UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

Año 2020 - 1er Cuatrimestre

Trabajo Práctico 2

Algoritmos y Programación III (75.07 / 95.02)

Integrantes	Padrón	E-mail
Luizaga, Ricardo	87528	rluizaga@fi.uba.ar
Leandro, Van Kemenade	102826	ivankemenade@fi.uba.ar
Julio, E. Castillo	80039	jecastillo@fi.uba.ar

Observaciones:

Introducción

La aplicación consiste en un juego por turnos, de dos jugadores conformado de un panel en el cual se mostraran preguntas con múltiples opciones de respuesta.
Cada pregunta será mostrada dos veces, una vez para cada jugador (al estar jugando los dos en la misma computadora, será responsabilidad de cada jugador no mirar la pantalla mientras el otro responde). El jugador dispone de un tiempo limitado para responder cada pregunta.
Existen varios tipos de preguntas, que asignan puntaje en forma diferenciada a cada jugador dependiendo de cómo responde cada uno.
También existen opciones como los multiplicadores y la exclusividad de puntaje que cada jugador puede utilizar para mejorar sus oportunidades de obtener puntos.
El objetivo del juego es lograr más puntos que el otro jugador respondiendo correctamente las preguntas.

Supuestos

Estas son algunas suposiciones que se tomaron para la implementación y que no están detalladas en el enunciado:

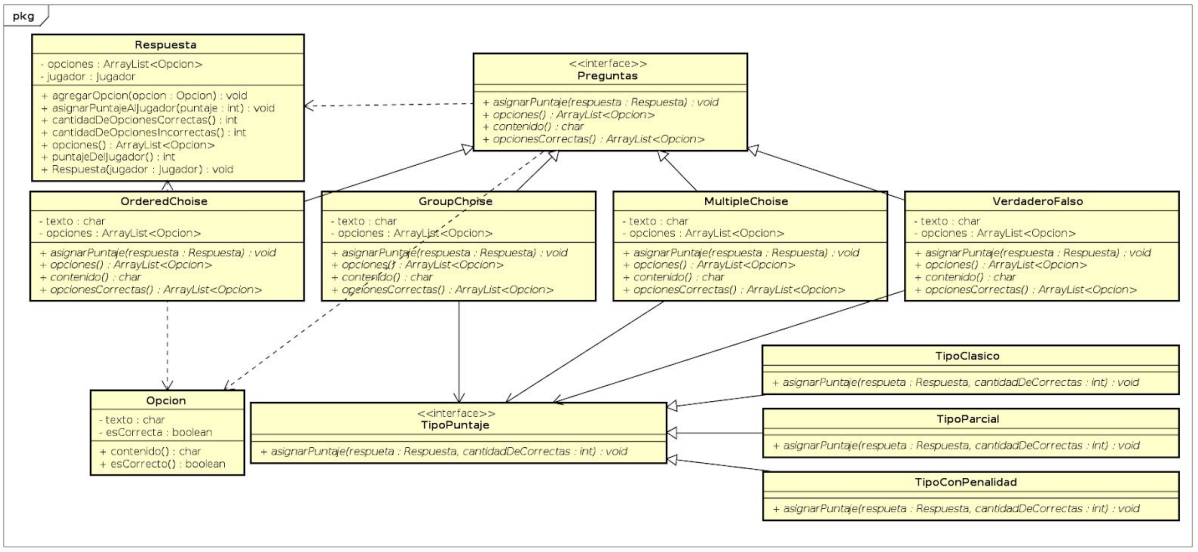
- Las preguntas tienen opciones las cuales pueden ser correctas o falsas.
- Solo se puede aplicar o multiplicador o exclusividad de puntaje por turno no ambas
- Los jugadores pueden tener puntos negativos

Modelo de dominio

- Clase Preguntas: Es responsable asignar el puntaje a los jugadores dependiendo de su respuesta.
- Clase Respuesta: Tiene almacenadas las opciones elegidas por el jugador.
  Tiene la responsabilidad de decirnos la cantidad de opciones correctas e incorrectas además también nos dice el puntaje del jugador y asigna el puntaje al jugador.
- Clase Opción: Las preguntas tienen una lista de opciones las cuales pueden ser correctas o incorrectas. La clase opción es responsable de ver si la opción es correcta.
- Interface Preguntas: Ante la variedad de preguntas esta es la encargada de proporcionar distintas alternativas para el cálculo del puntaje según el tipo de pregunta.
- Interface TipoPuntaje: Se decide implementar esta clase debido a que las preguntas del tipo GroupChoise, MultipleChoise, VerdaderoFalso pueden ser TipoClasico, TipoParcial, TipoConPenalidad debido a eso se asigna de forma distinta el puntaje.
- Clase KahootModel: Es la clase con la cual se implementa el modelo del juego, por lo cual tiene es responsable de cargar los jugadores, leer las preguntas del archivo JSON, mostrar las preguntas, opciones del turno, calcular los puntos del turno, cargar respuestas.
- Clase Turno: Es la responsable de avanzar en los turnos de los jugadores.

- Clase Ronda: Es la responsable de avanzar en cada ronda.
- Clase Jugador: Es usada para almacenar el nombre, puntaje además se encarga de saber si el jugador tiene aplicada el multiplicador de puntaje

Diagrama de Clases



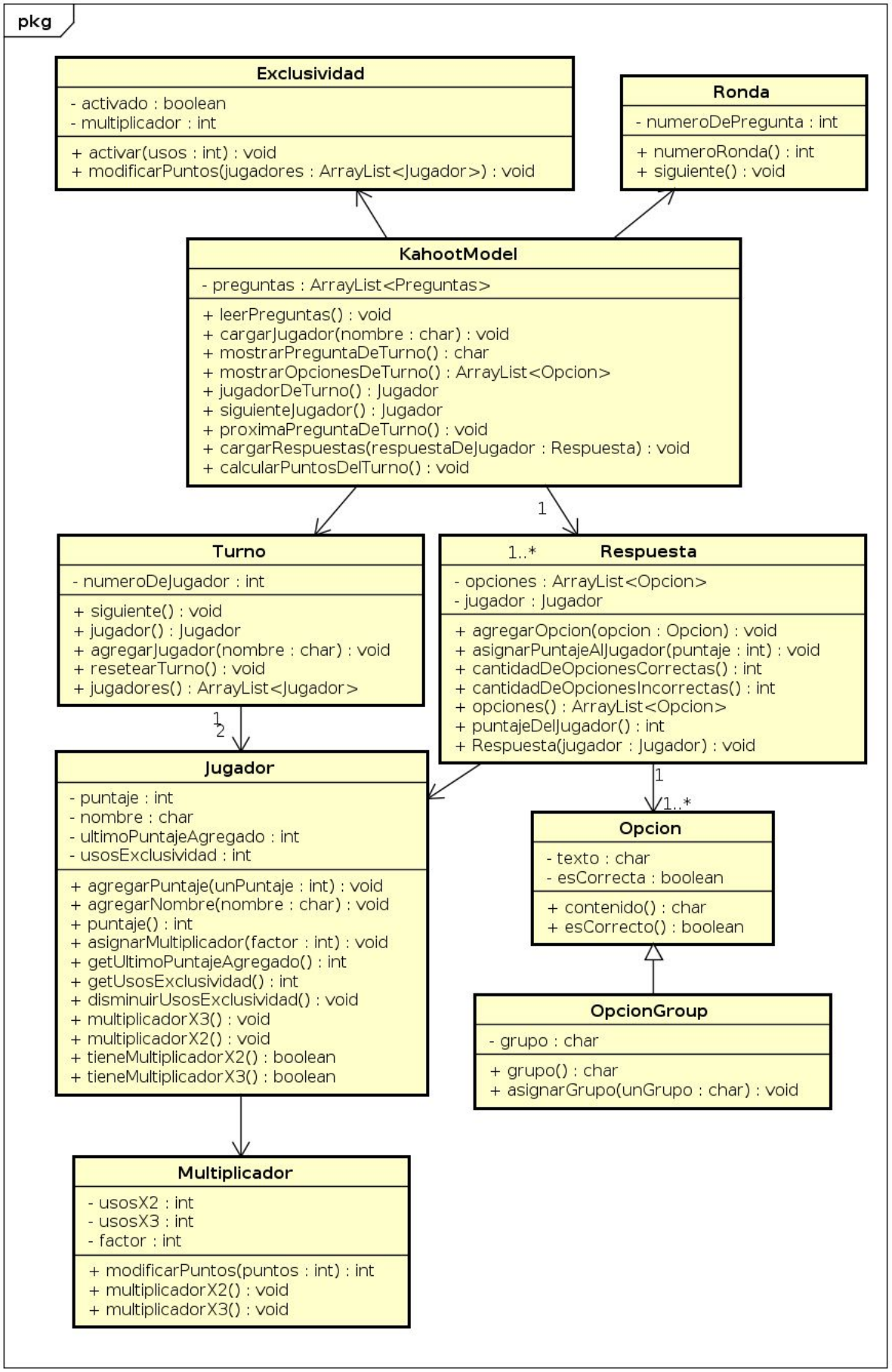
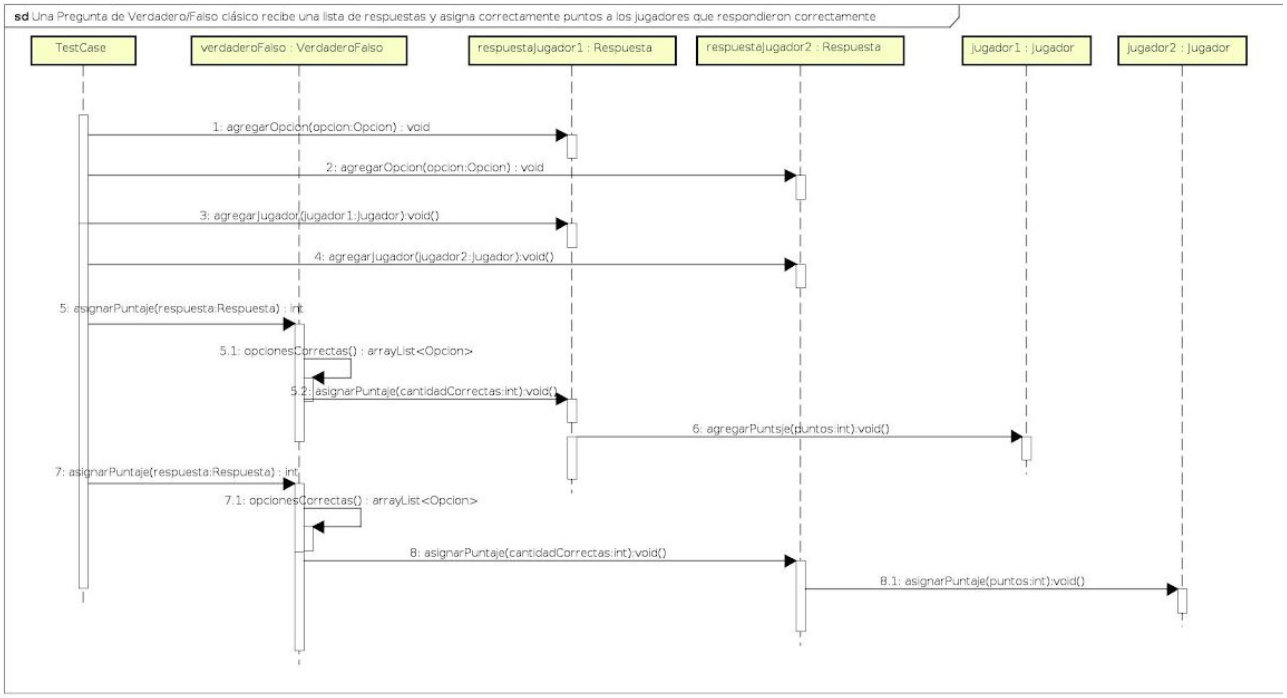
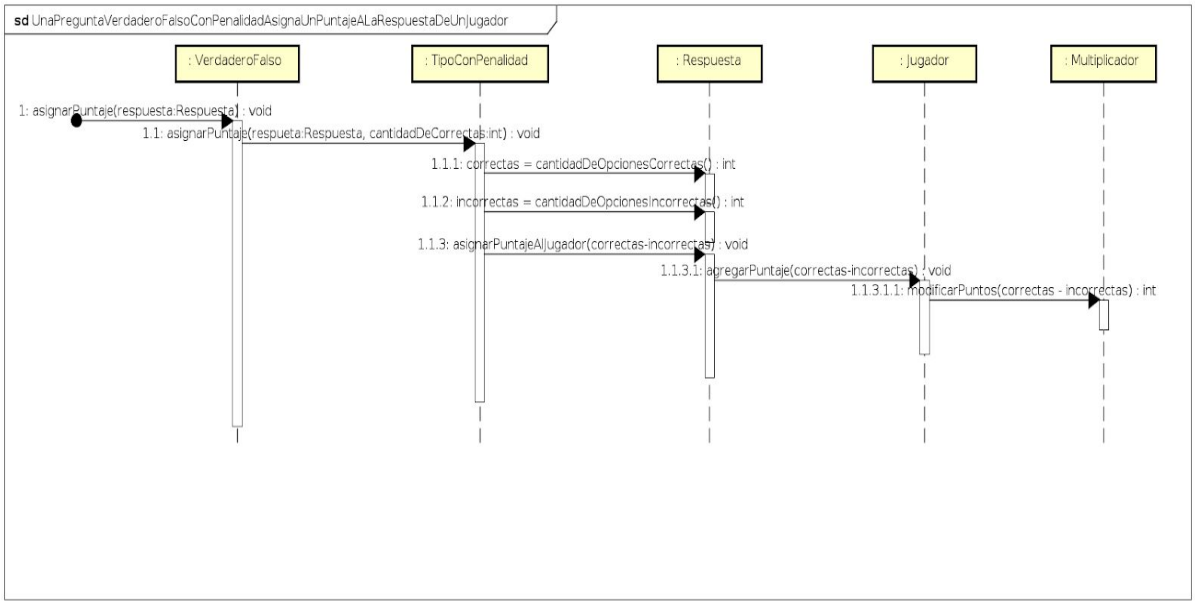
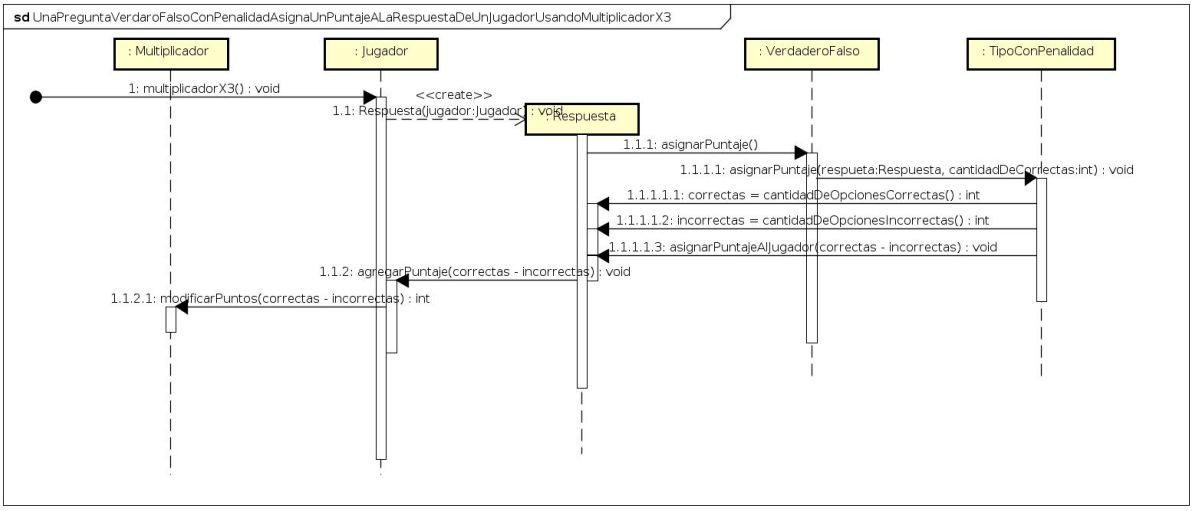
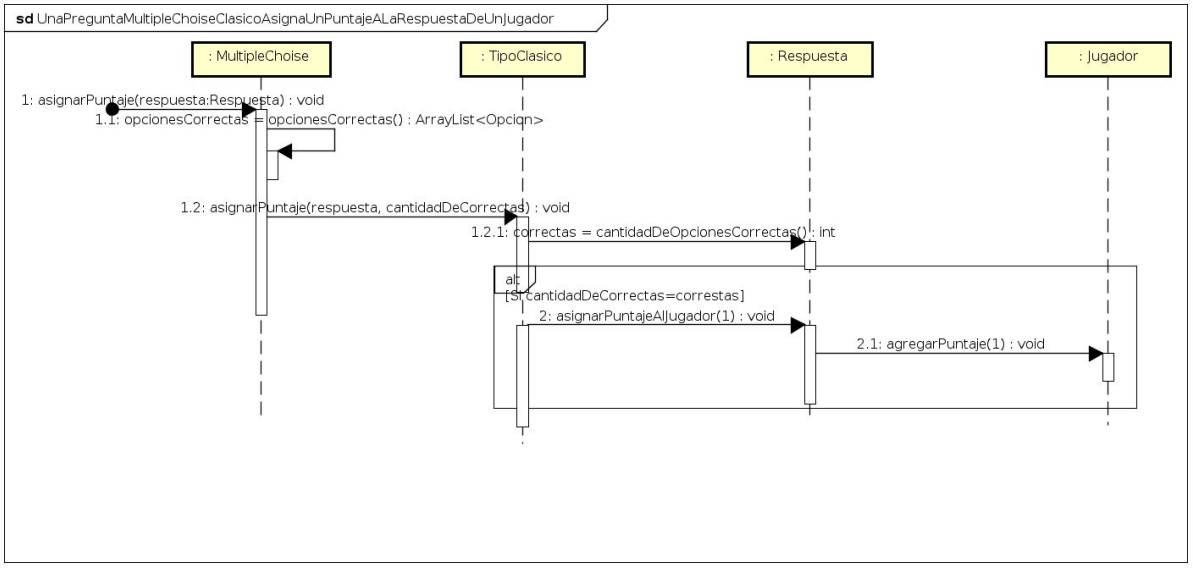

pkg
Exclusividad
- activado : boolean
- multiplicador : int
+ activar(usos : int) : void
+ modificarPuntos(jugadores : ArrayList<Jugador>) : void
Ronda
- numeroDePregunta : int
+ numeroRonda() : int
+ siguiente() : void
KahootModel
- preguntas : ArrayList<Preguntas>
+ leerPreguntas() : void
+ cargarJugador(nombre : char) : void
+ mostrarPreguntaDeTurno() : char
+ mostrarOpcionesDeTurno() : ArrayList<Opcion>
+ jugadorDeTurno() : Jugador
+ siguienteJugador() : Jugador
+ proximaPreguntaDeTurno() : void
+ cargarRespuestas(respuestaDeJugador : Respuesta) : void
+ calcularPuntosDelTurno() : void
1
Turno
- numeroDeJugador : int
+ siguiente() : void
+ jugador() : Jugador
+ agregarJugador(nombre : char) : void
+ resetearTurno() : void
+ jugadores() : ArrayList<Jugador>
1..*
Respuesta
- opciones : ArrayList<Opcion>
- jugador : Jugador
+ agregarOpcion(opcion : Opcion) : void
+ asignarPuntajeAlJugador(puntaje : int) : void
+ cantidadDeOpcionesCorrectas() : int
+ cantidadDeOpcionesIncorrectas() : int
+ opciones() : ArrayList<Opcion>
+ puntajeDelJugador() : int
+ Respuesta(jugador : Jugador) : void
1
2
Jugador
- puntaje : int
- nombre : char
- ultimoPuntajeAgregado : int
- usosExclusividad : int
+ agregarPuntaje(unPuntaje : int) : void
+ agregarNombre(nombre : char) : void
+ puntaje() : int
+ asignarMultiplicador(factor : int) : void
+ getUltimoPuntajeAgregado() : int
+ getUsosExclusividad() : int
+ disminuirUsosExclusividad() : void
+ multiplicadorX3() : void
+ multiplicadorX2() : void
+ tieneMultiplicadorX2() : boolean
+ tieneMultiplicadorX3() : boolean
1
1..*
Opcion
- texto : char
- esCorrecta : boolean
+ contenido() : char
+ esCorrecto() : boolean
OpcionGroup
- grupo : char
+ grupo() : char
+ asignarGrupo(unGrupo : char) : void
Multiplicador
- usosX2 : int
- usosX3 : int
- factor : int
+ modificarPuntos(puntos : int) : int
+ multiplicadorX2() : void
+ multiplicadorX3() : void

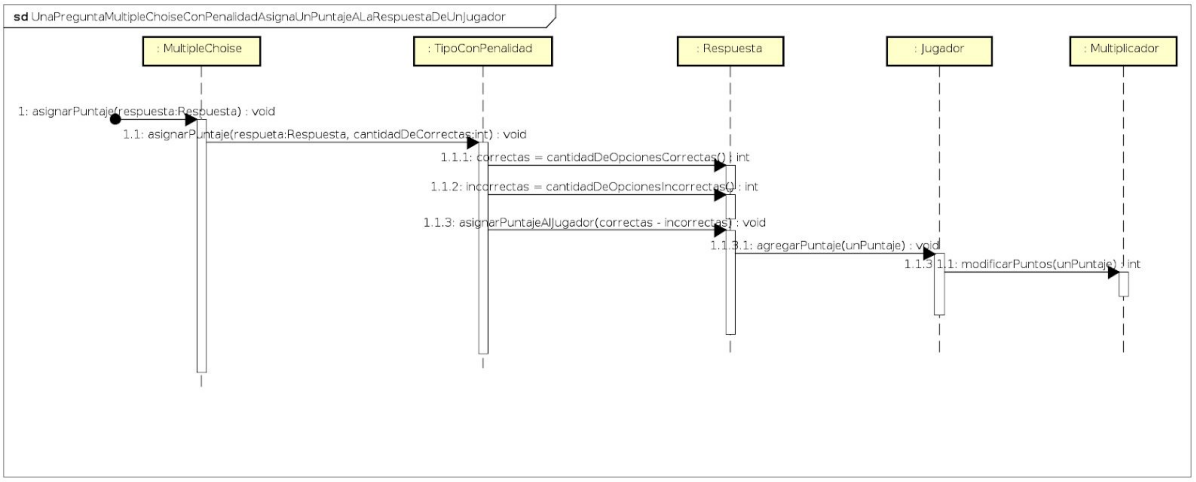
Diagrama de secuencia

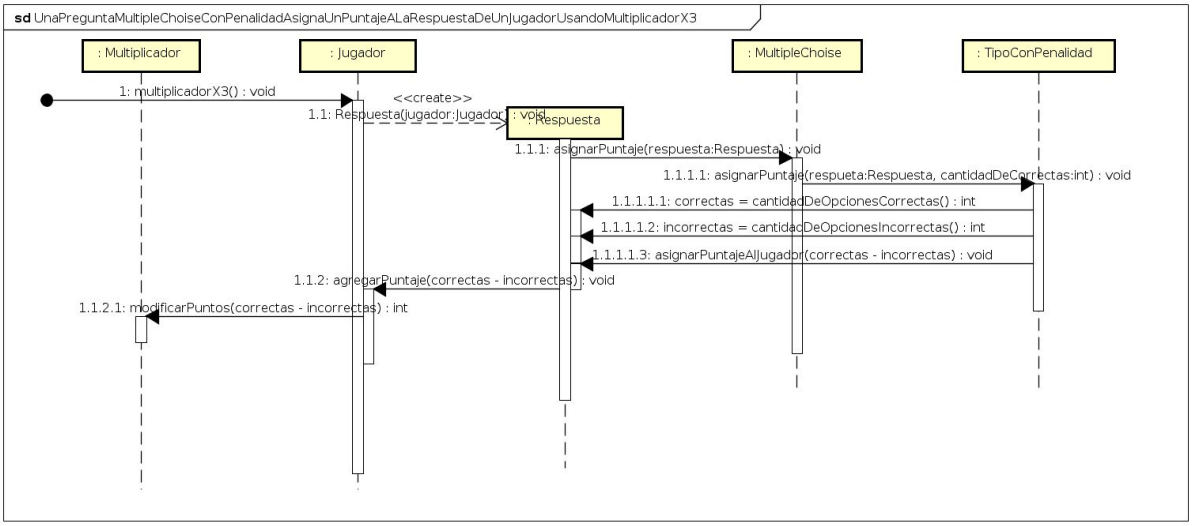


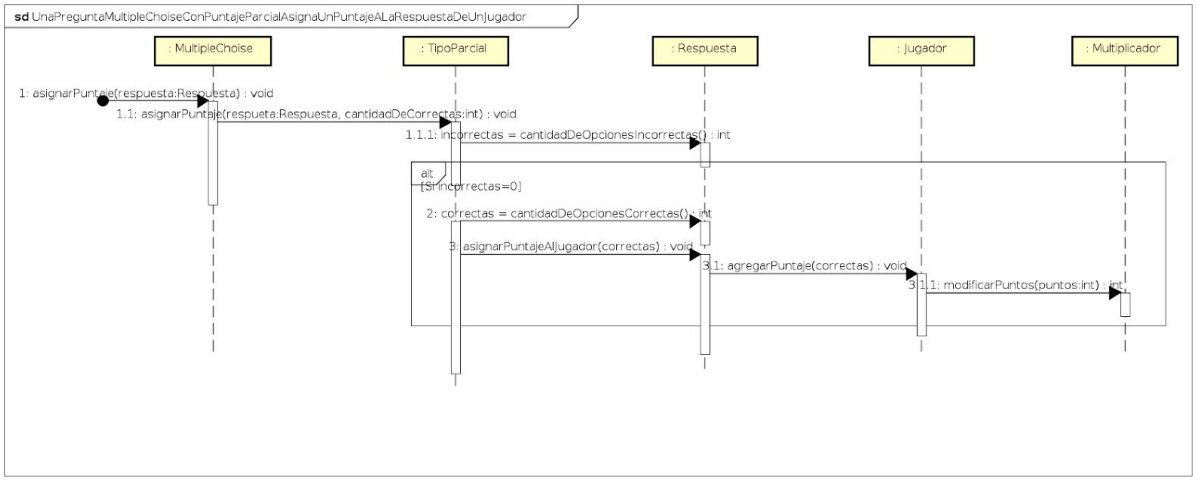


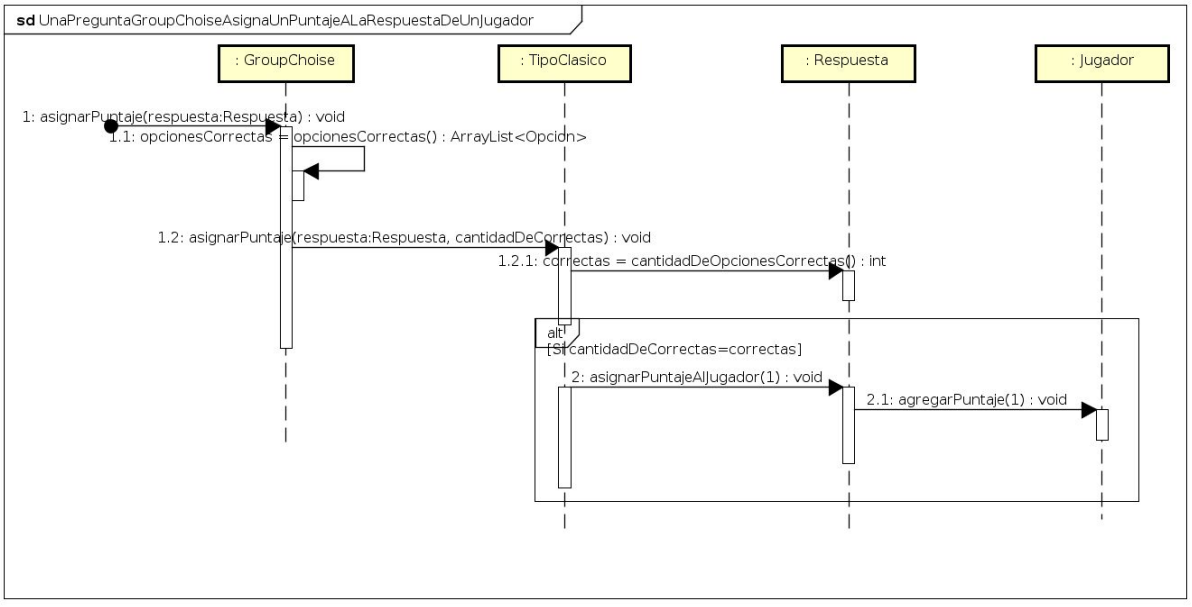

sd UnaPreguntaVerdaroFalsoConPenalidadAsignaUnPuntajeALaRespuestaDeUnJugadorUsandoMultiplicadorX3
: Multiplicador
: Jugador
: VerdaderoFalso
: TipoConPenalidad
1: multiplicadorX3() : void
<<create>>
1.1: Respuesta(jugador:Jugador) : void
: Respuesta
1.1.1: asignarPuntaje()
1.1.1.1: asignarPuntaje(respueta:Respuesta, cantidadDeCorrectas:int) : void
1.1.1.1.1: correctas = cantidadDeOpcionesCorrectas() : Int
1.1.1.1.2: incorrectas = cantidadDeOpcionesIncorrectas() : int
1.1.1.1.3: asignarPuntajeAlJugador(correctas - incorrectas) : void
1.1.2: agregarPuntaje(correctas - incorrectas) : void
1.1.2.1: modificarPuntos(correctas - incorrectas) : int


sd UnaPreguntaMultipleChoiseClasicoAsignaUnPuntajeALaRespuestaDeUnJugador
: MultipleChoise
: TipoClasico
: Respuesta
: Jugador
1: asignarPuntaje(respuesta:Respuesta) : void
1.1: opcionesCorrectas = opcionesCorrectas() : ArrayList<Opcion>
1.2: asignarPuntaje(respuesta, cantidadDeCorrectas) : void
1.2.1: correctas = cantidadDeOpcionesCorrectas() : int
alt
[Si cantidadDeCorrectas=correstas]
2: asignarPuntajeAlJugador(1) : void
2.1: agregarPuntaje(1) : void


sd UnaPreguntaMultipleChoiseConPenalidadAsignaUnPuntajeALaRespuestaDeUnJugador
: MultipleChoise
: TipoConPenalidad
: Respuesta
: Jugador
: Multiplicador
1: asignarPuntaje(respuesta:Respuesta) : void
1.1: asignarPuntaje(respueta:Respuesta, cantidadDeCorrectas:int) : void
1.1.1: correctas = cantidadDeOpcionesCorrectas() : int
1.1.2: incorrectas = cantidadDeOpcionesIncorrectas() : int
1.1.3: asignarPuntajeAlJugador(correctas - incorrectas) : void
1.1.3.1: agregarPuntaje(unPuntaje) : void
1.1.3.1.1: modificarPuntos(unPuntaje) : int


sd UnaPreguntaMultipleChoiseConPenalidadAsignaUnPuntajeALaRespuestaDeUnJugadorUsandoMultiplicadorX3
: Multiplicador
: Jugador
: MultipleChoise
: TipoConPenalidad
: Respuesta
1: multiplicadorX3() : void
<<create>>
1.1: Respuesta(jugador:Jugador) : void
1.1.1: asignarPuntaje(respuesta:Respuesta) : void
1.1.1.1: asignarPuntaje(respueta:Respuesta, cantidadDeCorrectas:int) : void
1.1.1.1.1: correctas = cantidadDeOpcionesCorrectas() : int
1.1.1.1.2: incorrectas = cantidadDeOpcionesIncorrectas() : int
1.1.1.1.3: asignarPuntajeAlJugador(correctas - incorrectas) : void
1.1.2: agregarPuntaje(correctas - incorrectas) : void
1.1.2.1: modificarPuntos(correctas - incorrectas) : int


sd UnaPreguntaMultipleChoiseConPuntajeParcialAsignaUnPuntajeALaRespuestaDeUnJugador
: MultipleChoise
: TipoParcial
: Respuesta
: Jugador
: Multiplicador
1: asignarPuntaje(respuesta:Respuesta) : void
1.1: asignarPuntaje(respueta:Respuesta, cantidadDeCorrectas:int) : void
1.1.1: incorrectas = cantidadDeOpcionesIncorrectas() : int
alt
[SI incorrectas=0]
2: correctas = cantidadDeOpcionesCorrectas() : int
3: asignarPuntajeAlJugador(correctas) : void
3.1: agregarPuntaje(correctas) : void
3.1.1: modificarPuntos(puntos:int) : int


sd UnaPreguntaGroupChoiseAsignaUnPuntajeALaRespuestaDeUnJugador
: GroupChoise
: TipoClasico
: Respuesta
: Jugador
1: asignarPuntaje(respuesta:Respuesta) : void
1.1: opcionesCorrectas = opcionesCorrectas() : ArrayList<Opcion>
1.2: asignarPuntaje(respuesta:Respuesta, cantidadDeCorrectas) : void
1.2.1: correctas = cantidadDeOpcionesCorrectas() : int
alt
[SI cantidadDeCorrectas=correctas]
2: asignarPuntajeAlJugador(1) : void
2.1: agregarPuntaje(1) : void

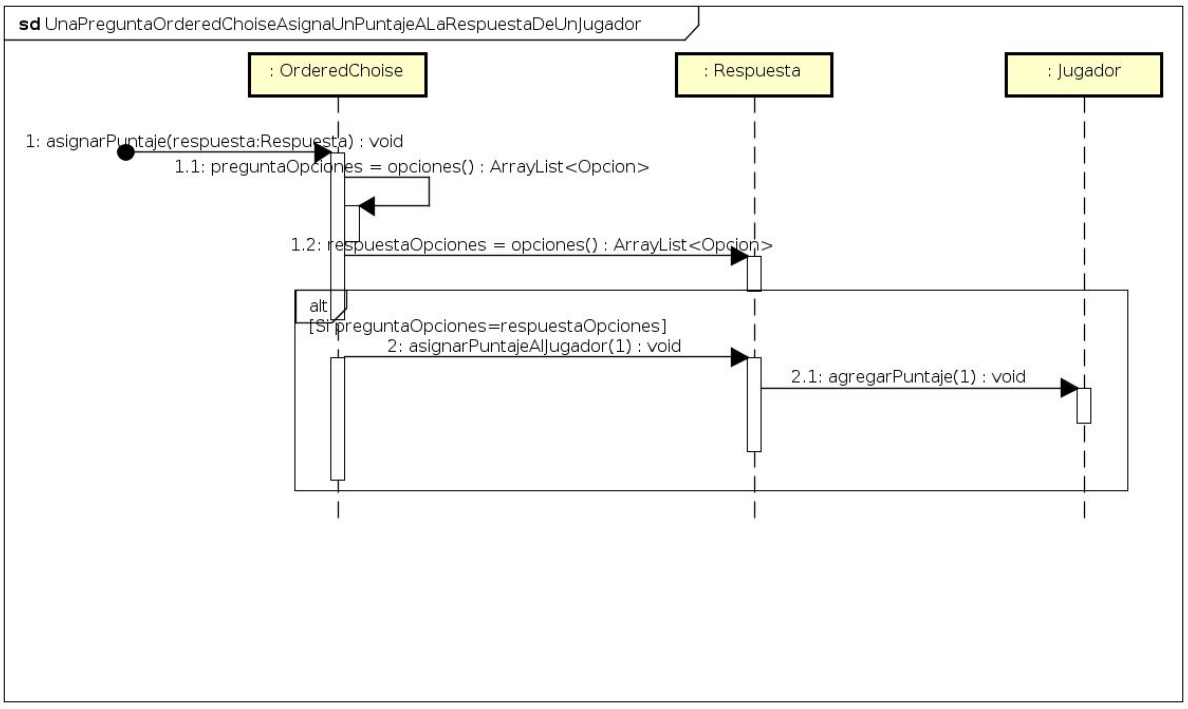
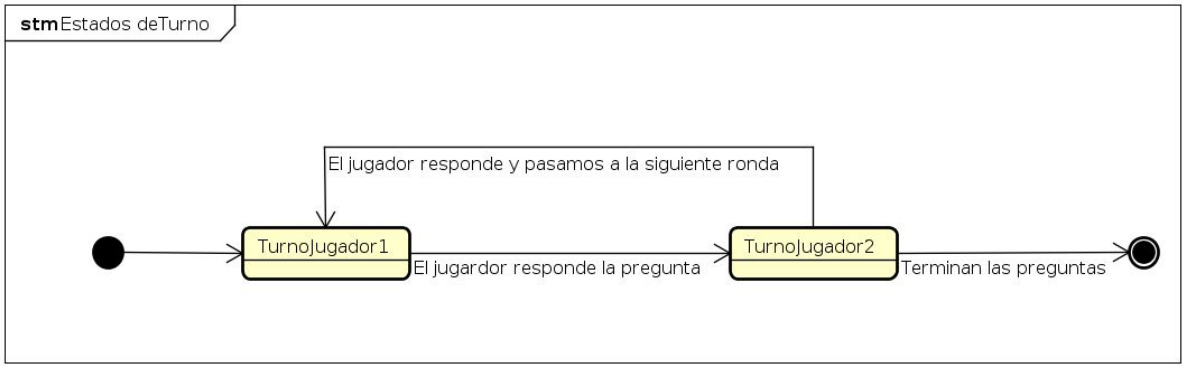
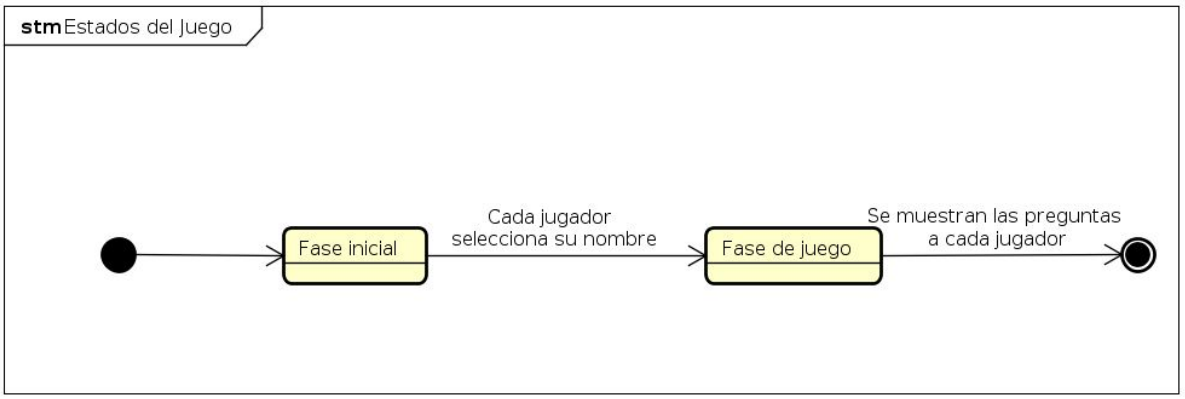
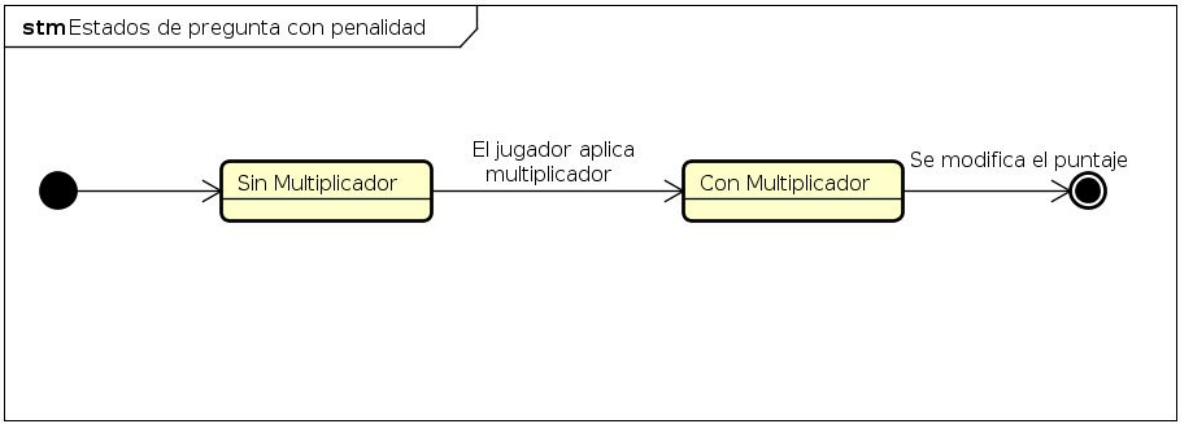


Diagrama de estado







Detalles de implementación

TIPO DE PUNTUACIÓN: Con el fin de manejar el cálculo de las puntuaciones según la respuesta, se crearon las clases TipoClásico, TipoConPenalidad y TipoParcial, a las cuales las preguntas que poseen las características equivalentes deriva ese componente del cálculo.

DESEREALIZACION: Para dicha tarea se utilizó la librería Jackson, y el proceso de lectura y creación de preguntas se realiza mediante las clases LectorDePregunta , CreadorDePregunta y PreguntaACrearInfo.

PATRON UTILIZADO:

MVC: Patrón de arquitectura, con el fin de separar responsabilidades.

Excepciones

- PreguntaNoIdentificadaExcepcion: Se creó con el fin de detectar si no se identificó la pregunta en el archivo JSON, se atrapa cuando se procede a crear las preguntas con la información leída.
- NoHayPreguntasCargadasExcepcion: Se creó para validar si el modelo posee preguntas con las cual trabajar, se llama al finalizar la creación de las preguntas.
- IOException: Se utilizó a la hora de leer el archivo JSON para verificar si existe.