Kelly y Ramiro Prem Silberay. El mismo consiste en el desarrollo de un juego multijugador de preguntas y respuestas, utilizando el lenguaje de programacion Java y el diseño orientado a objetos.

2. Supuestos

Los supuestos que hemos adoptado en la resolución del TP fueron los siguientes: decidimos que los jugadores, al momento de jugar, primero respondan y luego elijan el PowerUp a utilizar en el turno correspondiente. Por otro lado, en cuanto a la resolucion de los turnos (fase en la que se asignan los puntos a los jugadores), optamos por que se activen los PowerUps segun el orden que fueron jugados.

3. Diagramas de clase

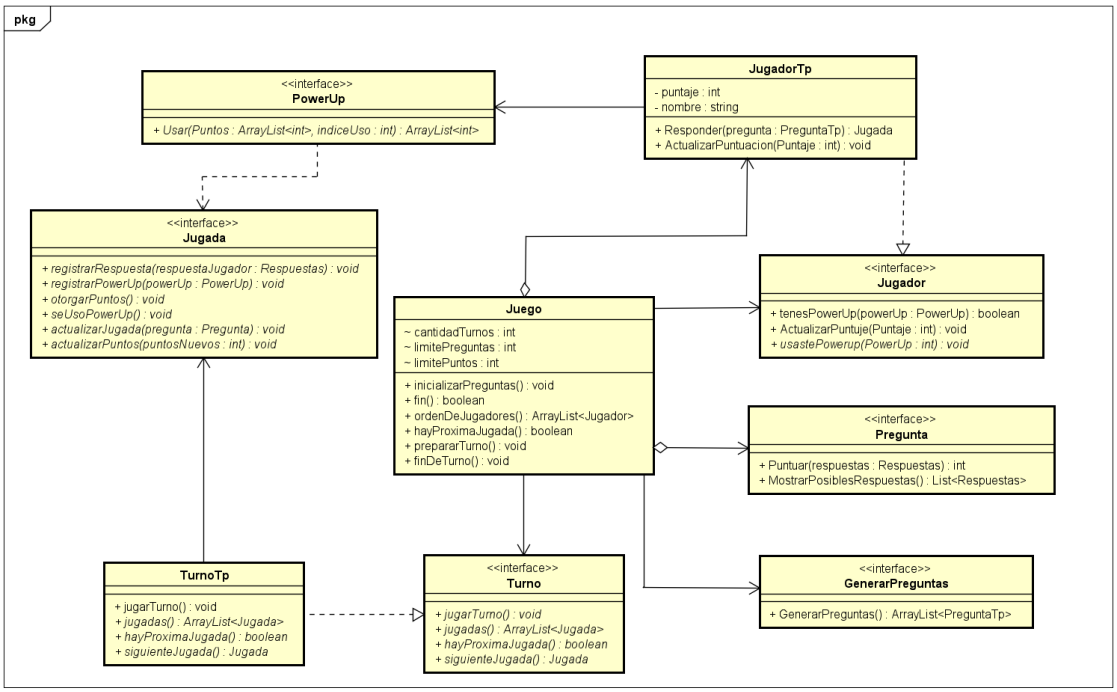

Figura 1: Diagrama de clase principal.

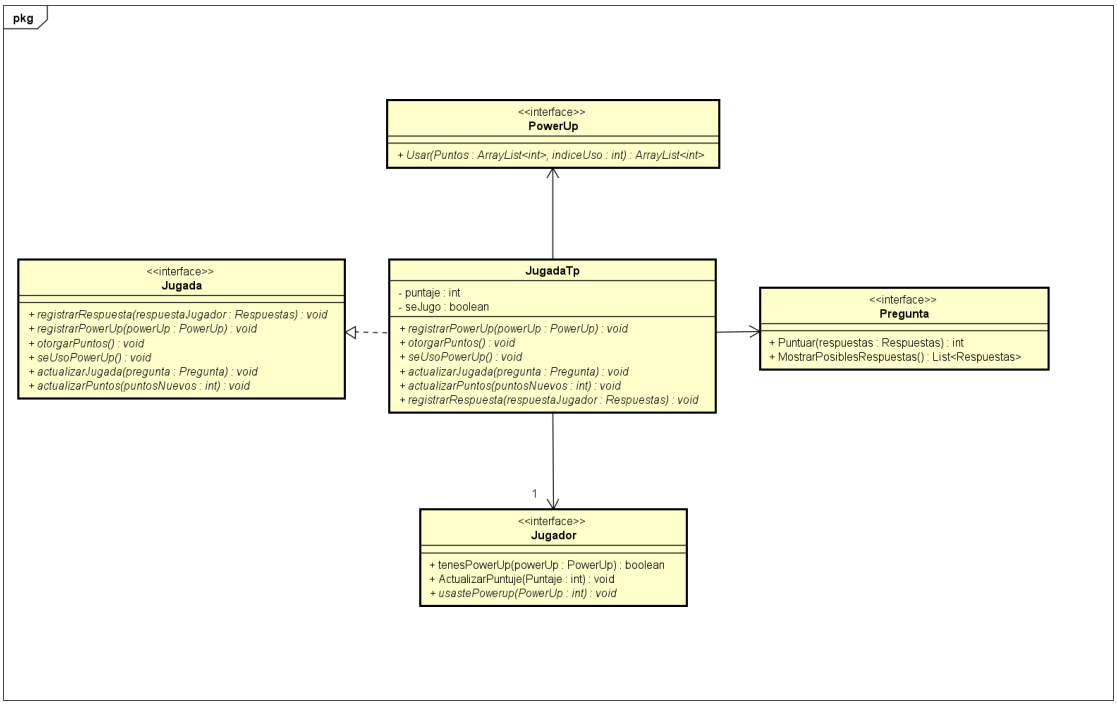

Figura 2: Diagrama de clase de Jugada.

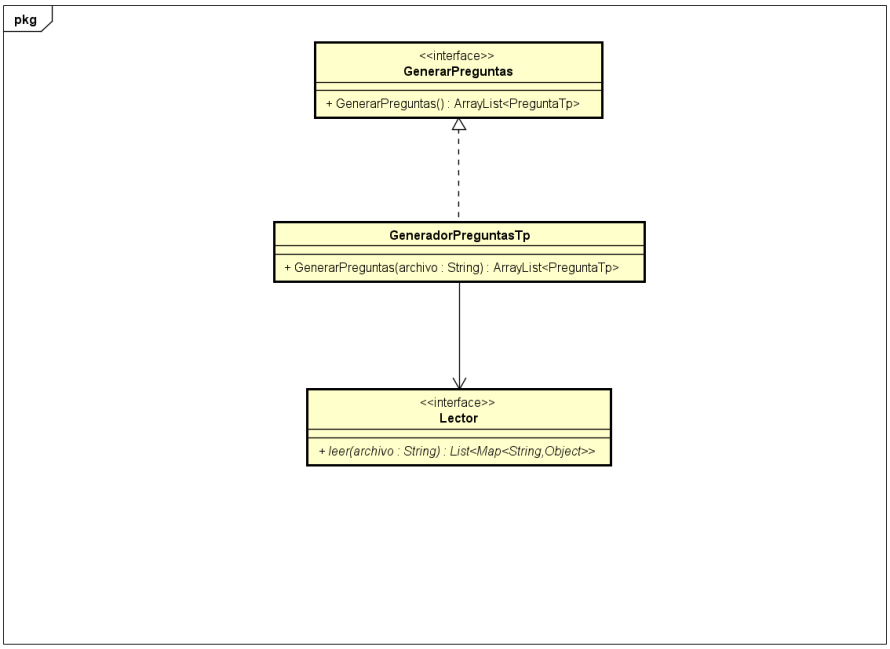

Figura 3: Diagrama de clase del Generador de Preguntas.

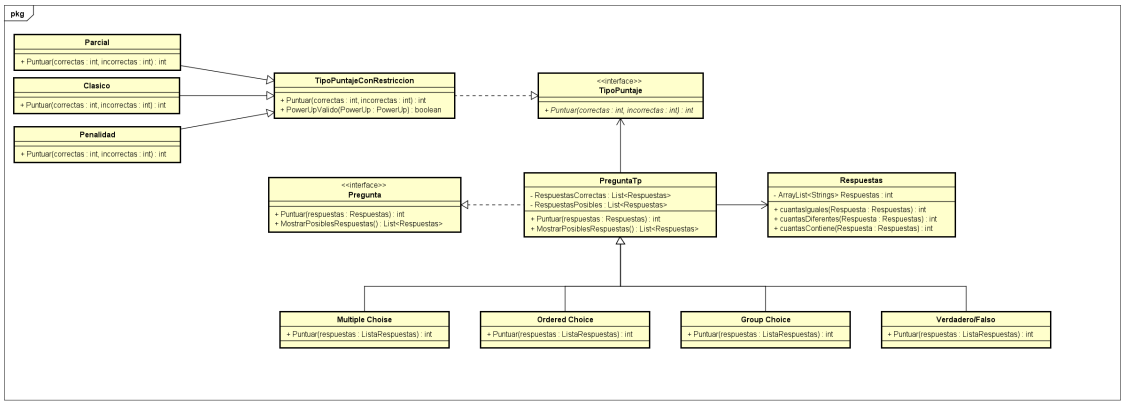

Figura 4: Diagrama de clase de Preguntas.

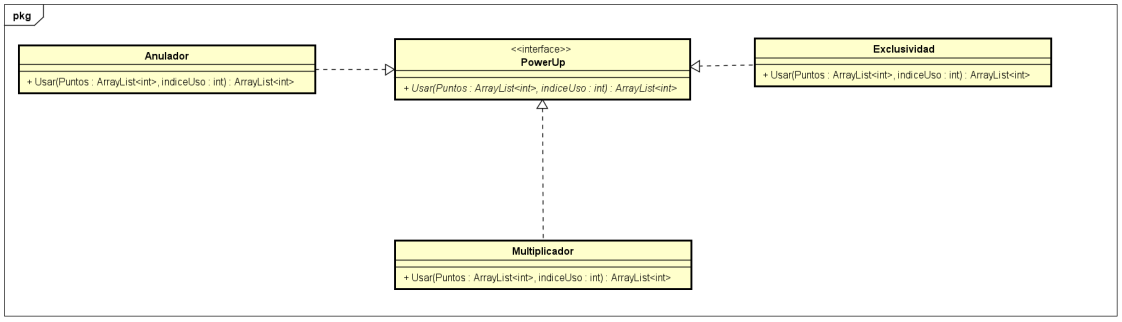

Figura 5: Diagrama de clase de PowerUps.

4. Diagramas de secuencia

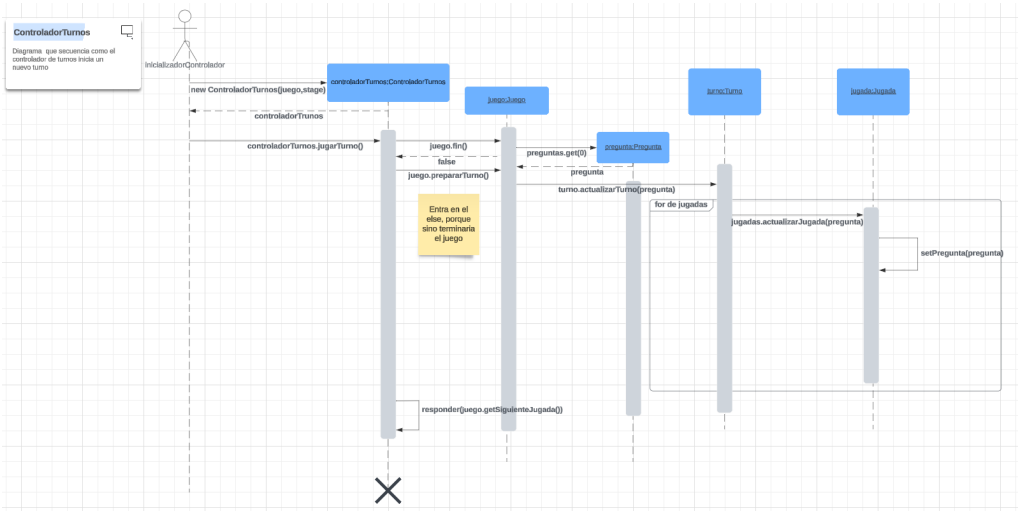

Figura 6: Controlador Turnos

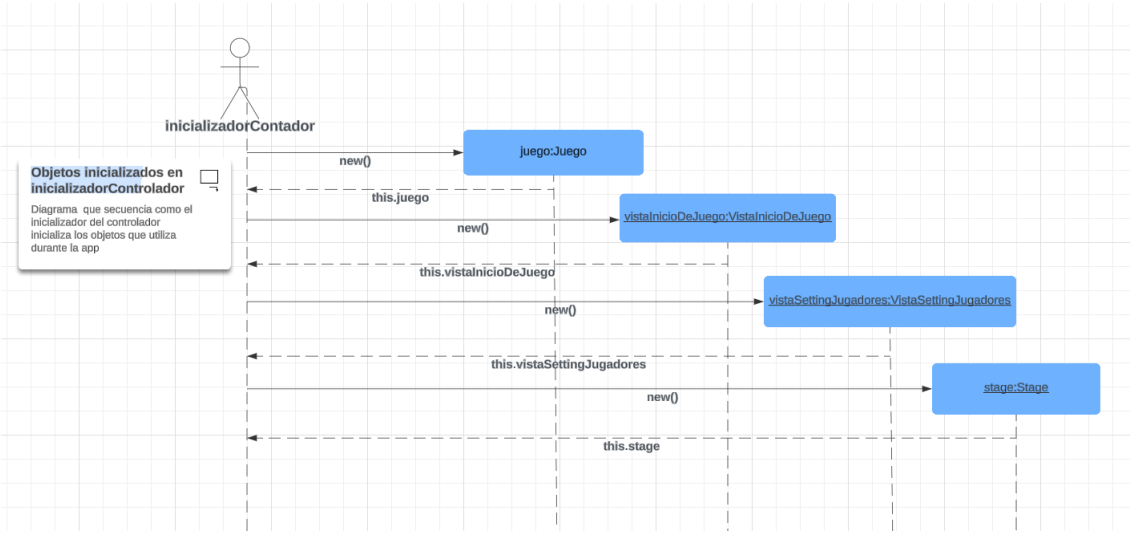

Figura 7: Diagrama de Secuencia de Inicializacion del Controlador.

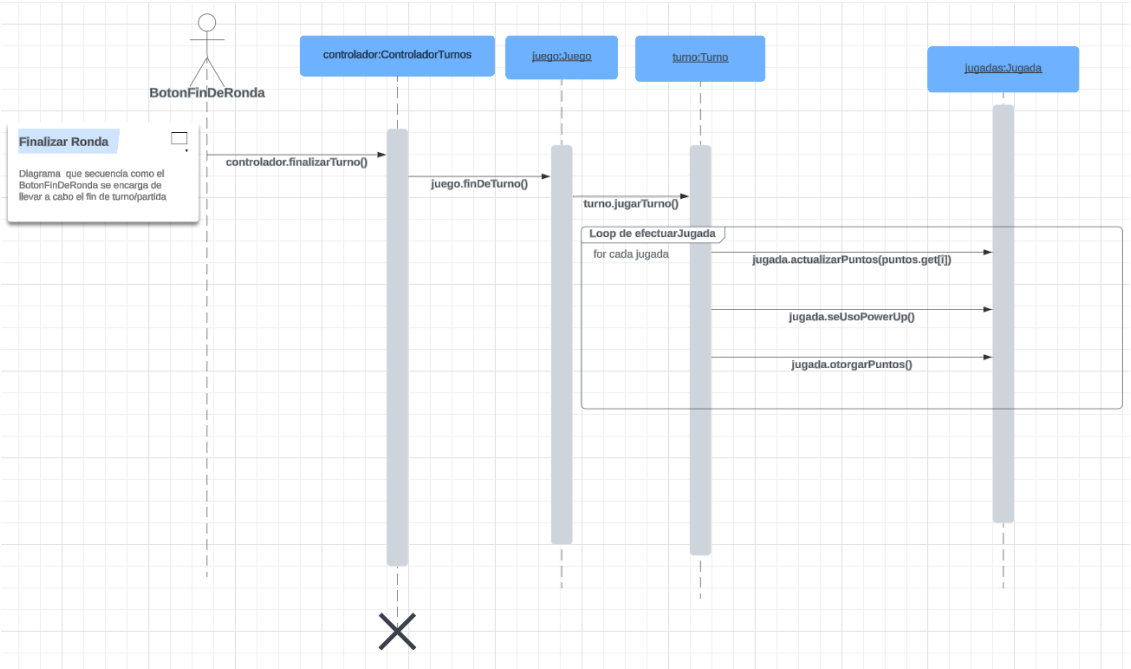

Figura 8: Diagrama de Secuencia Boton De Fin De Rondas siendo que el juego todavia no termina.

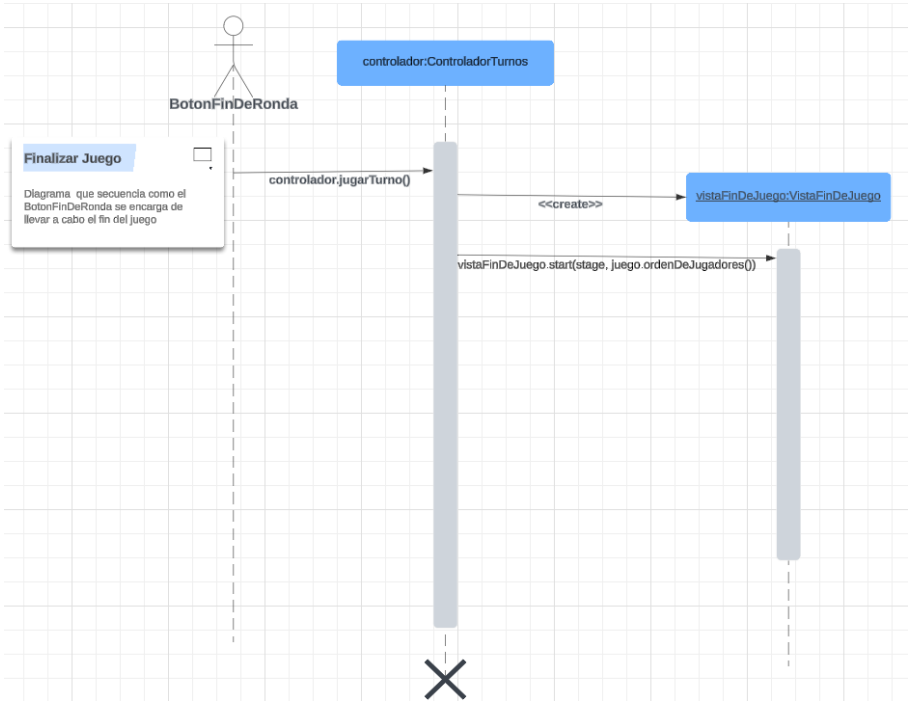

Figura 9: Diagrama de Secuencia Boton Fin De Ronda siendo fin del juego.

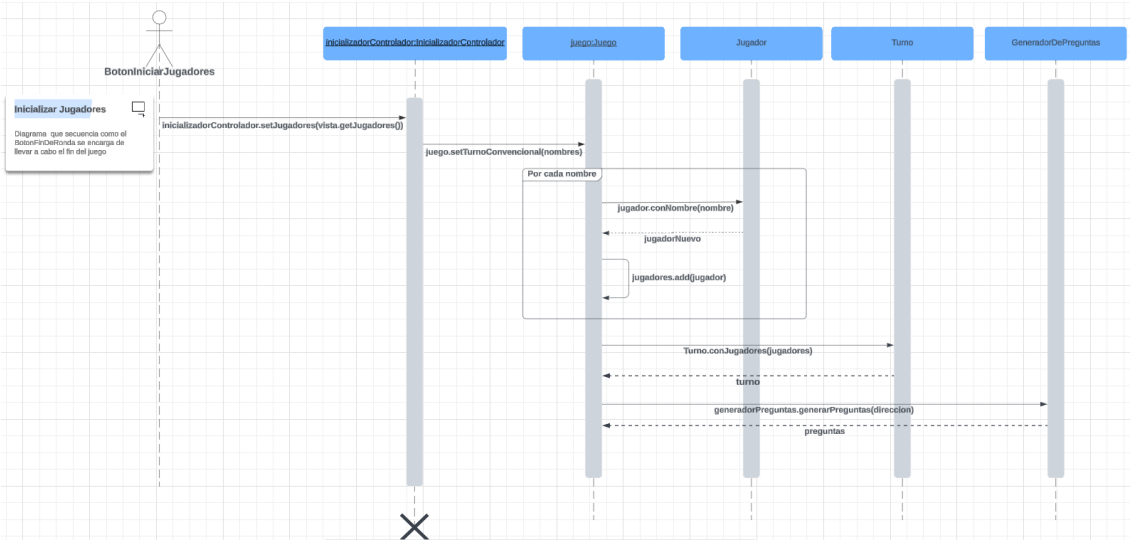

Figura 10: Diagrama de Secuencia de Inicializacion de Jugadores bajo el Controlador

5. Diagramas de paquetes

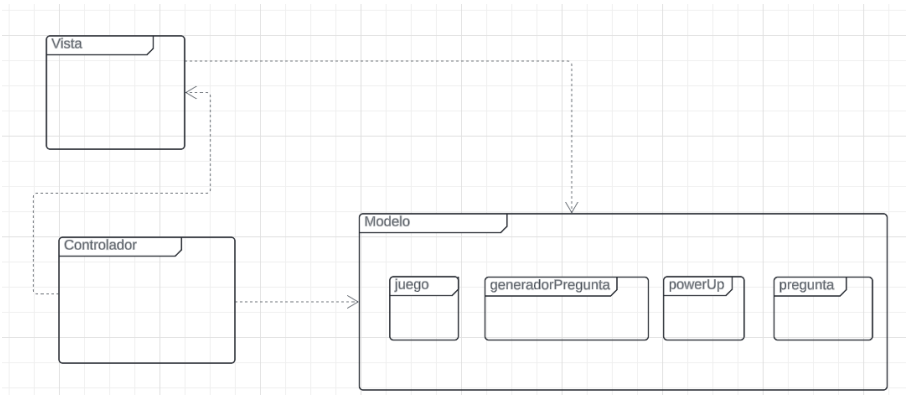

Figura 11: Diagrama de Paquete.

6. Detalles de implementación

Los patrones de diseño que utilizamos fueron:

Modelo-vista-controlador (MVC), patron que se caracteriza por separar los datos/logica del programa (en nuestro caso el modelo dele juego) del modulo que se encarga de gestionar los eventos/comunicaciones (en nuestro seria el apartado visual). Empleando este patron, hemos logrado separar la logica del modelo del de la visual, y que toda transferencia de datos que entre estos sea gestionada por los controladores.

Strategy, patron que determina como se debe realizar el intercambio de mensajes entre diferenntes objetos, permitiendo que el objeto cliente pueda intercambiar dinamicamente de algoritmo a utilizar. Empleamos dicho patron para la gestion de PowerUps, asociadas a la jugada de cada jugador, el cual, en cada turno, debe seleccionar el PowerUp que usara.

Empleamos la herencia en las clases abstractas PreguntaTp, TipoPuntajeConRestriccion y PowerUpTp. En PreguntaTp agrupamos todos los tipos de preguntas, en TipoPuntajeConRestriccion todos los tipos de puntaje y en PowerUpTp todos los tipos de powerup. De esta manera, facilitamos el manejo de los mismos, como al momento de su creacion, de determinar los puntos (tanto en el caso del tipo de puntaje como el powerup), de mostrar la pregunta, etc.

Por otro lado, hemos utilizado interfaces para mejorar la expansibilidad del codigo (Open-Closed), reducir las dependencias tradicionales de arriba abajo (Dependency Inversion) y para poder implementar mock objects, mediante el framework de Mockito, los cuales fueron de gran ayuda al momento de armar los tests.

En cuanto a la delegacion, ha sido fundamental al momento de estructurar las secuencias del modelo. El Juego delega la logia de los turnos a TurnoTp. Este delega la logica de las jugadas de cada jugador a JugadaTp. Luego, TurnoTp, delega la determinacion de los puntos a PreguntaTp, y este se lo delega a TipoPuntajeConRestriccion. Finalmente, TurnoTp delega la logica de la asignacion de puntos a PowerUpTp.

7. Excepciones

Las excepciones que creamos son:

RespuestaInvalida: segun el tipo de pregunta evalua si la respuesta es valida. Se le avisa al jugador y se le pide que vuelva a contestar.

PowerUpInvalido: evalua si el PowerUp ingresado en la jugada es valido. Se le avisa al jugador y se le pide que vuelva a seleccionar PowerUp.

JugadorNoTienePowerUp: evalua si el PowerUp tiene disponible el PowerUp seleccionado. Se le avisa al jugador y se le pide que vuelva a seleccionar PowerUp.

FaltanRespuestasDeJugadores: evalua si, al terminar la ronda, faltan respuestas de jugadores. Se le avisa a los jugadores y se le da la oportunidad de responder a aquellos que faltasen.

NoHayJugador: evalua si se creo un Juego sin jugadores. Se vuelve a pedir los nombres de los jugadores.