Grupo Weboo
Universidad de La Habana

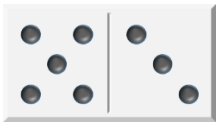
Jugando al Dominó

Examen Extraordinario de Programación

Curso 2016-2017

NOTA: Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios no se guarden. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

Sobre una mesa se han sentado a jugar N personas un juego de dominó. A cada persona se le reparte un conjunto de fichas. Cada ficha tiene dos números, uno a la izquierda y otro a la derecha.

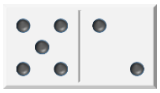


Ficha [5|3]

Los jugadores se enumeran del 0 al $N-1$. El juego comienza por el jugador 0, siguiéndole el 1 y así sucesivamente hasta que vuelve el turno al jugador 0 nuevamente.

Al principio el jugador 0 puede colocar sobre la mesa cualquiera de sus fichas. Se colocará una única ficha por turno. Si un jugador tiene el turno y existen fichas en la mesa, éste deberá jugar alguna ficha que tenga con un número igual al extremo izquierdo o al extremo derecho del tablero. La ficha se coloca haciendo corresponder la cara que tiene igual número que el extremo del tablero con éste. Por ejemplo, dada la configuración siguiente:

Si se decide jugar la ficha



La misma debería ser jugada por el extremo izquierdo y girada de la forma

Quedando el tablero con la configuración:

Note que no basta con especificar únicamente qué ficha se desea jugar, puesto que la ficha [1|2] podría jugarse en este caso por cualquiera de los dos extremos, obteniéndose dos configuraciones distintas.

Si un jugador no posee ninguna ficha con algún número igual a los extremos del tablero (“se pasa”), pierde el turno y se continúa con el siguiente. Si se pasan de forma seguida los N jugadores, entonces el juego termina (“se tranca”) y ganan los que tengan la menor puntuación. La puntuación está dada por la suma de los números en las fichas que se quedaron sin poner. Si algún jugador logra ubicar todas sus fichas (“se pega”), también se termina el juego y éste se proclama único vencedor (aun cuando pudiera alguien tener 0 puntos por poseer únicamente la ficha [0|0]).

En este juego existen distintas estrategias que se pueden aplicar. Algunas de las más conocidas son el “bota gorda” intentando jugar siempre la ficha que mayor puntuación tenga, y el “fallo” que intenta conservar la mayor variedad de valores distintos en la data del jugador.

Usted deberá implementar la lógica del juego de dominó que permita simular una partida en la que se han configurado varios jugadores con distintas estrategias.

Para ello debe implementar el tipo `JuegoDeDomino` que implementa la interfaz `IJuegoDeDomino`. Esta interfaz está definida en el ensamblado `Weboo.Domino.dll` y luce como se muestra a continuación:

```
/// <summary>
/// Representa la lógica de un árbitro para un juego de dominó
/// </summary>
public interface IJuegoDeDomino {
    /// <summary>
    /// Se invoca inicialmente para determinar la cantidad de jugadores que podrán jugar
    /// </summary>
    void IniciaJuego(int cantidadDeJugadores);
    /// <summary>
    /// Devuelve la cantidad de jugadores con que se inició el juego.
    /// </summary>
    int CantidadDeJugadores { get; }
    /// <summary>
    /// Se invoca al inicio, una vez por cada uno de los jugadores
    /// determinando la estrategia que utilizará y las fichas iniciales.
    /// La estrategia puede ser el mismo objeto para varios jugadores.
    /// </summary>
    void CreaJugador(IEstrategia estrategia, IEnumerable<Ficha> fichasIniciales);
    /// <summary>
    /// Devuelve el número del jugador actual al que le toca jugar
    /// (el primer jugador es el número 0).
    /// </summary>
    int JugadorActual { get; }
    /// <summary>
    /// Devuelve el estado actual de las fichas dispuestas en la mesa (de izquierda a derecha)
    /// </summary>
    IEnumerable<Ficha> Tablero { get; }
    /// <summary>
    /// Ordena al jugador actual a que juegue según su estrategia y actualiza el tablero
    /// (en caso que éste posea una posible ficha), luego pasa el turno al siguiente jugador.
    /// </summary>
    void Juega();
    /// <summary>
    /// Devuelve un enumerable de las fichas que posee determinado jugador.
    /// </summary>
    IEnumerable<Ficha> FichasDe(int jugador);
    /// <summary>
    /// Determina si el juego llegó a su final.
    /// </summary>
    bool JuegoTerminado { get; }
    /// <summary>
    /// Enumerable que devuelve los índices de los jugadores ganadores.
    /// En caso de un tranque, todos los que tengan el mínimo de puntos.
    /// En caso de un pegao', un único valor con el índice de dicho jugador.
    /// En caso de no haberse terminado el juego, un enumerable vacío.
    /// </summary>
    IEnumerable<int> Ganadores { get; }
}
```

La estructura `Ficha` permite representar una pieza de dominó. Note que dos fichas son la misma si tienen el mismo par de números ([x|y] es la misma ficha que [y|x]). Las fichas tienen implementados los métodos `Equals` y `GetHashCode` apropiadamente.

Una estrategia (objeto de tipo `IEstrategia`) permite determinar qué ficha de la data de un jugador deberá ser jugada en cada turno. Para ello se le debe invocar el único método `DimeJugada` pasándole el conjunto de fichas del jugador y el estado actual del tablero.

El enumerable de las fichas del tablero determinan las piezas que se han puesto en la mesa. El primer elemento del enumerable representa la ficha del extremo izquierdo y el último elemento, qué ficha se ubica en el extremo derecho.

A su lógica se le invocarán las funcionalidades en el orden adecuado. El siguiente código muestra un caso de uso válido para el tipo `JuegoDeDomino`.

```
JuegoDeDomino juego = new JuegoDeDomino();

var estrategia = new LaPrimeraQueMeEncuentre();

juego.IniciaJuego(4);

juego.CreaJugador(estrategia, new Ficha[] { new Ficha(2, 4), new Ficha(1, 2), new Ficha(1, 4) });
juego.CreaJugador(estrategia, new Ficha[] { new Ficha(2, 3), new Ficha(2, 2), new Ficha(0, 1) });
juego.CreaJugador(estrategia, new Ficha[] { new Ficha(1, 3), new Ficha(1, 1), new Ficha(0, 4) });
juego.CreaJugador(estrategia, new Ficha[] { new Ficha(0, 2), new Ficha(4, 4), new Ficha(3, 3) });

while (!juego.JuegoTerminado)
    juego.Juega();

Console.WriteLine("Juego terminado");
Console.WriteLine("Ganador(es) " + string.Join(", ", juego.Ganadores));
```

A diferencia de los juegos tradicionales de dominó, en nuestro caso:

- La cantidad de jugadores puede ser cualquiera (siempre mayor que 1).
- Las fichas repartidas se pueden repetir (aunque siempre tienen valores no negativos).
- Las cantidades de fichas repartidas inicialmente por jugador puede ser diferentes (pero siempre se reparte al menos una ficha).

En este ejemplo las jugadas son (teniendo en cuenta que la estrategia es poner la primera de izquierda a derecha que sea válida):
Jugador 0 → **[2|4]**
Jugador 1 → **[3|2]** [2|4]
Jugador 2 → **[1|3]** [3|2] [2|4]
Jugador 3 → [1|3] [3|2] [2|4] **[4|4]**
Jugador 0 → **[2|1]** [1|3] [3|2] [2|4] [4|4]
Jugador 1 → **[2|2]** [2|1] [1|3] [3|2] [2|4] [4|4]
Jugador 2 → [2|2] [2|1] [1|3] [3|2] [2|4] [4|4] **[4|0]**
Jugador 3 → **[0|2]** [2|2] [2|1] [1|3] [3|2] [2|4] [4|4] [4|0]
Jugador 0 → se pasa
Jugador 1 → **[1|0]** [0|2] [2|2] [2|1] [1|3] [3|2] [2|4] [4|4] [4|0] (se pega)

NOTA: Los casos de prueba que aparecen en este proyecto son solamente de ejemplo. Que usted obtenga resultados correctos con estos casos no es garantía de que su solución sea correcta y de buenos resultados con otros ejemplos. De modo que usted debe probar con todos los casos que considere convenientes para comprobar la validez de su implementación.