DESCRIPCIÓN BREVE

En esta primera entrega, se presenta el diagrama de clases inicial para el planteamiento del juego, y junto con una especificación de las entidades, atributos y los métodos de cada una. En cada uno de los métodos, se especifican los parámetros de entrada y el tipo de dato, la función de cada método así como el valor que retorna.

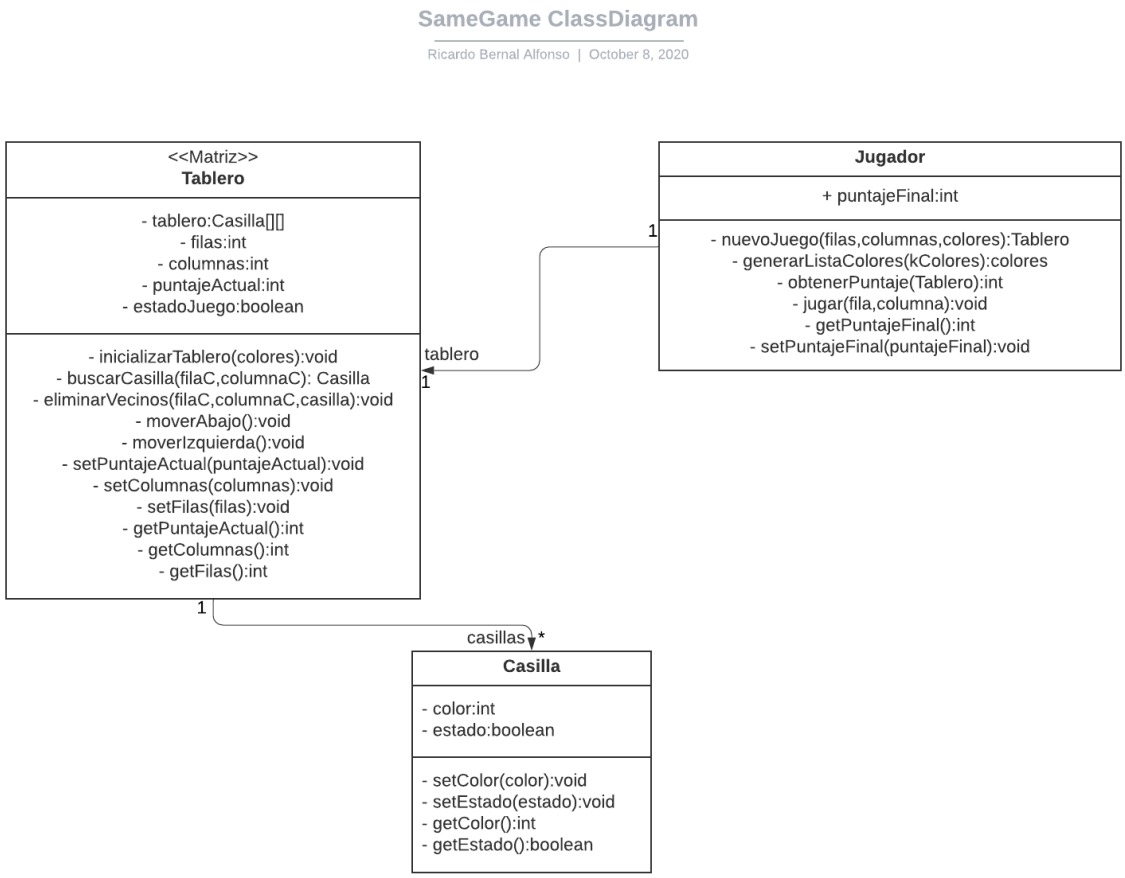
Ricardo Bernal Alfonso – Diego Mauricio Bulla Nuñez

Análisis de Algoritmos – Pontificia Universidad Javeriana

jueves, 8 de octubre de 2020

Tabla de ilustraciones

Diagrama de clases



Descripción general del juego

SameGame es un juego para una persona que utiliza una cuadrícula de dimensiones m × n, donde m y n pueden variar entre 2 y 50, y cada bloque o casilla está pintada de un color escogido de una lista de colores.

Se desarrolló un diagrama de clases en el cual se representan las entidades que componen el juego, así como sus atributos o características de cada, los métodos que cada una de ellas puede ejecutar y los valores de precondición y post-condición para dichos métodos.

- Jugador: La entidad principal será el Jugador, este será el responsable de crear o generar un nuevo tablero y su único atributo será el puntaje obtenido al finalizar cada juego (ya sea que completo dicho juego o cometió alguna falta). Para ello disponemos de los siguientes métodos:
    - *void nuevoJuego(int columnas, int filas, List<int> colores)*: Este método inicia un nuevo juego al crear una nueva instancia de Tablero. Para ello, como datos de entrada recibirá dimensiones del tablero (número de filas, número de columnas) y la lista de colores con los cuales se va colorear cada uno de las casillas que conformaran el nuevo tablero de juego. Una vez inicializa el tablero con la función inicializarTablero(colores), se llama al método jugar() al final de esta.
    - *int obtenerPuntaje(Tablero tablero)*: Permite obtener el ultimo puntaje acumulado en la última jugada ejecutada con getPuntajeActual() y lo actualizamos con setPuntajeFinal().
    - *void jugar()*: Este método es llamado por el método nuevoJuego(). Este método es el encargado de invocar los métodos para seleccionar una casilla con el metodoBuscarCasilla() de la clase Tablero (la fila y la columna se genera de forma aleatoria pero su valor debe estar dentro de las dimensiones especificadas del tablero). Una vez obtenida la casilla, se verifica si 'existe' y si es así busca y elimina los vecinos con posean el mismo color con el método eliminarVecinos(). Posteriormente, calcula el puntaje obtenido al eliminar dichos vecinos (esto se realiza dentro del método eliminarVecinos()), actualiza el puntaje actual y mueve las casillas de forma vertical y horizontal (si es posible) con los métodos moverAbajo() y moverIzquierda(). Lo descrito anteriormente se ejecuta en una iteración y cada una de estas supone una jugada. Esto se realiza mientras el juego no haya terminado (estadoJuego = true).
- Tablero: La entidad principal será el Tablero, esta ejecutará las distintas acciones sobre el tablero como inicializar las casillas (pintar las casillas con los respectivos colores), buscar la casilla selecciona por el Jugador y eliminar los vecinos semejantes de dicha casilla si tiene, redistribuir las casillas después de cada acción donde se eliminen casillas y calcular el puntaje en cada acción.
    - *void inicializarTablero(List<int> colores)*: Una vez creado el tablero con sus respectivas dimensiones, se invoca este método para crear una casilla con un color en cada una de las posiciones del tablero. El color de cada casilla es elegido de forma aleatoria de la lista 'colores' que ingresa como parámetro de entrada.
    - *Casilla buscarCasilla(int filaC, int columnaC)*: Dada unas coordenadas, encontramos una casilla en dicha posición. Si la casilla no existe, es decir, que su atributo 'estado' es igual a 'false', se retorna null y la jugada acaba, es decir, la iteración se da por terminada.
    - *void eliminarVecinos(int filaC, int columnaC, Casilla casilla)*: Este método se encarga de eliminar los vecinos que compartan el mismo color del objeto 'casilla' que llega por parámetro. Para ello, se cambia el atributo 'estado' a 'false' para las casillas que son eliminadas.

      Una vez que son eliminadas las casillas, se calcula el puntaje obtenido en esta jugada y se suma al puntaje actual (puntajeActual).

- *void moverVertical()*: Este método mueve las casillas existentes hacia abajo si efectuó alguna eliminación de casillas. Si se eliminaron casillas del tablero se recorre cada una de las columnas del tablero y por cada columna, verificamos cuales casillas se encuentran vacías, es decir, que el 'estado' de dicha casilla es igual a false. Las casillas que "no existen", son intercambiadas (swap) de posición con las casillas que si "existen" y se encuentran arriba.
- *void moverIzquierda()*: Este método mueve las columnas hacia la izquierda si existe al menos 1 columna vacía. Para ello, se recorre cada columna y en cada se revisa el 'estado' de la última casilla (de arriba abajo): si la casilla no existe, entonces dicha columna no existe por lo que las columnas delante de ella se mueven hacia la izquierda.

- Casilla: Esta entidad se utiliza para denotar las casillas que van a conformar el tablero del juego, así mismo esta entidad tiene los atributos color que se utiliza para identificar qué color representa, este atributo es usado para determinar cuáles son sus vecinos. El atributo 'estado' permite diferenciar si una casilla ha sido eliminada.