# [75.07 / 95.02]

# Algoritmos y programación III

*Trabajo práctico 2: Algo*

Estudiantes:

| Nombre | Padrón | Mail |
|---|---|---|
| Gonzalo Laos | 109977 | llaos@fi.uba.ar |
| Joani Pranteda | 108303 | jpranteda@fi.uba.ar |
| | | |
| | | |
| | | |

Tutor: Pablo Suárez

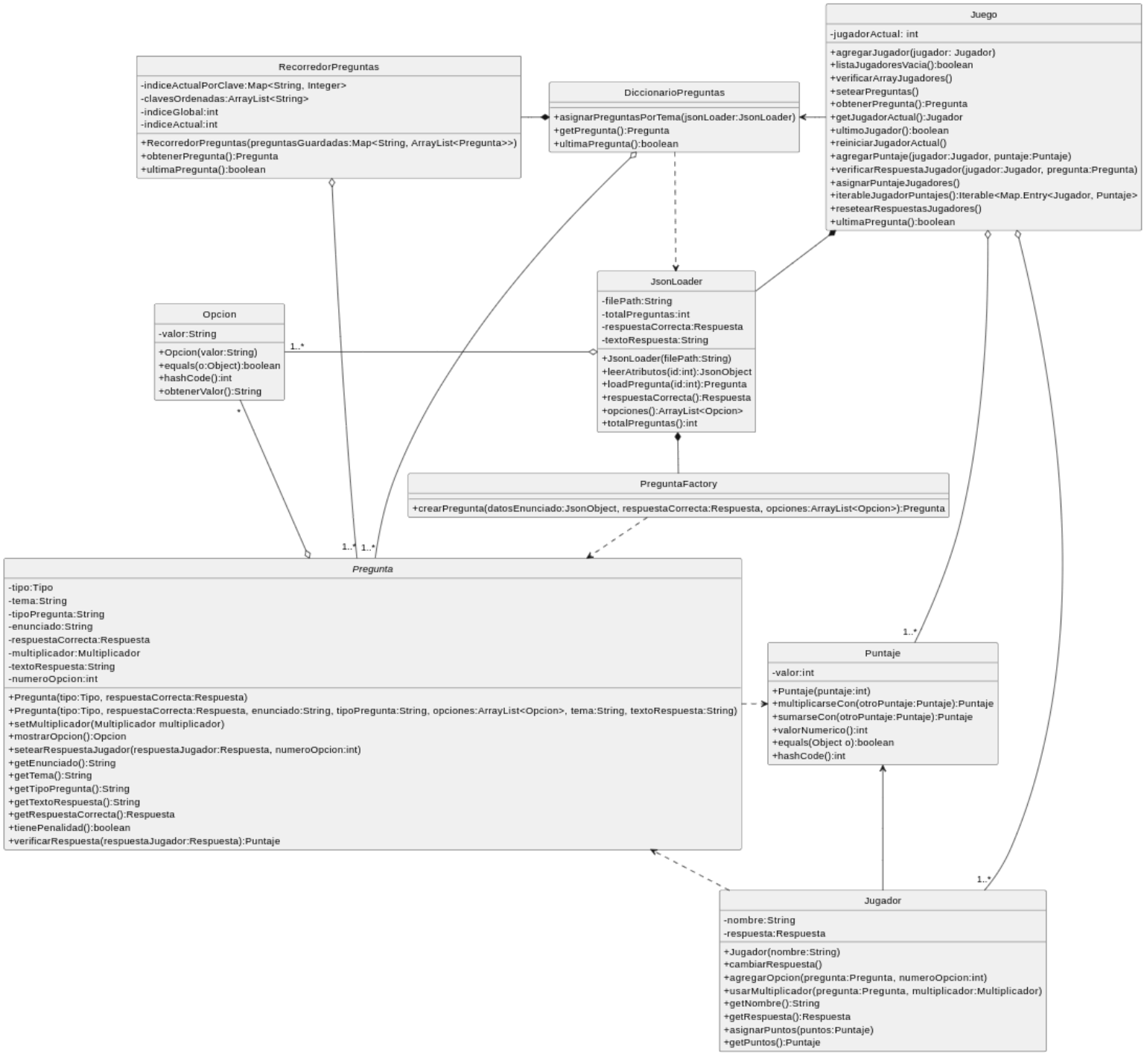
Nota Final:

## Supuestos:

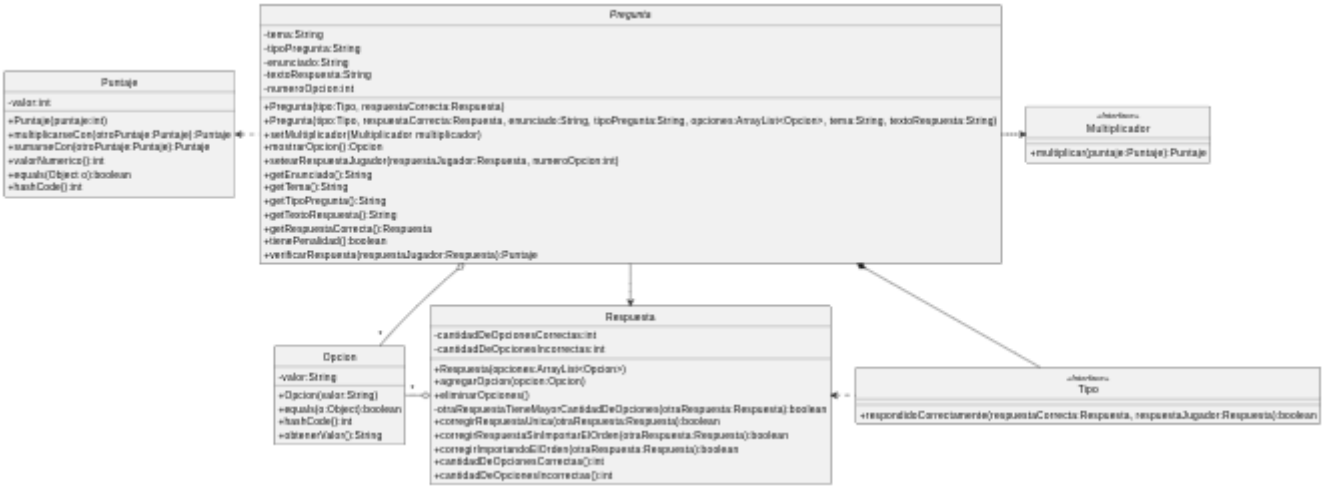
- El jugador una vez seleccionado el **multiplicador** no puede volver atrás pero sí puede seleccionar otra instancia del mismo.
- Durante la ejecución de cada **pregunta** el jugador puede elegir corregir las opciones seleccionadas, eliminando todas las opciones seleccionadas anteriormente.
- Una vez que los **jugadores** contesten una pregunta podrán elegir finalizar el juego.
- Para responder correctamente la pregunta **Ordered Choice** se deberán elegir las opciones en orden correcto.

## Diagrama De Clases:

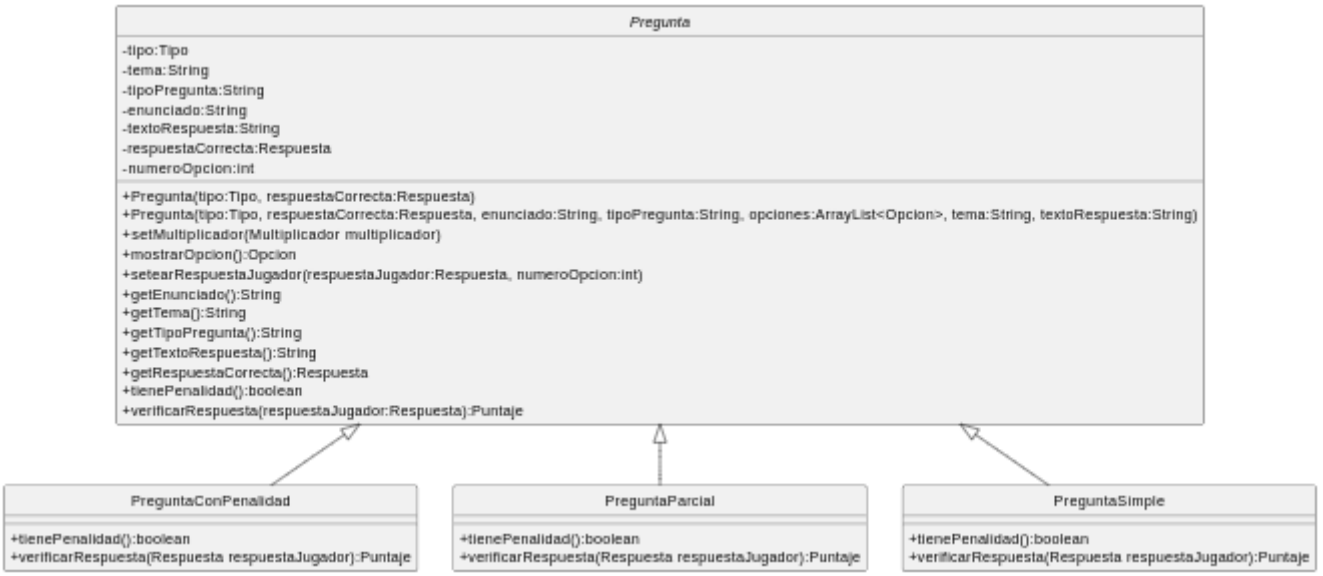
- Diagrama que representa con que clases interactúan todas las clases del juego.



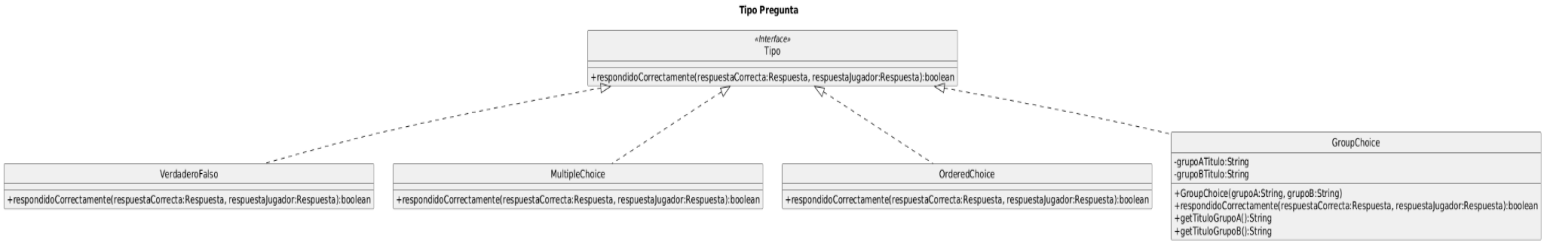
- Clase **Pregunta** en su totalidad:




- Hijos de la clase **Pregunta**:

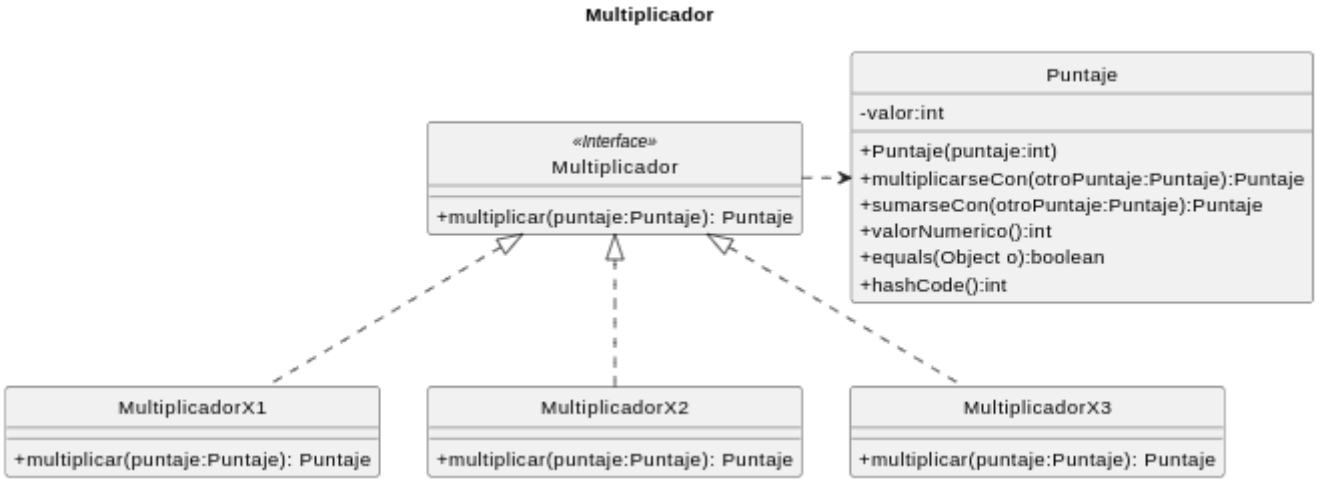



- Clases del **Tipo** de pregunta:



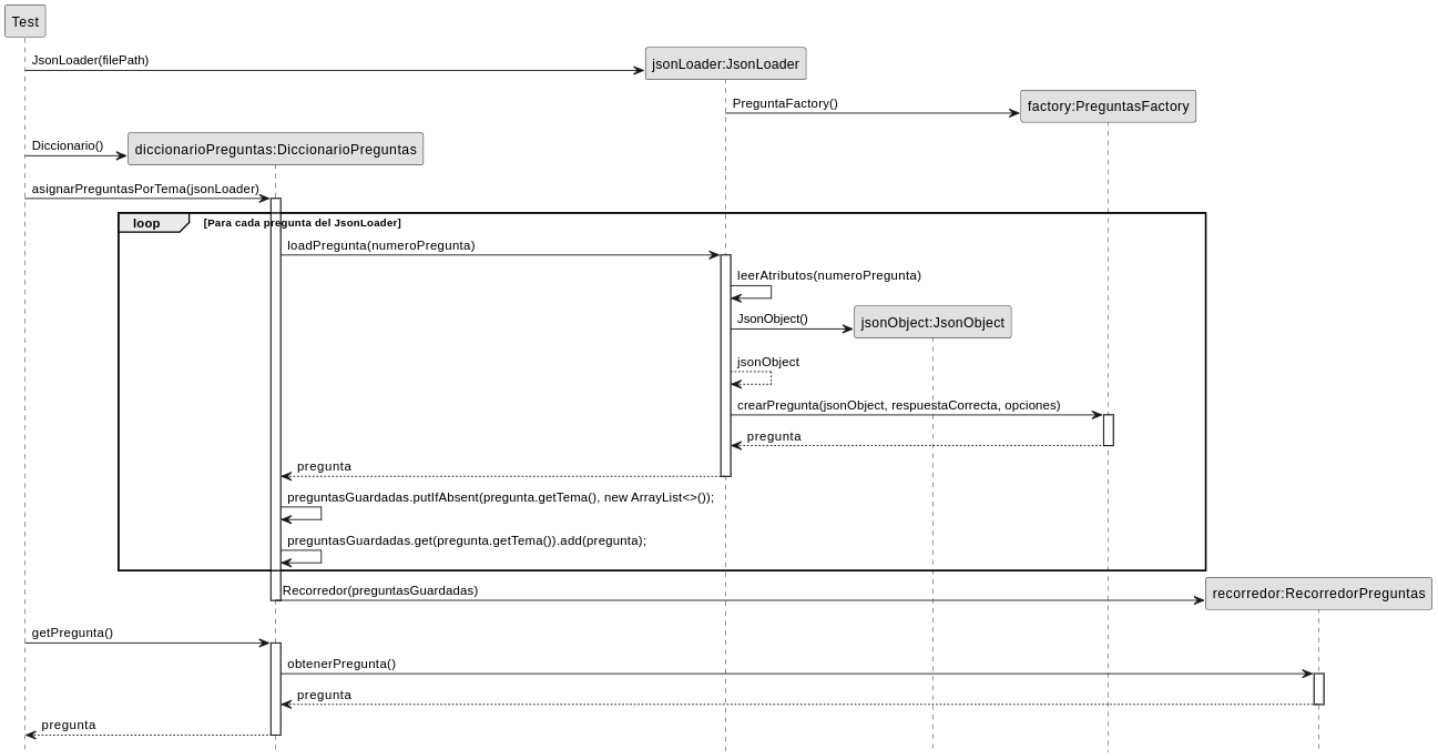
- Clase **jugador** completo:


- Interfaz **Multiplicador**:

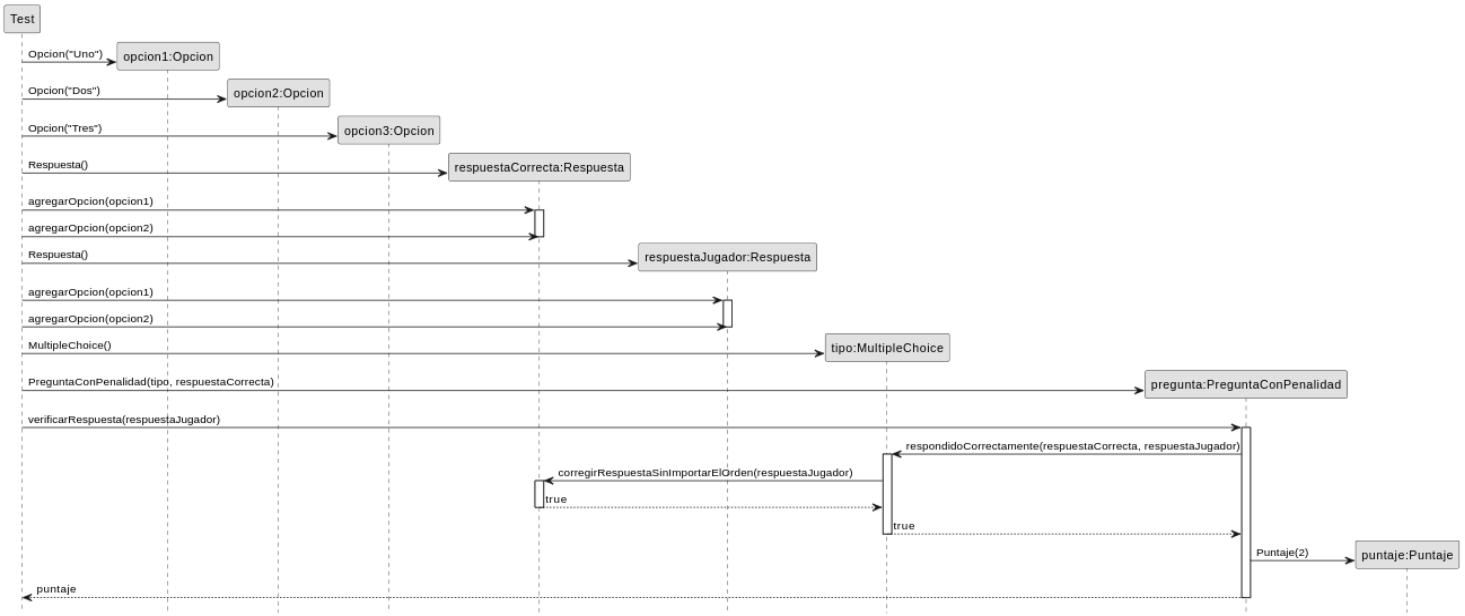


# Diagramas Secuencia:

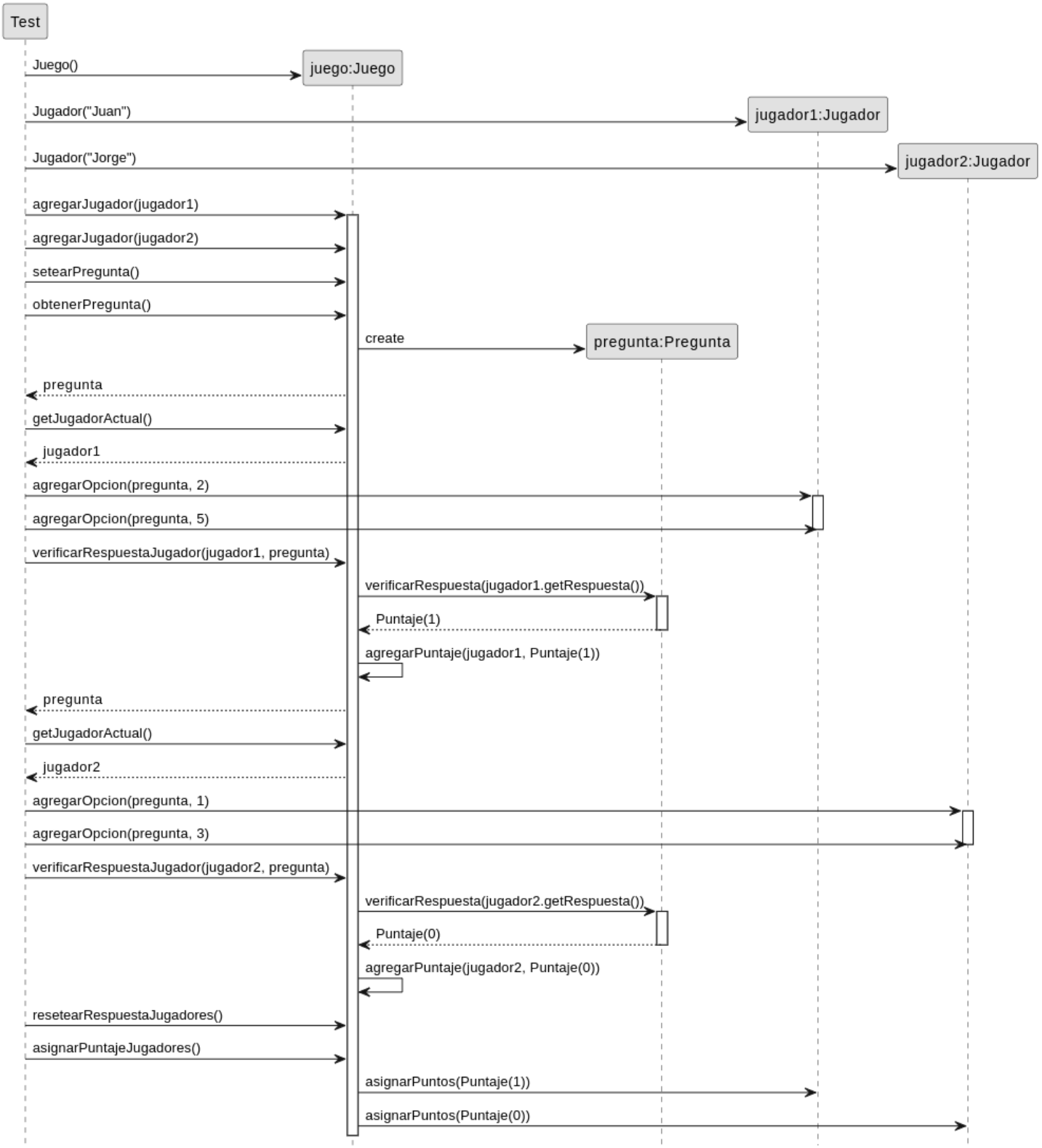
- Inicialización **Pregunta:**




- Verificar **Pregunta**:

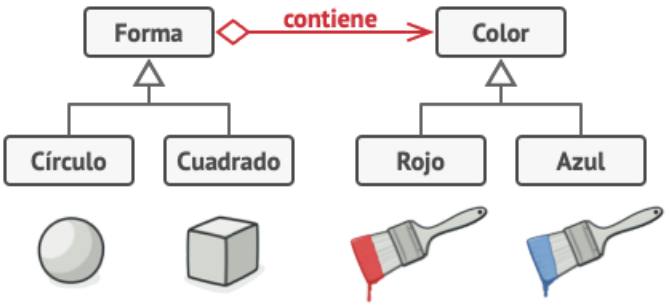


- Dos jugadores contestan una pregunta:



## Detalles de implementación:

Para el diseño de las preguntas hemos decidido utilizar el patrón de diseño Bridge(Puente) que se caracteriza por utilizar esta implementación:




Siendo la **Forma** las **Preguntas** y el Color el **TipoPregunta.**

El juego delega el comportamiento de setear las preguntas al **DiccionarioPreguntas** y este a su vez hace lo mismo delegando la tarea al **JsonLoader** el cual se encarga de leer la pregunta del archivo **.json** y este último en conjunto con el **PreguntaFactory** crean una **Pregunta** la cual se va ir agregando al **DiccionarioPregunta**.

Los puntos más conflictivos fueron el modelaje de las preguntas y respuestas que se fueron resolviendo de esta manera.

La **Pregunta** delega al **Tipo** la manera de corrección entre la  respuesta del jugador y las respuesta correcta.

Las subclases de **Pregunta** se encargarán de calcular el puntaje heredando de la clase padre un método abstracto **verificarRespuesta** en el cual cada subclase calculará el **Puntaje** de distinta manera.

Cada **Tipo** de **Pregunta** va a llamar a un método específico de la respuesta para verificar que la comparación entre las **Respuesta** del jugador y la correcta coincidan.

## Excepción

`ListaJugadoresVaciaException`

Esta excepción se lanza cuando no se ingresa ningún jugador y se quiere comenzar con el juego y la acción a realizar es en la interfaz se muestra una etiqueta con un mensaje de error, y no te deja comenzar con el juego.