Enunciado del Problema – Juego *Racer*

Racer es un juego de tablero multijugador. Es un juego que tiene componentes de azar y también de conocimiento, pues se premia a los jugadores que contestan preguntas correctamente, tanto para evitar penalizaciones como también para avanzar más rápidamente hacia la meta. Las características del juego hacen que se mantenga el suspenso de quién resultará ganador hasta el último momento.

Dado que la idea es que puedan participar jugadores de distintas edades y niveles de conocimiento, habrá dos tipos de jugadores: *novatos* y *avanzados*. Al iniciar el juego, deberá indicarse la cantidad de jugadores que participarán en la partida (2 a 4 jugadores)

Reglamento del Juego

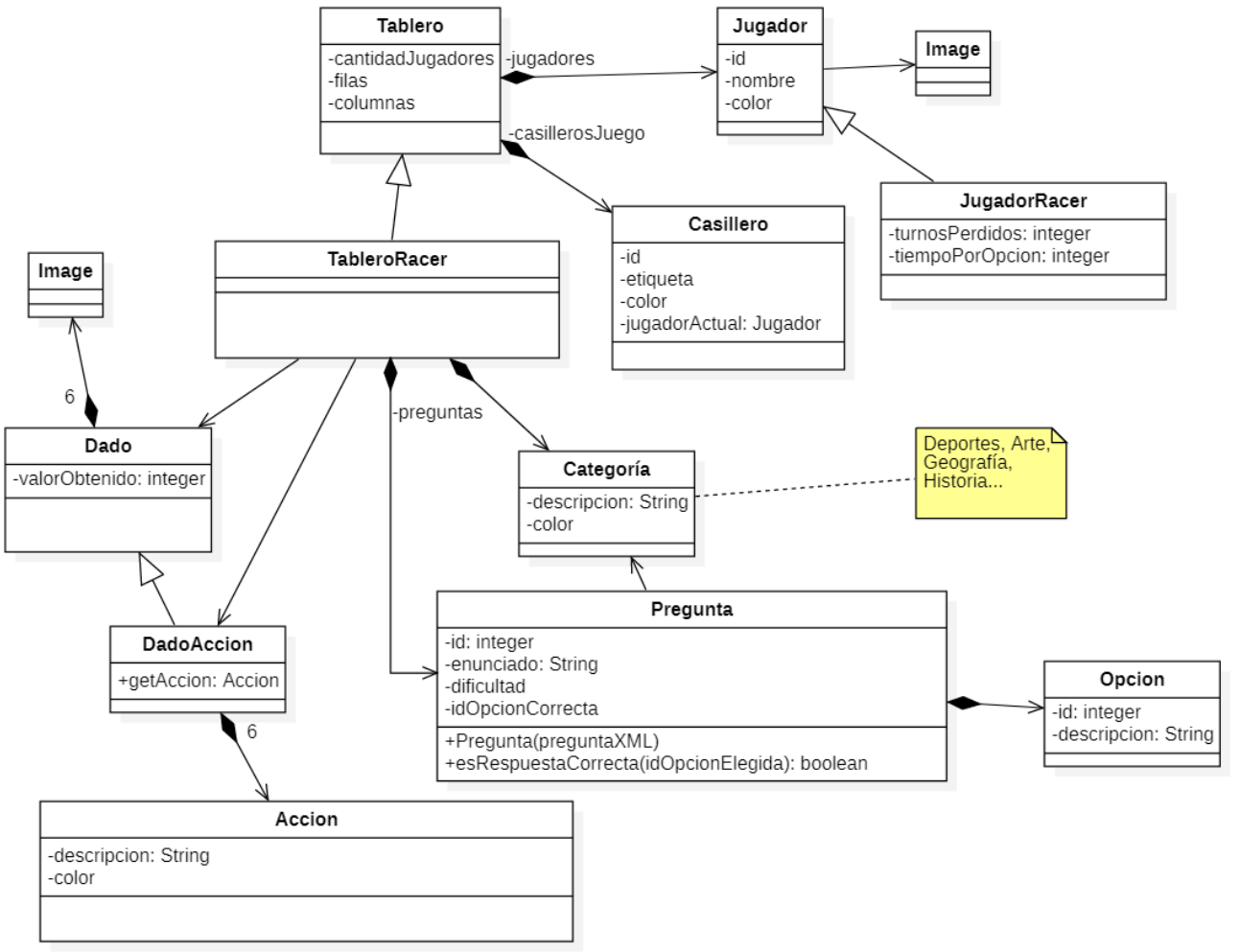
- Cada jugador elige la ficha del color de su preferencia, una imagen/icono de su preferencia, ingresa su nombre y su nivel (novato o experto). Su ficha se coloca en un casillero de Punto de Partida.
- Existen 2 dados diferentes:
    - Dado numérico: el tradicional dado con valores de 1 a 6.
    - Dado de *Acciones*: Cada cara está identificada con un color diferente, cuyos significados son:
        - *Rojo*: Avanza el valor obtenido en el dado numérico, pero pierde el próximo turno (no podrá lanzar los dados en el siguiente turno)
        - *Azul*: Exhibe una pregunta al jugador. Duplica casillas a avanzar si contesta bien. Si contesta mal, no avanza y pierde el próximo turno.
        - *Naranja*: El jugador que lanzó el dado no avanza ni retrocede. El jugador de “la derecha” (el del turno siguiente) deberá responder una pregunta. Si contesta bien, dicho jugador avanza el número obtenido en el dado numérico. Si contesta mal, dicho jugador retrocede esa cantidad de casillas.
        - *Amarillo*: Avanza si contesta bien. Si contesta mal, no avanza.
        - *Verde*: Avanza directamente la cantidad de casillas indicada por el dado numérico.
        - *Fucsia*: Avanza si contesta bien. Retrocede si contesta mal.
- Cada jugador, a su turno, lanza ambos dados. Hecho esto, se procede a analizar ambos dados para determinar la acción a realizar.
- Si un jugador al avanzar cae en la casilla que está ocupando otro jugador, este jugador regresa al punto de partida. Sin embargo, si un jugador al retroceder cae en la casilla que está ocupando otro jugador, quien regresa al punto de partida es el jugador que venía retrocediendo.
- El tablero se conforma de casillas de distintos tipos:
    - Casillas de *punto de partida*: No pertenecen al tablero en sí, sino que alojan a los jugadores previo al inicio del juego, o bien cuando un jugador es desplazado por otro que cae en la casilla que dicho jugador estaba ocupando.
    - 37 casillas neutrales: No implican ninguna acción adicional.
    - 4 casillas de *pregunta*: El jugador que cae en ella (o bien el jugador siguiente) deberá contestar una pregunta. Si responde bien, avanza los números obtenidos en el dado numérico. Si contesta mal, queda en el mismo lugar. Estas casillas de preguntas deberán distribuirse aleatoriamente antes de cada partida. Deben estar separadas por una distancia mayor de 6, unas de otras. Ej:
        - Casillas 4, 11, 20 y 27, es correcto
        - Casillas 4, 12, 16 y 32, no es correcto

  - Casilla de *llegada*: Es la última casilla. Quien llega a esta casilla, gana la partida. No es necesario obtener el número exacto con el dado numérico para llegar a esta casilla.

- Las preguntas se importarán desde un archivo XML. Son preguntas de tipo *selección múltiple* con las siguientes consideraciones:
  - La pregunta tendrá una *categoría* y un *grado de dificultad* asignado (1 a 5, donde 1 representa la menor complejidad y 5 la mayor complejidad).
  - La pregunta tendrá N opciones (N >= 3), donde una sola de las opciones es la correcta.
    - En el caso de que la pregunta se exhiba a un jugador novato, se le mostrarán la mitad de las N opciones (Si la N / 2 no da un valor entero, redondear la cantidad hacia el número mayor). Dentro de las opciones a mostrar, deberá obviamente incluirse la opción correcta
    - Al jugador experto se le mostrarán todas las opciones
    - Las opciones deben mostrarse en orden aleatorio
  - Los jugadores de nivel novato contestarán preguntas de dificultad 1, 2 y 3 y dispondrán de N * 15 segundos como máximo para contestar.
  - Para jugadores de nivel avanzado, los niveles de dificultad serán 3, 4 y 5 y dispondrán de N * 10 segundos para contestar. Si el jugador no contesta en el tiempo otorgado se considera que la respuesta fue errónea.
  - Durante el juego, una pregunta sólo podrá exhibirse una vez durante la partida: Si una pregunta fue exhibida a un jugador, no puede volver a exhibirse al mismo o a otro jugador.

Sugerencias y comentarios

Diagrama de clases preliminar



Considerar:

- El uso de las clases contenedoras provistas por Java para administrar listas, conjuntos, etc.
- Disponer de una clase **TableroRacer** que centralice el control del juego y contenga listas de casillas, de jugadores, de preguntas, etc.
- Aplicar el lanzamiento de **excepciones / notificación de eventos** en las clases del dominio (Jugador, Casilleros, etc.) con el objetivo de desacoplarlas de la Interfaz de Usuario.

Requerimientos no funcionales:

- Para todos los formularios / ventanas de la aplicación, verificar que soporten adecuadamente cambios de tamaño y diferentes resoluciones de pantalla, utilizando **Alineaciones, Anclajes, Paneles,** etc
- Deberá exhibirse una pantalla de bienvenida al juego.
- Se deberá crear un único *JPanel* reutilizable que permita ingresar los datos de cada jugador.
- Cuando una partida finaliza, deberá darse la posibilidad de iniciar una nueva partida.
- Deberá ser posible finalizar la partida en cualquier momento
- Las clases, atributos, métodos así como la documentación deberán codificarse en idioma Inglés

Condiciones de Aprobación

- Conformar un **grupo** de 2 o 3 personas, de ser posible.
- Implementar la totalidad de la **funcionalidad** solicitada en el enunciado del problema.
- Aplicar indefectiblemente en la solución los siguientes conceptos de la Programación orientada a Objetos: **encapsulamiento**, **polimorfismo**, **herencia**.
- En la Entrega Final:
    - presentar los **archivos fuentes** del proyecto.
    - deberán estar presentes **todos** los integrantes del grupo.
- Otros **conceptos** que serán considerados positivamente en la aprobación del trabajo práctico son:
    - Reutilización adecuada del código.
    - Eficiencia en los algoritmos (ej: búsquedas, ordenamientos)
    - Bajo acoplamiento entre interfaz y lógica de dominio
    - Validaciones de ingresos de datos y consistencia de la información.
    - Código prolijo, claro y correctamente documentado. (ej: nombres representativos, crear variables e instancias necesarias, sobrecargar métodos, uso de *javadoc*)
    - Amigabilidad de las interfaces de usuario

- La **nota** del trabajo práctico es **individual**, basada en la participación en la resolución y defensa del trabajo práctico, y en los conocimientos conceptuales exhibidos en la entrega.