Tarea Sesión 2. Juego 2048

1 Preliminares

Este ejercicio debe estar implementado 24 horas antes de la siguiente clase de laboratorio.

Renombra el proyecto 2048 implementado en la sesión 1 como:

apellido1_apellido2_nombre_lab02_task_game2048

Renombra también los paquetes

uo.mp.lab02…..

2 Objetivo

Se trata de implementar las operaciones finales que podrá hacer el usuario sobre el tablero: mover a la derecha, a la izquierda, arriba y abajo. Estas operaciones usarán las que ya se han hecho de compactar a la derecha, izquierda, arriba y abajo y añadirán la operación de suma de números iguales consecutivos.

Al realizar las pruebas se incluirá en cada caso de uso el comentario formado por GIVEN… WHEN… THEN.

3 Descripción

En la tarea Lab01 se realizaron los movimientos de compactación. En esta segunda sesión se deberá realizar la operación completa de mover, que consiste en compactar, sumar valores y volver a compactar.

Añade a la clase Game2048 los siguientes métodos:

1. Método ***public void moveRight()*** compacta a la derecha los números de cada fila y a continuación suma de dos en dos los números consecutivos iguales dejando el resultado en la casilla derecha. Finalmente, se vuelve a compactar.

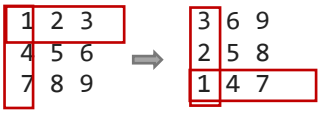
   Ejemplo:
   Dada una fila 2 2 0 2 2 el resultado de moveRight() sería: 0 0 0 4 4

   2 2 0 2 2 -> compacta a la derecha -> 0 2 2 2 2
   0 2 2 2 2 -> suma consecutivos -> 0 0 4 0 4
   0 0 4 0 4 -> compacta a la derecha -> 0 0 0 4 4

2. Método ***public void moveLeft()*** Análogo a moveRight pero a la izquierda. Para realizar esta operación, en lugar de usar el método de mover a la izquierda se pueden hacer rotaciones del tablero y se reutiliza siempre el movimiento a la derecha. En este caso sería:

   ```
   rotar, rotar, mover a la derecha, rotar, rotar
   ```

   Para rotar el tablero, se debe rotar a la izquierda (contrario a las agujas del reloj)

   Crea un método privado ***private void rotateBoard()*** para esta operación.

3. Método ***public void moveUp().*** Similar al anterior.

4. Método ***public void moveDown().*** Similar al anterior.

5. Método ***public boolean isFinished().*** Este método comprueba si se da la condición para terminar el juego y ganar, es decir, si alguna celda contiene un 2048 o bien ya no se puede seguir jugando porque el tablero está lleno, devolviendo verdadero. En caso contrario, devuelve falso.

   Se debe sustituir la llamada a ***isBoardFull*** por ***isFinished*** en el método ***play()*** de la clase ***GameApp.*** Con este cambio el método ***isBoardFull()*** dejará de ser público y pasará a ser privado y se deben sustituir sus tests por los del método ***isFinished()***.

   Se debe sustituir también en el método ***doMovement()*** de la clase ***GameApp*** las llamadas `compactXXX()` por `moveXXX()`

6. **Añade** al folder test, **las pruebas de los métodos**: moveRight(), moveLeft(), moveUp(), moveDown() y isFinished()

7. **Refactorización**. Observa que todos los **métodos de compactar** excepto el de compactar a la derecha ya no se usan en esta versión. Por lo tanto, **debes eliminarlos**, **así como los tests** correspondientes. Debes **cambiar** además el

modificador de acceso del método compactar a la derecha, elimínalo para que quede como modificador de paquete (pues solo se usa dentro de Game2048 y en los test que están en el mismo paquete).

8. **Genera la documentación** con JavaDoc. El código del proyecto debe incluir, comentarios de documentación y comentarios de implementación.

Nota. Sustituye el fichero ForTesting.java por el nuevo que se proporciona con el material de esta sesión L02. Si no quieres usar el nuevo ForTesting podrías crear internamente tus propias matrices como se indica en la siguiente figura:

```
@Test
public void testMoveLeftHalfMatrix() {

    int[][] initial = { { 2, 0, 2 }, { 0, 2, 2 }, { 2, 2, 0 } };
    int[][] expected = { { 4, 0, 0 }, { 4, 0, 0 }, {4, 0, 0 } };
    game = new Game2048(initial);
    game.moveLeft();
    assertArrayEquals(expected, game.getBoard());
}
```