Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2020

Alumno	**Padrón**	**Correo electrónico**
Agustin Ariel Andrade	104046	aandrade@fi.uba.ar
Emmanuel Walter Sarzi	104047	esarzi@fi.uba.ar
Gabriel Lorenzo Ortega	104548	gortega@fi.uba.ar
Nahuel Cellini Rotunno	103320	ncellini@fi.uba.ar
Pablo Salvador Dimartino	101231	psdimartino@fi.uba.ar

Índice

1. Introducción

El presente informe reúne la documentación de la solución de la entrega 5 del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un juego de preguntas y respuestas en Java utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

1. El archivo de preguntas en el formato que fuere tendrá un lector capaz de leer las mismas.
2. El archivo tendrá por cada pregunta al menos una opción correcta
3. En caso de que el jugador decida no responder la pregunta, no sumará ni restará puntos
4. En caso de que el jugador no ingrese un nombre, automáticamente el nombre asociado al mismo será una cadena vacía.

3. Diagramas de clase

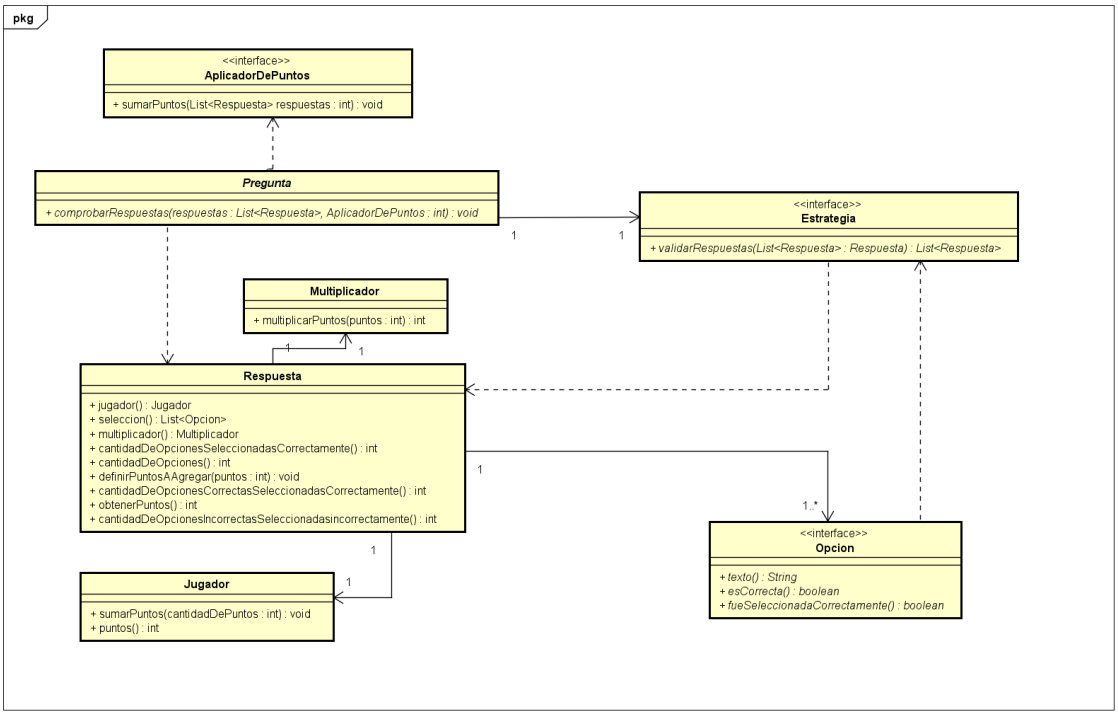

Figura 1: Diagrama de clases general del modelo .

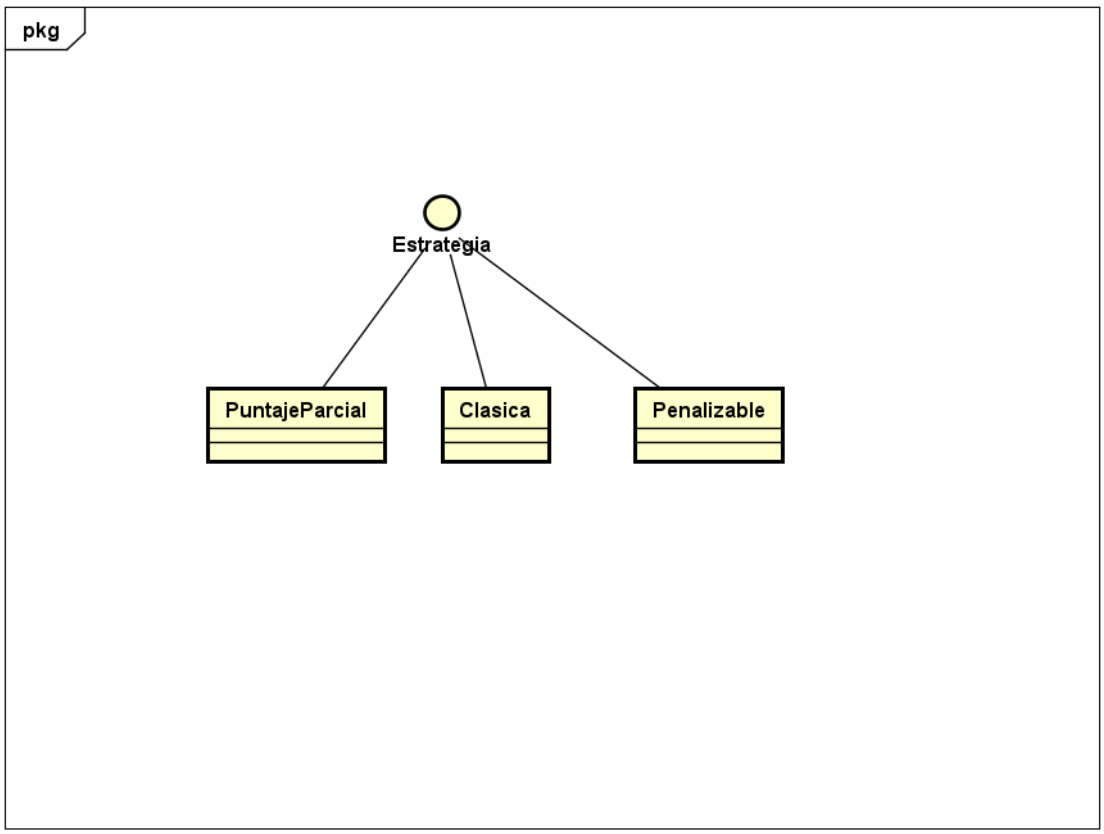

Figura 2: Diagrama de clases provenientes de la interfaz Estrategia.

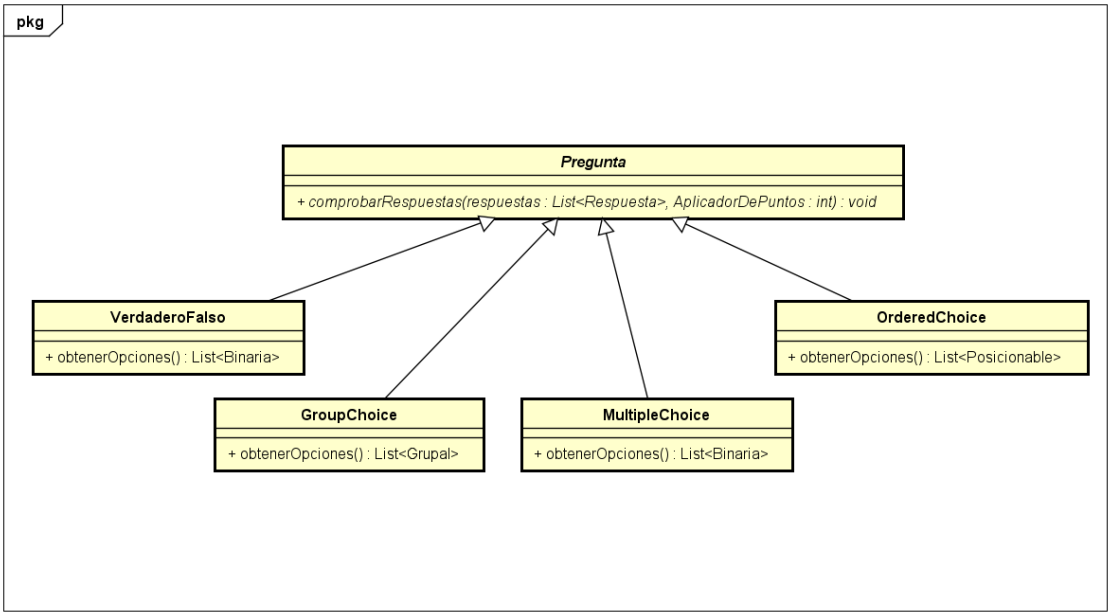

Figura 3: Diagrama de clases provenientes de la clase abstracta Pregunta.

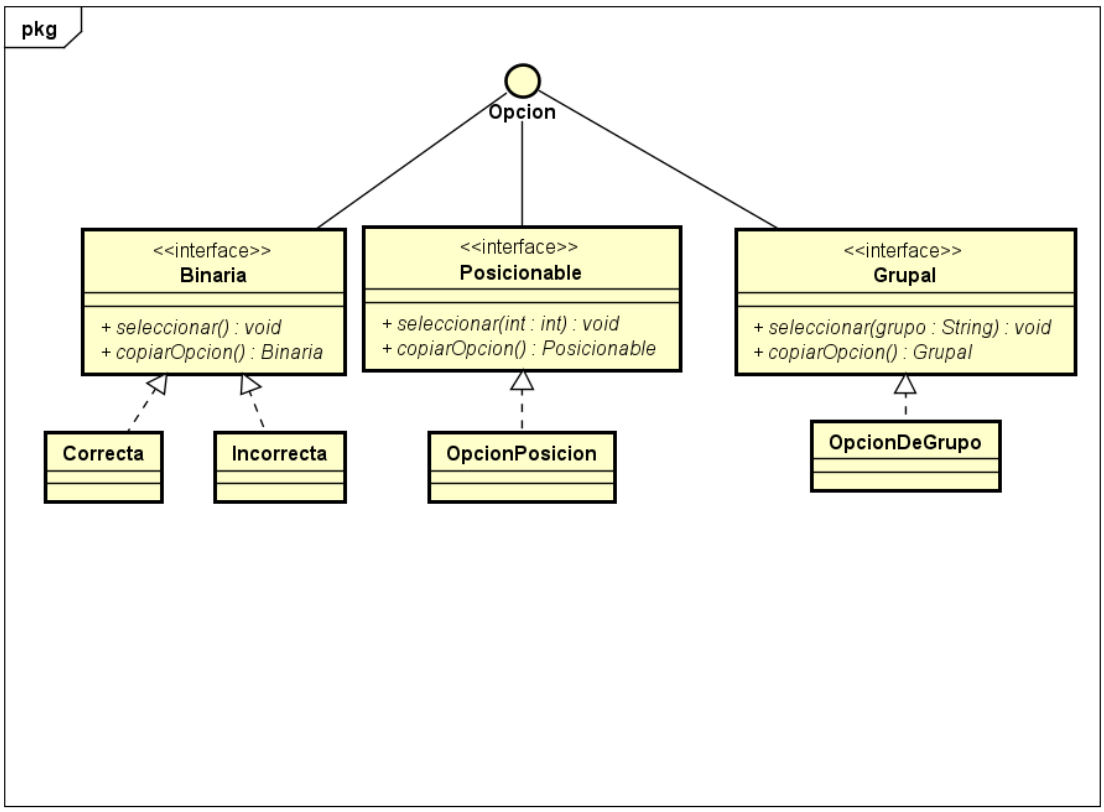

Figura 4: Diagrama de clases provenientes de la interfaz Opcion.

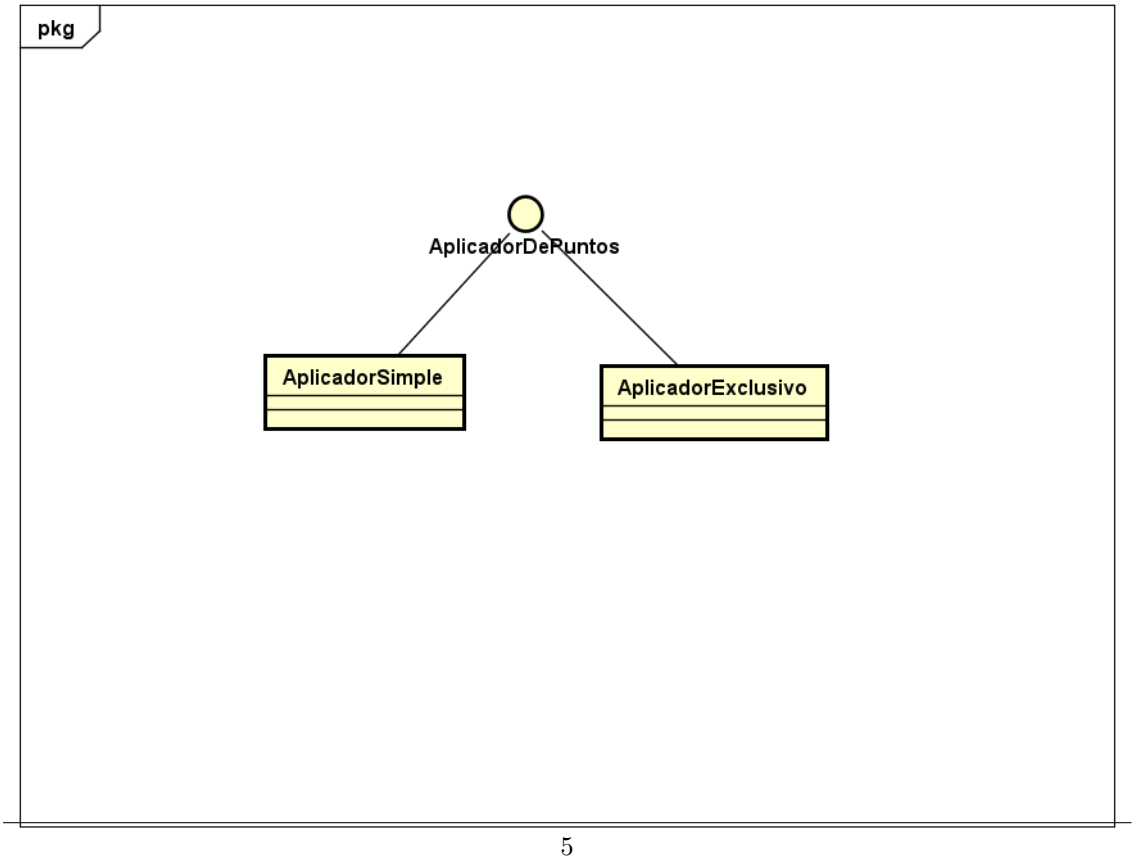

Figura 5: Diagrama de clases provenientes de la interfaz aplicador de puntos.

4. Diagramas de secuencia

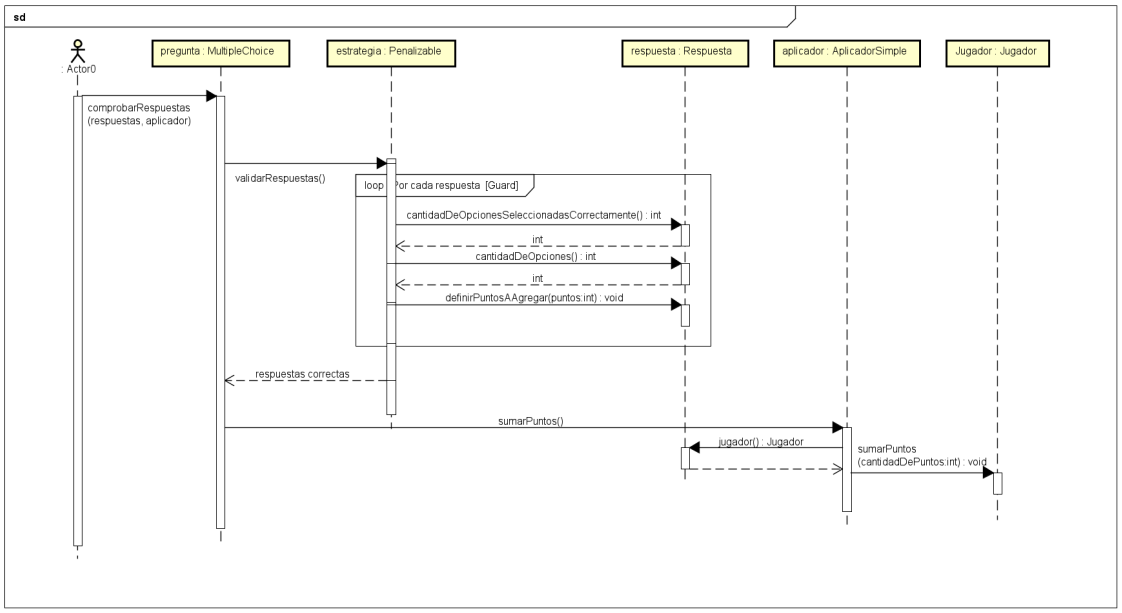

Figura 6: Pregunta Multiple Choice Penalizable recibe una lista de respuestas y puntúa para una selección correcta.

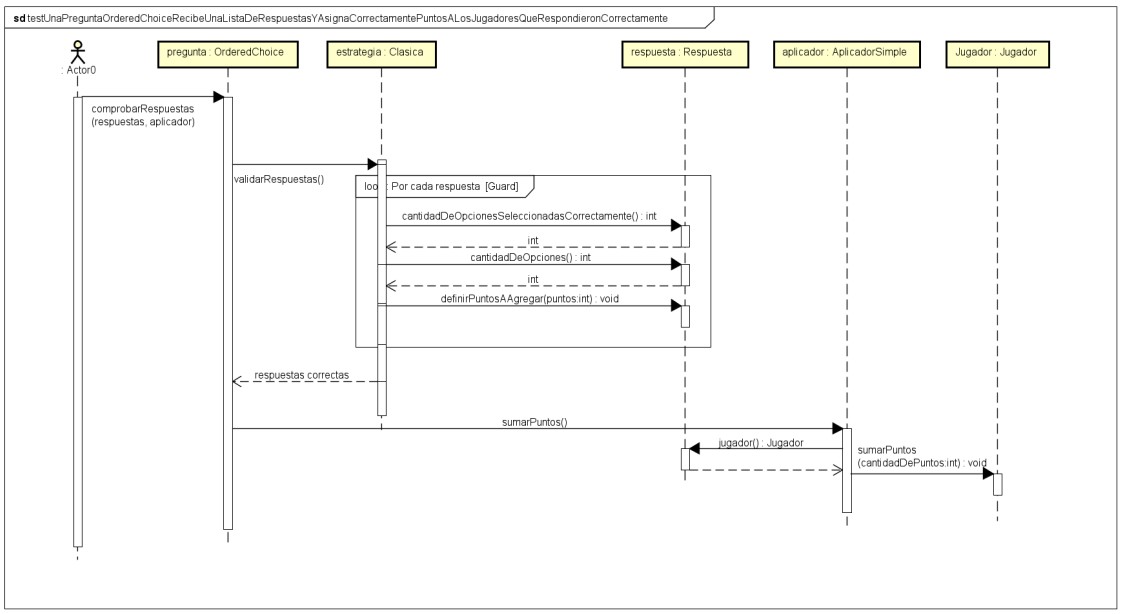

Figura 7: Pregunta Ordered Choice Clásica recibe una lista de respuestas y puntúa para una selección correcta.

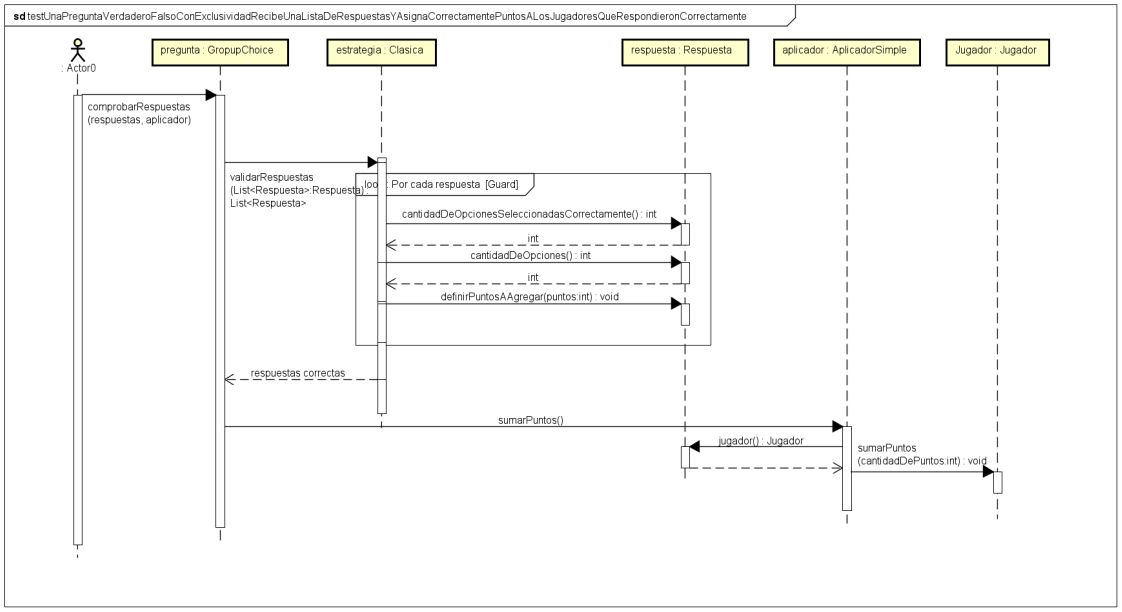

Figura 8: Pregunta Group Choice Clásica recibe una lista de respuestas y puntúa para una selección correcta.

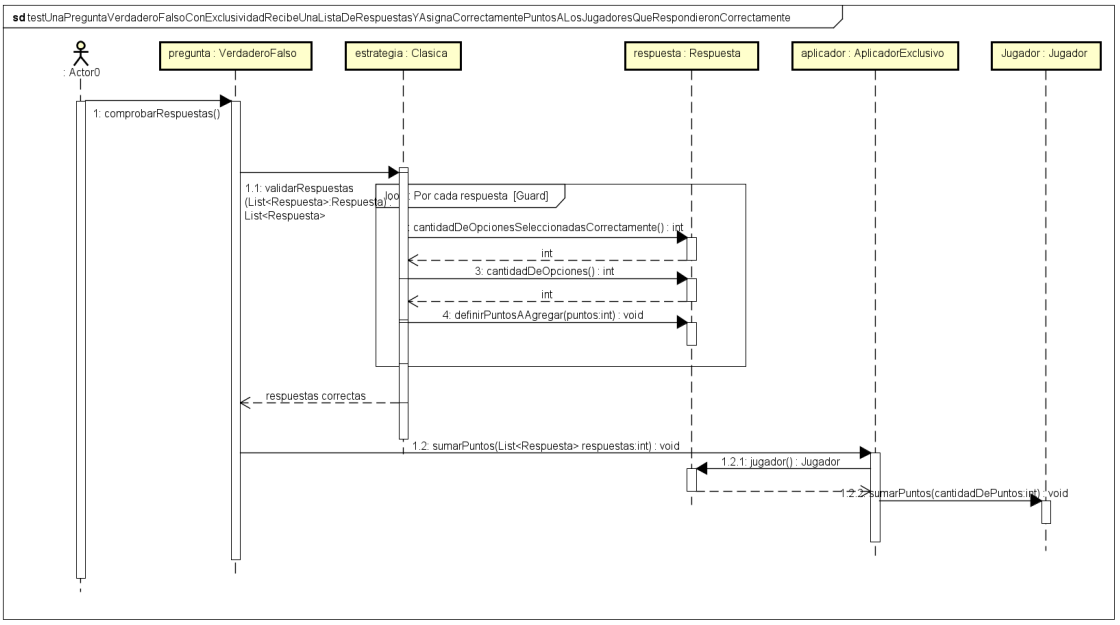

Figura 9: Pregunta Verdadero o falso exclusiva recibe una lista de respuestas y puntúa para una selección correcta.

5. Diagramas de paquetes

Figura 10: Relación de los paquetes más generales de nuestro programa.

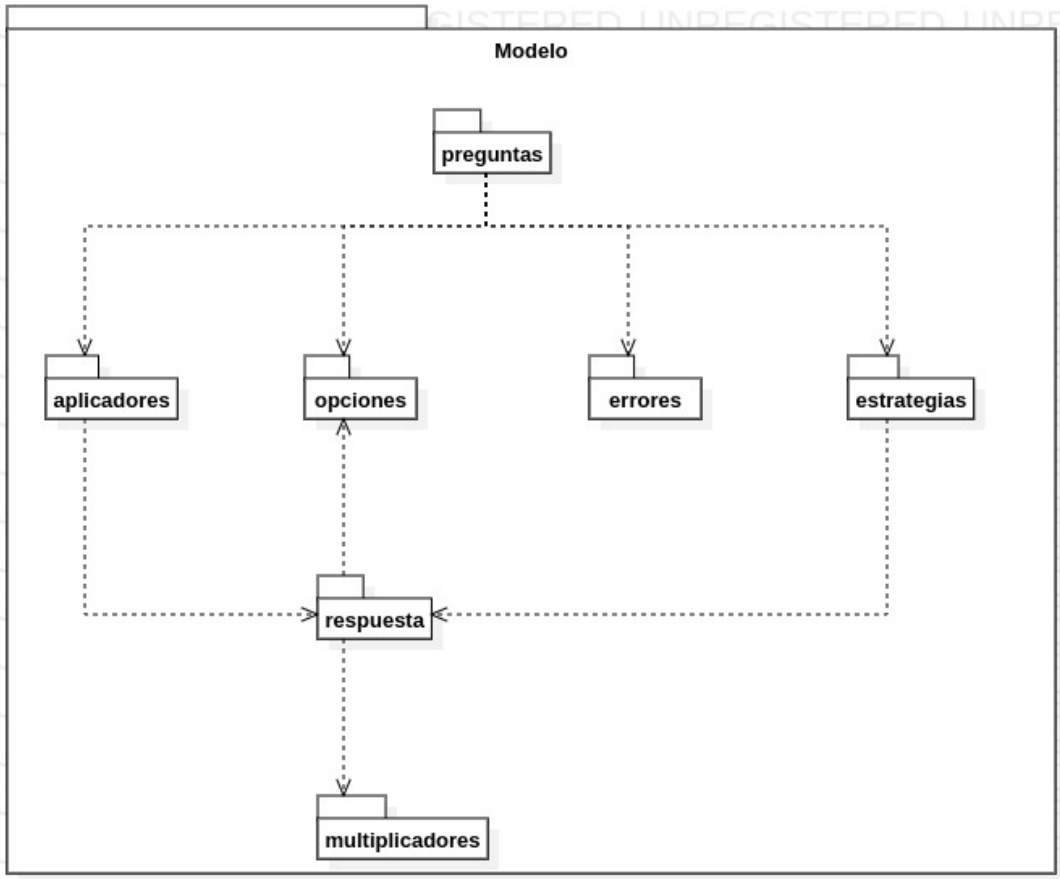

Figura 11: Relación de los paquetes dentro del paquete Modelo.

6. Diagramas de estados

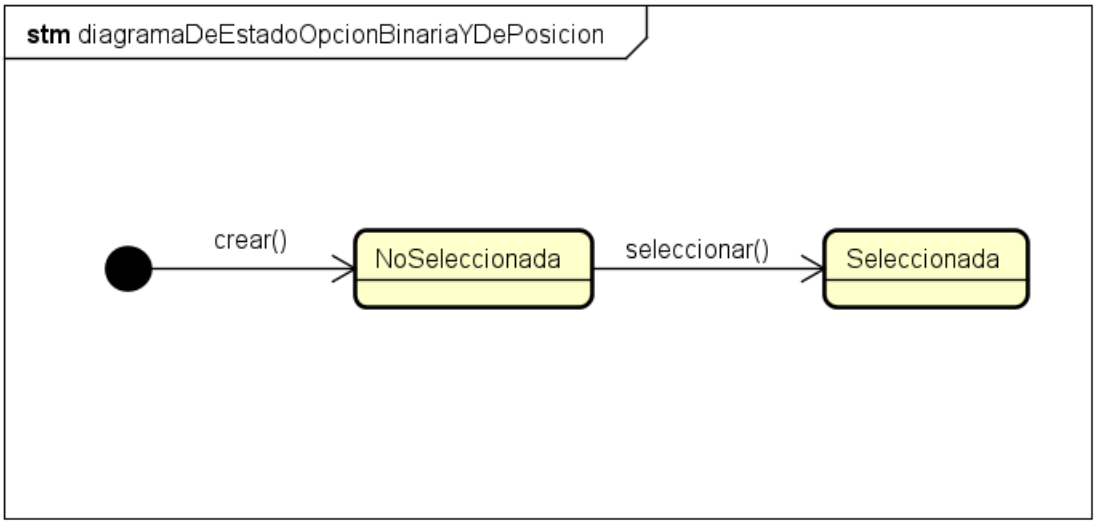

Figura 12: Una opción binaria y una opcion de posición cambian su estado de no seleccionadas a seleccionadas.

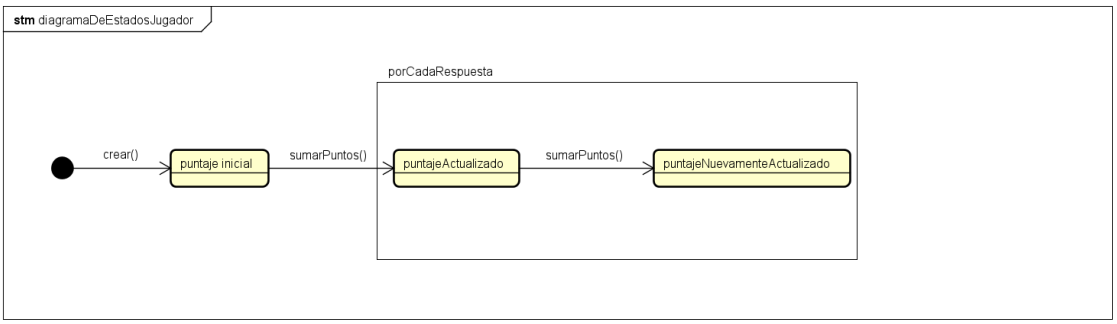

Figura 13: Un jugador cambia sus puntos por cada respuesta.

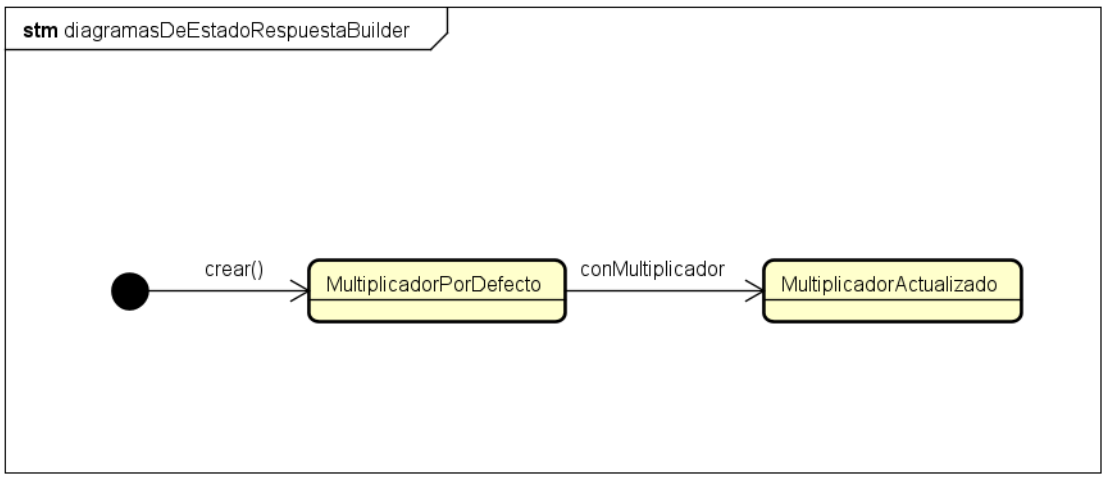

Figura 14: Un builder de respuesta puede cambiar el multiplicador con el que crea la respuesta del que tiene inicialmente.

7. Detalles de implementación

7.1. Los tipos de pregunta delegan el comportamiento de la puntuación en la clase Estrategia

Existen distintos tipos de preguntas con la misma estrategia de puntuación. Por ejemplo, las preguntas de tipo verdadero/falso clásico y multiple choice clásico comparten el mismo comportamiento al aplicar los puntos. Por esto se creó la clase Estrategia donde se delega el comportamiento. Esto permite no repetir código y que futuras modificaciones sean mas sencillas.

7.2. Las distintas opciones se dividen en interfaces distintas las cuales delegan la forma en la que se seleccionan

Para pedirle a una pregunta sus opciones se deberá conocer su clase exacta y no tratarla como a una clase madre porque según la pregunta es su tipo de opción. Esto se debe a que las preguntas se pueden seleccionar de formas distintas (sin parámetro, con un entero de posición, o con un string de el grupo).

7.3. Patrones de diseño utilizados

7.3.1. Factory

Para la creación de preguntas y opciones usamos un patrón de diseño factory ya que independientemente del tipo de pregunta u opción que quisiéramos crear solo basta llamar al método de la factory y la misma verifica que se cree bien, y así abstraer a la clase encargada de la lectura del archivo de preguntas, del modelo, sin que ésta sepa de tipos(para preguntas u opciones) ni estrategias (preguntas).

7.3.2. Builder

El patrón builder lo usamos para la creación de respuestas ya que nos permite setear los atributos antes de la creación. De esta manera, podemos construir objetos con diferentes características y de la manera deseada, usando los mismos métodos y así poder crear las respuestas con diferentes multiplicadores, listas de diferentes tipos de opción y otros atributos deseados.

7.3.3. Strategy

El patrón strategy fue seleccionado para simular, valga la redundancia las estrategias de puntuación, teniendo distintos tipos de estrategias concretas como pueden ser penalizable, clasica o puntaje parcial, permitiéndonos, utilizar la que fuera necesaria acorde a la funcionalidad requerida en dicho momento.

7.3.4. MVC

El patrón modelo-vista-controlador nos permite poder separar el modelo de la vista y comunicándolos por medio de uno o varios controladores.

8. Excepciones

1. CantidadDeOpcionesIncorrectaException: Sirve para informar que hay un error en la cantidad de opciones en una pregunta