EMULADOR DE TORNEOS PARA JUEGOS DE MESA

Segundo Proyecto de Programación

Colectivo de Programación
www.matcom.uh.cu
Curso 2019-2020

EMULADOR DE TORNEOS PARA JUEGOS DE MESA

Un juego de mesa es una actividad que requiere una mesa para jugar o un soporte similar y donde juegan generalmente un grupo de personas alrededor. Aunque el azar puede ser una parte muy importante en este tipo de juegos, también los hay en los que son necesarios estrategia y razonamiento para poder jugar y en los que el azar no aparece.

Los juegos de mesa se pueden agrupar en distintas categorías según sus características. Generalmente se juegan por turnos en los que cada jugador tiene su momento para decidir qué jugada hacer y luego pasa el turno al jugador siguiente. Algunos deben jugarse en equipos de a 2.

Los juegos como el parchís, involucran dados para determinar aspectos del juego o de la jugada a realizar. Otros como el dominó y el mahjong utilizan fichas marcadas para formar secuencias con determinada regla y los jugadores deben ir desprendiéndose de las piezas hasta quedarse sin ninguna. Los juegos de cartas (naipes) pueden ser los más conocidos. Comúnmente involucran una baraja (francesa o alemana) y ciertas reglas para repartir y utilizar los naipes. Los juegos de tablero como el ajedrez y las damas se juegan sobre una base marcada (tablero) en la que se mueven o colocan las piezas.

JUGADOR

Los jugadores son las entidades que toman las decisiones en un juego. El objetivo de un jugador siempre será ganar o que su equipo gane. Un jugador debe tener un identificador para diferenciarlo de otros jugadores.

Su diseño debe permitir que un mismo jugador pueda participar en varias partidas a la vez (similar a una simultánea). Si un jugador realiza una jugada inválida pierde automáticamente el juego. Debe asegurarse que un jugador no pueda cometer fraude, es decir, alterar directamente el estado del juego, "comunicarse" con otros jugadores, analizar los datos privados de otros, jugar fuera de su turno, tirar varias veces los dados, escoger varias cartas… Usted debe garantizar todo esto haciendo uso correcto de los conceptos claves en el diseño como son el encapsulamiento, la descomposición modular y la abstracción.

Existen dos estrategias triviales que pueden ser concebidas para cualquier tipo de juego y usted deberá proveer de forma genérica.

JUGADOR ALEATORIO

Este jugador juega siempre de forma aleatoria escogiendo una jugada de entre todas las jugadas válidas.

JUGADOR GOLOSO

Este jugador juega utilizando siempre la jugada que parezca "mejor" en cada caso, aunque no tiene que ser la jugada que lleve a una victoria. Por ejemplo en el dominó el juego bota-gorda, en las damas jugar siempre a comer más…

Lo que significa que estos jugadores sean genéricos es que la lógica puede ser concebida con anterioridad a la implementación de un juego específico. No hay por qué tener dos implementaciones de jugador aleatorio en tipos distintos `JugadorAleatorioDeDominó` y `JugadorAleatorioDeDamas`.

JUEGO

El juego es la acción concreta efectuada entre varios jugadores que termina en un puntaje para cada jugador (o equipo). Para ejecutar el juego se debe "iterar" por un enumerable que devuelva las jugadas. Note que el iterador de jugadas debe ser perezoso y permitir ejecutar una por una las jugadas del juego. Esto significa que si no se iteran todas las jugadas el juego no se termina y no se sabría el ganador. Además, el enumerable de jugadas sólo puede recorrerse una única vez puesto que después de "finalizado" el juego no habrían más jugadas por hacer.

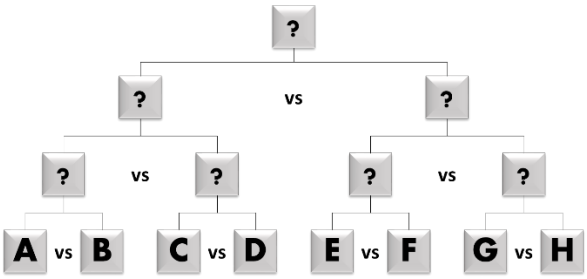
PARTIDA

En algunos juegos una partida es un único juego (por ejemplo ajedrez o damas), en otros como el dominó se necesitan varios juegos para determinar quién gana la partida (aquel equipo que primero sume 100 pts.). Una partida se inicia con un conjunto de jugadores (o equipos) que se mantienen invariantes en todos los juegos. Para ejecutar la partida hay que iterar por los juegos de la misma y ejecutarlos. Si no se termina un juego de la partida y se pide el siguiente juego, se deberá lanzar una excepción de operación no válida.

TORNEO

Un torneo representa un esquema en que ocurre la competición entre varios jugadores. Los jugadores se suscriben al torneo en un principio y queda establecido un conjunto de "encuentros" o partidas. Análogo a la ejecución de una partida a partir de un iterador de juegos, y a la ejecución de un juego a través de un iterador de jugadas, un torneo se ejecuta a través de un iterador de sus partidas.

Uno puede pasar a la siguiente partida si los jugadores de esta no dependen de una partida todavía sin finalizar. Por ejemplo, en un torneo de eliminatoria como el de la figura (con 8 jugadores y 7 encuentros), las partidas de la base pueden ser ejecutadas en cualquier orden, pero para iterar por las partidas del nivel siguiente tienen que haber concluido las partidas de las que estas dependen para saber sus jugadores.



La organización de las partidas en un torneo puede ser por eliminatoria, un todos contra todos o por calificación. A continuación veremos varios tipos de torneos que se pueden organizar.

CALIFICACIÓN INDIVIDUAL

En este tipo de competición cada jugador participa en solitario y obtiene una puntuación. Al final se hace un escalafón para ordenar los participantes y determinar los lugares. Note que no es posible realizar este tipo de torneo con cualquier juego.

TÍTULO

En esta organización existe un campeón y los demás jugadores retan al actual campeón. Si gana, obtiene el título de campeón hasta que es vencido por otro jugador. Al final todos los jugadores terminan con puntuación 0 exceptuando al campeón que concluye con 1.

Dos a dos

En este tipo de competición todos los jugadores se enfrentan entre sí. Cada partida termina con un ganador y un perdedor o un empate (en caso que el tipo de juego lo permita). Las victorias, derrotas o empates dan una puntuación que se acumula y permite tener al final una puntuación para cada jugador (o equipo).

Otros tipos de organización que existen son el sistema suizo y el sistema de eliminatorias. No obstante, aunque usted no tiene que implementar estos dos últimos sistemas, su diseño debería ser lo suficientemente flexible como para permitir incorporarlos.

Sobre su solución

El proyecto deberá entregarse en dos etapas. En una primera etapa implementará la plataforma para la realización de torneos y una implementación concreta del juego Tic-tac-toe como juego de ejemplo. La plataforma no debe tener ninguna implementación de juego particular, sólo los tipos de torneos y los conceptos abstractos para las futuras definiciones de tipos de juego, jugadores, etc. Para la segunda etapa de entrega se le pedirá que implemente un segundo juego SIN MODIFICAR la plataforma.

NOTA: Las fechas de entrega serán informadas en la Semana 8.

Interfaz gráfica

En la segunda etapa deberá entregar una implementación de una interfaz gráfica que permita configurar un torneo y ejecutarlo. Al final se deben mostrar los resultados del torneo.

La aplicación debe funcionar en forma de Wizard. En la primera pantalla deberá aparecer un listado con los juegos implementados (al menos el tic-tac-toe y el segundo juego) para seleccionar de qué juego se hará el torneo. Luego un listado de todos los tipos de jugadores implementados que pueden jugar dicho juego, incluyendo el jugador aleatorio y el goloso. En esta vista se podrán escoger los jugadores que participarán en el torneo y nombrarlos (Un mismo tipo de jugador puede ser escogido varias veces siempre y cuando se le cambie el nombre, por ejemplo: random1, random2, goloso1, goloso2). En la siguiente vista se seleccionará el tipo de torneo y permitirá ejecutar el torneo. Una última vista muestra los resultados del torneo.

La ejecución del torneo puede realizarse por pasos o ser ejecutado sin pausas hasta el final. Para ello se deberán mostrar las siguientes opciones:

- Ejecutar torneo. Manda a ejecutar todas las partidas restantes del torneo.
- Ejecutar partida actual. Ejecuta hasta el final de la partida actual.
- Ejecutar juego actual. Ejecuta hasta el final del juego actual.
- Ejecutar jugada. Ejecuta una única jugada en el juego actual solamente.
- Ejecutar automáticamente. Ejecuta jugada por jugada de forma automática con un tiempo de espera entre jugadas personalizable.

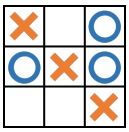
Bitácora (Log)

En la vista de ejecución del torneo se deberá mostrar un cuadro de texto con notificaciones por cada suceso ocurrido en la competición: comienzo del torneo, comienzos de partida, comienzos de juego, jugadas, finalización de juegos, partidas y el fin del torneo.

Debe existir un filtro que permita escoger qué tipos de notificaciones se desean mostrar. Por ejemplo, a lo mejor se desea desactivar “mostrar jugadas” porque solo interesan las partidas y los resultados de las mismas.

Descripción de algunos de los juegos posibles a implementar

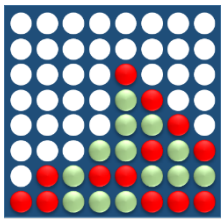
A continuación se muestran algunos de los juegos de mesa más conocidos y que deben poder implementarse en su plataforma sin mucha dificultad.



Tic-tac-toe

Un tablero de 3x3 casillas. Cada jugador en su turno pone en una casilla vacía una marca. Si logra poner 3 marcas en línea gana.

4 en Línea



Un tablero de 8x8 casillas. Cada jugador en su turno escoge una columna no llena y puede colocar en esta una pieza de su color. La pieza cae hasta tocar el fondo (fila más abajo o hasta chocar con una celda ocupada). El primero que ubique cuatro piezas de su color en línea (horizontal, diagonal o vertical) gana.

Othello

Un tablero de 8x8 casillas. Cada jugador en su turno escoge una celda vacía para jugar. Una vez que se pone una pieza, toda pieza contraria alineada entre dicha pieza y otra pieza del mismo color se convierte al color del jugador. Sólo pueden colocarse piezas en lugares en los que al menos se logre virar una pieza. Si no existen posibles jugadas el jugador se "pasa". Gana el jugador que más piezas tenga al final.

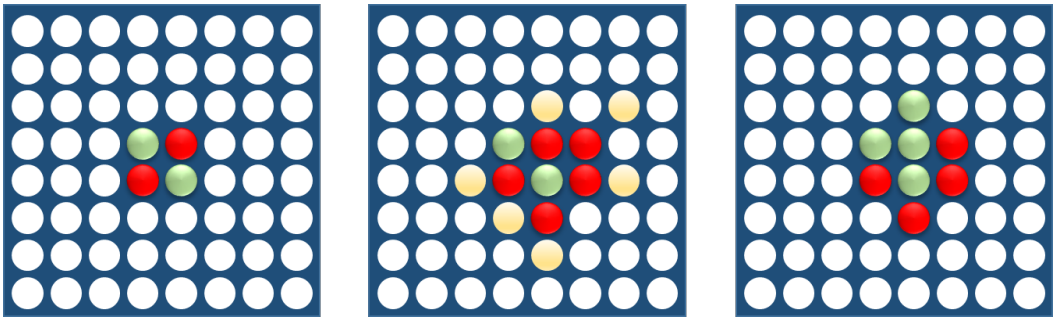
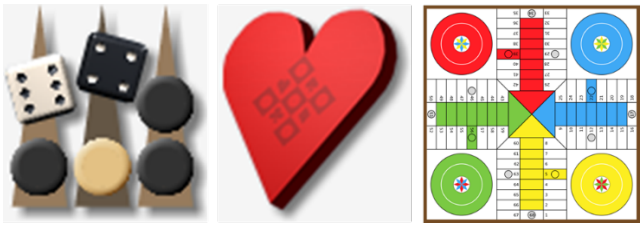


FIGURA 1. IZQUIERDA, TABLERO INICIAL. CENTRO, POSIBLES JUGADAS PARA EL JUGADOR VERDE. DERECHA, TABLERO RESULTANTE DE ESCOGER UNA DE ELLAS.

Ajedrez, Scrabble, Damas, Dominó

Backgammon, Corazones, Parchís



SOBRE LEGIBILIDAD DEL CÓDIGO.

IDENTIFICADORES

Todos los identificadores (nombres de variables, métodos, clases, etc) deben ser establecidos cuidadosamente, con el objetivo de que una persona distinta del programador original pueda comprender fácilmente para qué se emplea cada uno.

Los nombres de las variables deben indicar con la mayor exactitud posible la información que se almacena en ellas. Por ejemplo, si en una variable se almacena "la cantidad de obstáculos que se ha encontrado hasta el momento", su nombre debería ser `cantidadObstaculosEncontrados` o `cantObstaculos` si el primero le parece demasiado largo, pero nunca ~~`cOb`~~, ~~`aux`~~, ~~`temp`~~, ~~`miVariable`~~, ~~`juan`~~, ~~`contador`~~, ~~`contando`~~ o ~~`paraQueNoChoque`~~. Note que el último identificador incorrecto es perfectamente legible, pero no indica "qué información se guarda en la variable", sino quizás "para qué utilizo la información que almaceno ahí", lo cual tampoco es lo deseado.

- Entre un nombre de variable un poco largo y descriptivo y uno que no pueda ser fácilmente comprensible por cualquiera, es preferible el largo.
- Como regla general, los nombres de variables **no** deben ser palabras o frases que indiquen acciones, como ~~eliminando~~, ~~saltar~~ o ~~parar.~~

Existen algunos (muy pocos casos) en que se pueden emplear identificadores no tan descriptivos para las variables. Se trata generalmente de pequeños fragmentos de código muy comunes que "todo el mundo sabe para qué son". Por ejemplo:

```
int temp = a;
a = b;
b = temp;
```

"Todo el mundo" sabe que el código anterior constituye un intercambio o *swap* entre los valores de las variables a y b, así como que la variable `temp` se emplea para almacenar por un instante uno de los dos valores. En casi cualquier otro contexto, utilizar `temp` como nombre de variable resulta incorrecto, ya que solo indica que se empleará para almacenar "temporalmente" un valor y en definitiva todas las variables se utilizan para eso.

Como segundo ejemplo, si se quiere ejecutar algo diez veces, se puede hacer

```
for (int i = 0; i < 10; i++)
  ...
```

en lugar de

```
for (int iteracionActual = 0; iteracionActual < 10; iteracionActual++)
  ...
```

Para los nombres de las propiedades (*properties* en inglés) se aplica el mismo principio que para las variables, o sea, expresar "qué devuelven" o "qué representan", solo que los identificadores deben comenzar por mayúsculas. No deben ser frases que denoten acciones, abreviaturas incomprensibles, etc.

Los nombres de los métodos deben reflejar "qué hace el método" y generalmente es una buena idea utilizar para ello un verbo en infinitivo o imperativo: Agregar, Eliminar, ConcatenarArrays, ContarPalabras, Arranca, Para, etc.

En el caso de las clases, obviamente, también se espera que sus identificadores dejen claro qué representa la clase: Jugador, Partida, Torneo, Programa.

COMENTARIOS

Los comentarios también son un elemento esencial en la comprensión del código por una persona que lo necesite adaptar o arreglar y que no necesariamente fue quien lo programó o no lo hizo recientemente. Al incluir comentarios en su código, tome en cuenta que no van dirigidos solo a Ud., sino a cualquier programador. Por ejemplo, a lo mejor a Ud. le basta con el siguiente comentario para entender qué hace determinado fragmento de código o para qué se emplea una variable:

// lo que se me ocurrió aquel día

Evidentemente, a otra persona no le resultarán muy útiles esos comentarios.

Algunas recomendaciones sobre dónde incluir comentarios

- Al declarar una variable, si incluso empleando un buen nombre para ella pueden quedar dudas sobre la información que almacena o la forma en que se utiliza
- Prácticamente en la definición de todos los métodos para indicar qué hacen, las características de los parámetros que reciben y el resultado que devuelven
- En el interior de los métodos que no sean demasiado breves, para indicar qué hace cada parte del método

Es cierto que siempre resulta difícil determinar dentro del código qué es lo obvio y qué es lo que requiere ser comentado, especialmente para Ud. que probablemente no tiene mucha experiencia programando y trabajando con código hecho por otras personas. Es preferible entonces que "por si acaso" comente su código lo más posible.

Otro aspecto a tener en cuenta es que los comentarios son fragmentos de texto en lenguaje natural, en los cuales deberá expresarse lo más claramente posible, cuidando la ortografía, gramática, coherencia, y demás elementos indispensables para escribir correctamente.

Todos estos elementos son importantes para la calidad de todo el código que produzca a lo largo de su carrera, pero además **tendrán un peso considerable en la evaluación de su proyecto de programación**