FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Java

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

Alumno	Padrón	Mail
Ignacio Zarlenga	104089	ignaciozarlenga@hotmail.com
Ivo Arbanas	102287	ivo.arbanas@gmail.com
Manuel Elia	102425	mlongoe@fi.uba.ar
Luciano Montes	102536	lmontes@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar el juego Kahoot en Java con interfaz grafica utilizando JavaFX y usando diferentes herramientas para la integracion continua (Travis) y la build del proyecto (Maven).

2. Supuestos

Puntaje Negativo: Se contempla como valido el caso de que un jugador pueda tener un puntaje negativo, en el caso de responder una pregunta incorrecta la cual pueda quitarle los puntos necesarios con puntaje final menor a cero.

Respuesta Jugador: El jugador siempre debe tener una respuesta, no puede pasar ronda hasta que no haya seleccionado su opcion.

Carga de Preguntas: Para el mismo usamos el formato de intercambio de datos **JSON**. Cada tipo de pregunta debe presentarse de una forma especifica e indicando su clase. Todas las preguntas requieren datos como **enunciado**, **tipoDePregunta** y sus respectivas **opciones**.

- **VerdaderoFalso:** Ademas de detallar el enunciado de la pregunta es necesario indicar opcion correcta sea **false** o **true**.
- **MultipleChoice:** Es necesario indicar cuales opciones son correctas e incorrectas mediante dos listas al cargar la pregunta.
- **OrderedChoice:** Las opciones deberan presentarse ordenadas correctamente para su posterior verificacion.
- **GroupChoice:** La presente pregunta requiere al igual que MultipleChoice dos listas de opciones, cada una correspondiente a su dicho grupo. Ademas se debera especificar los nombres de los grupos.

3. Modelo de dominio

Paquete Modelo:

- Clase Kahoot: Esta clase contiene a los jugadores, las preguntas de cada ronda, el sistemaDePuntaje y el numero de ronda actual. La misma interactua con las clases contenidas delegando comportamiento para el correcto funcionamiento de la aplicacion.
- Clase SistemaPuntaje: Se encarga de modificar el puntaje de los jugadores, ya sea sumando o restando sus puntos, dependiendo el puntaje indicado por una previa verificacion de la pregunta.
- Clase Puntaje: Contiene el valor entero de un puntaje, el cual puede ser cambiado o simplemente obtener dicho valor mediante sus metodos.
- Clase Ronda: En la misma se encuentra la pregunta la cual sera presentada a los usuarios en dicha ronda. Se encarga de obtener los puntajes de cada jugador delegando a la pregunta, sea una respuesta normal, con bonificadores o exclusividad de puntaje.
- Clase Usuario: Representa al jugador, conteniendo su nombre, puntaje, respuesta y bonificadores disponibles. Cada ronda se carga una respuesta correspondiente a dicha pregunta. La misma se vacia al avanzar al siguiente turno.
- Clase Opcion: Contiene el nombre de la opcion. Representa una unica opcion de las cuales estan compuestas las preguntas y pueden ser elegidas por el Usuario.

- Clase Pregunta: Clase Abstracta y madre de todas las preguntas, tiene los atributos en comun de todas ellas, como lo son enunciado y la lista de opciones. Tambien esta la declaracion de metodos abstractos como calcularPuntajeParaRespuesta, calcularPuntajeConMultiplicador y calcularPuntajeConExclusividad, para los cuales usamos polimorfismo, con una distinta implementacion para el mismo mensaje dependiendo la clase de pregunta.
- Clase VerdaderoFalso: Clase hija de Pregunta, su constructor recibe por parametro si la respuesta correcta es verdadera o falsa. Almacena cual es la opcionCorrecta y opcionIncorrecta como atributos. Esta clase tambien es abstracta ya que los metodos de calcular puntaje mencionados en Pregunta, se implementan segun la entidad.
- Clase VerdaderoFalsoClasico: Hereda de VerdaderoFalso, implementa todos metodos abstractos de ella, chequeando la respuesta del jugador con la opcion correcta y devolviendo un punto si la respuesta es correcta o nulo si no lo es.
- Clase VerdaderoFalsoPenalidad: Hereda de VerdaderoFalso, responde a los mismos metodos que VerdaderoFalsoClasico resolviendo el polimorfismo, pero con la posibilidad de restar puntos en el caso de que la respuesta sea incorrecta.
- Clase MultipleChoice: Hereda de Pregunta. Representacion general de todos los tipos de multiple choice. Contiene dos listas, opcionesCorrectas y opcionesIncorrectas las cuales son pasadas por parametros en el constructor. Esta clase es abstracta por la misma razon que VerdaderoFalso aunque implementa el metodo esCorrecta() el cual es comun en todas sus clases hijas hijas.
- Clase MultipleChoiceClasico: Hereda de MultipleChoice. Implementa los metodos abstractos devolviendo su respectivo puntaje, sea uno o nulo.
- Clase MultipleChoicePenalidad: Hereda de MultipleChoice. Al igual que el clasico, implementa los metodos de calcular puntaje, devolviendo tambien puntaje negativo.
- Clase MultipleChoiceParcial: Hereda de MultipleChoice. Esta entidad es capaz de sumar mas de un punto si es que ninguna opcion de la respuestas del usuario haya sido incorrecta.
- Clase OrderedChoice: Hereda de Pregunta. Representa a una pregunta con un orden de opciones especifico. Implementa los metodos de pregunta, verificando la respuesta del usuario con su lista de opciones la cual se encuentra ordenada correctamente, devolviend el valor correspondiente a asignar al jugador.
- Clase GroupChoice: Hereda de Pregunta. Representa a una pregunta la cual debe ser contestada asigando cada opcion en el grupo correspondiente. Tiene atributos propios como los nombres de los grupos y listas con las opciones de ambos. Es capaz de implementar los metodos abstractos de Pregunta, dividiendo tareas con metodos como opcionEstaEnListaGrupo(listaGrupo,Opcion) y devuelve el puntaje correspondiente.
- Clase Multiplicador: Clase abstracta madre de MultiplicadorInactivo, X2 y X3. Representa la clase general de bonificaciones, con el metodo aplicarValorDelMultiplicador(Puntaje) el cual se resuelve con polimorfismo dependiendo de la entidad.
- Clase MultiplicadorInactivo: Hereda de Multiplicador. Representa un bonificador ya utilizado. Implementa el metodo aplicarValorDelMultiplicador(Puntaje) sin modificar valores del puntaje.
- Clase X2: Hereda de Multiplicador. Implementa aplicarValorDelMultiplicador(Puntaje) retornando el doble del puntaje recibido.
- Clase X3: Hereda de Multiplicador. Implementa aplicarValorDelMultiplicador(Puntaje) retornando el triple del puntaje recibido.

- Clase ExclusividadDePuntaje: Clase abstracta y madre de ExclusividadDePuntajeNulo, ExclusividadDePuntajeDoble y ExclusividadDePuntajeCuadruple. Declarado el metodo abstracto aplicarExclusividad(Puntaje) resuelto con polimorfismo en sus clases hijas.
- Clase ExclusividadDePuntajeNulo: Hereda de ExclusividadDePuntaje. Implementa el metodo aplicarExclusividad(Puntaje) sin hacer cambios en el puntaje.
- Clase ExclusividadDePuntajeDoble: Hereda de ExclusividadDePuntaje. Implementa el metodo aplicarExclusividad(Puntaje) devolviendo el doble del puntaje.
- Clase ExclusividadDePuntajeCuadruple: Hereda de ExclusividadDePuntaje. Implementa el metodo aplicarExclusividad(Puntaje) retornando el cuadruple del puntaje.

4. Diagramas de clase

En el diagrama de la figura 1 podemos observar las clases principales, como ellas interactuan entre si para su correcto funcionamiento

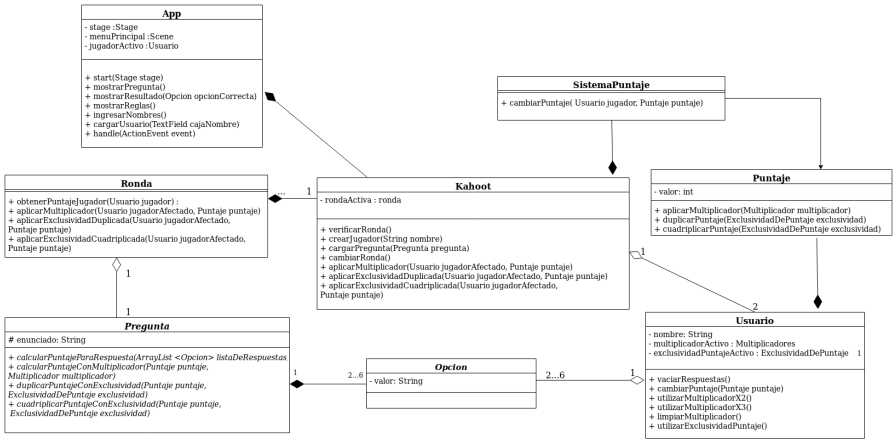

Figura 1: Diagrama de Clases Principales.

En la figura 2 tenemos una vista mas detallada de la clase Pregunta y todas su hijas, las cuales son capaces de resolver con polimorfismo un mismo metodo, implementado de distinta manera.

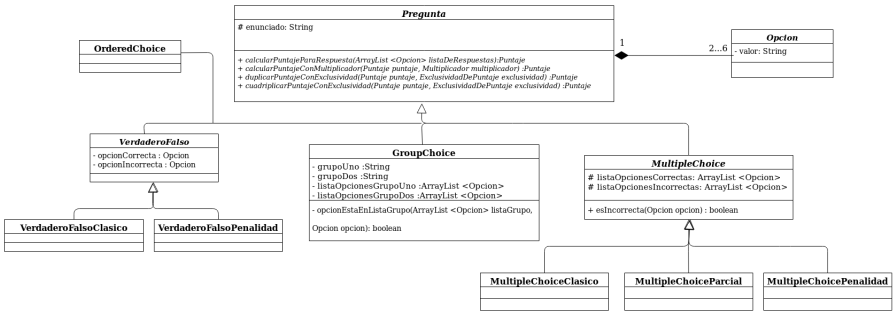

Figura 2: Diagrama de Clases Pregunta.

Por ultimo en la figura 3, vemos el diagrama de clases para Usuario, que clases contiene y como interactua con ellas.

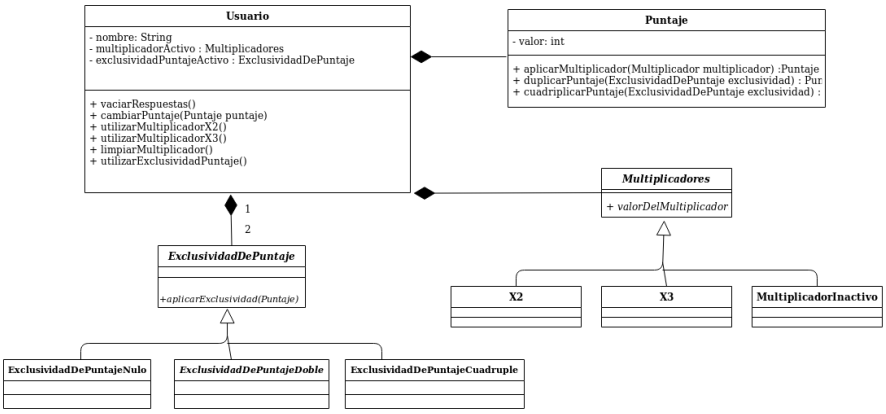

Figura 3: Diagrama de Clases Usuario.

5. Detalles de implementación

Método calcularPuntajeParaRespuesta(Arraylist<Opcion>listaDeRespuestas)

-Metodo abstracto declarado en la clase madre de todas las preguntas e implementado en todas sus hijas.

Clase Pregunta:

```
public abstract Puntaje calcularPuntajeParaRespuesta(ArrayList<Opcion> listaDeRespuestas);
```

Figura 4: Codigo Pregunta.

- En el caso de VerdaderoFalso solo requiere verificar si el unico elemento de la lista de respuestas (anteriormente seleccionado por el Usuario) es igual a la opcion correcta de dicha instancia. Retornando en el caso de Clasico un punto si es correcto y cero puntos si es incorrecta. Tambien para Penalidad con la posibilidad de retonar menos un punto si a respuesta es incorrecta.

Clase VerdaderoFalsoClasico:

```
public  Puntaje calcularPuntajeParaRespuesta(ArrayList<Opcion> listaDeRespuestas) {
    if (listaDeRespuestas.get(0).valor().equals(opcionCorrecta().get(0).valor())){
        Puntaje puntajePositivo = new Puntaje( valor: 1);
        return puntajePositivo;
    }
    Puntaje puntajeNulo = new Puntaje( valor: 0);
    return puntajeNulo;
}
```

Figura 5: Codigo VerdaderoFalsoClasico.

Clase VerdaderoFalsoPenalidad:

```
public  Puntaje calcularPuntajeParaRespuesta(ArrayList<Opcion> listaDeRespuestas) {
    if (listaDeRespuestas.get(0).valor() == opcionCorrecta().get(0).valor()){
        Puntaje puntajePositivo = new Puntaje( valor: 1);
        return puntajePositivo;
    }
    Puntaje puntajeNegativo = new Puntaje( valor: -1);
    return puntajeNegativo;
}
```

Figura 6: Codigo VerdaderoFalsoPenalidad.

- Para la instancia de OrderedChoice lo que hace es comparar el orden de la lista de respuestas del Usuario con las lista de opciones de la pregunta, la cual esta ordenada de forma correcta. Si ambas coinciden en todos los casos retorna un punto y cero en el caso de tener algun error.

Clase OrderedChoice:

```
public Puntaje calcularPuntajeParaRespuesta(ArrayList<Opcion> listaDeRespuestas){

    for (int i=0; i < listaDeRespuestas.size();i++){
        if(!opciones.get(i).valor().equals(listaDeRespuestas.get(i).valor())){
            Puntaje puntajeNulo = new Puntaje( valor: 0);
            return puntajeNulo;
        }
    }
    Puntaje puntajePositivo = new Puntaje( valor: 1);
    return puntajePositivo;
}
```

Figura 7: Codigo OrderedChoice.

6. Excepciones

Para el caso de las excepciones resolvimos el problema reemplazando las clases Exceptions por carteles emergentes destinados al usuario. Casos mas comunes como por ejemplo, cuando no ingresan nombres los jugadores:

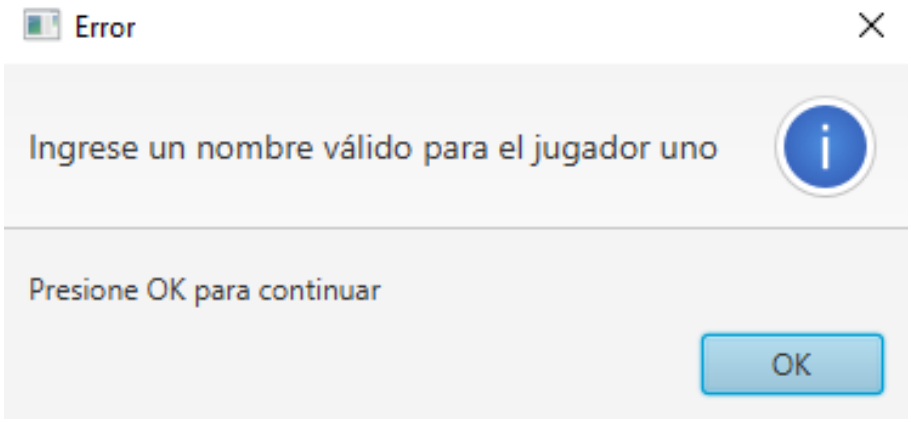

Figura 8: Alerta 1.

Caso comun, cuando el jugador no selecciono una respuesta a la pregunta actual:

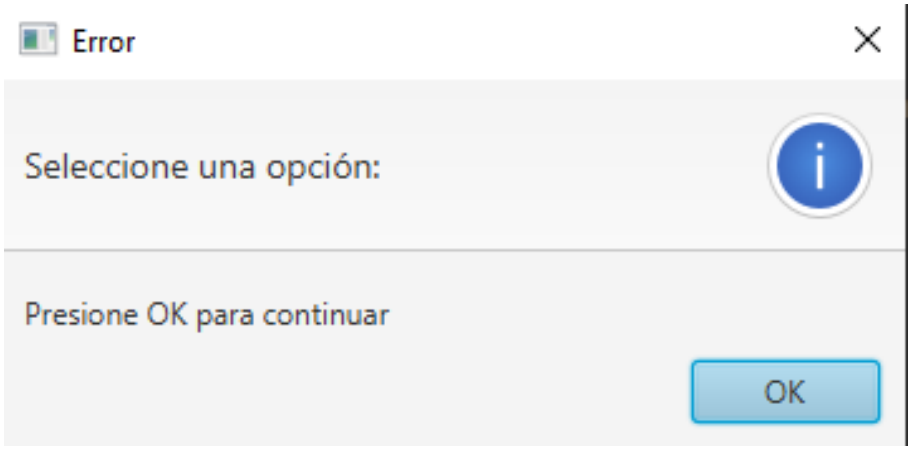

Figura 9: Alerta 2.