Tarea Sesión 1 . Juego 2048

1 Preliminares

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

- Carga el proyecto: student_lab01_game2048. Contiene el esqueleto inicial que será el punto de partida.
- Renombra el proyecto dentro del Eclipse (usando `Refactor- Rename`) como sigue: **apellido1_apellido2_nombre_lab01_task_game2048**, en letras minúsculas y sin acentos. Por ejemplo, un estudiante llamado Pablo Peláez Nuño quedaría renombrado como:
  pelaez_nuno_pablo_lab01_task_game2048)
- Para entregar la tarea, exporta este proyecto `game2048` y el proyecto `util`, comprímelos en **un solo fichero** formato **ZIP** y súbelo al campus virtual.
- **Otra forma de crear un fichero ZIP** (seleccionar proyecto(s) / botón derecho / Enviar a / carpeta comprimida (en zip)).

2 Introducción

2048 es un juego en línea creado en 2014 por Gabriele Cirulli (con 19 años), cuyo objetivo es mover los números del tablero para combinarlos hasta conseguir en una celda del tablero el número 2048, 4096, … Se puede probar en: http://juego2048.es/.

Consta de un tablero cuadrado en el que partiendo de un 2 se trata de sumar números iguales. El usuario realiza movimientos con las flechas: derecha, izquierda, arriba o abajo para mover los números.

Cada movimiento del usuario, desplaza los valores según dirección de la flecha, suma si hay dos consecutivos iguales y añade un nuevo 2 al tablero. En el juego original también puede añadir un 4 con una frecuencia mucho menor. Este hecho no lo tendremos aquí en cuenta. El juego consiste en combinarlos para incrementar la suma al máximo posible (2, 4, 8, 16, 32…2048, 4096..), y conseguir la mayor puntuación posible. La partida termina cuando se llena el tablero y no es posible realizar más sumas.

3 Primera parte. Compactación

Este proyecto se va a implementar en las dos primeras sesiones (dos semanas).

En esta primera sesión se deberá completar el proyecto proporcionado (student_lab01_game2048) realizando solo desplazamientos, o lo que es lo mismo, compactación. Es decir, **hacer que queden consecutivos todos los 0 del lado contrario al que se pulse la flecha**.

Por ejemplo, si se pulsa flecha a la derecha, todas las filas quedarán con todos sus ceros consecutivos a la izquierda y los valores diferentes de cero consecutivos a la derecha y en el mismo orden en el que estaban antes de pulsar la flecha y compactar. Si se pulsa flecha arriba, serán las columnas las que se compacten quedando los ceros compactados en las últimas posiciones de la columna y los valores diferentes de cero en las primeras posiciones en el mismo orden en el que estaban.

Además, hay que tener en cuenta que cada vez que se realiza un nuevo movimiento por parte del usuario, **el juego añade un 2 en una posición aleatoria del tablero**.

Por ejemplo, al comienzo de una nueva partida, la interfaz de la aplicación puede mostrar los siguientes valores (al inicio siempre hay un 2 en una posición aleatoria).

```
JUEGO 2048
0 0 0
2 0 0
0 0 0

Mueve los números en una dirección [r R]/[l L]/[u U]/[d D]:
```

A continuación, se debe introducir un carácter para indicar el movimiento de los números: [r R] Right, [l L] Left, [u U] Up, [d D] Down. Si pulsamos la tecla R el tablero quedaría:

```
2 0 0
0 0 2
0 0 0

Mueve los números en una dirección [r R]/[l L]/[u U]/[d D]:
```

Se ha movido el 2 que estaba en la posición (1,0) a la posición (1,2) (por pulsar la tecla R) y además aparece un nuevo 2 en la posición aleatoria (0,0).

Si pulsamos de nuevo la tecla R el tablero quedaría:

```
0 0 2
0 0 2
0 0 2

Mueve los números en una dirección [r R]/[l L]/[u U]/[d D]:
```

En este caso el 2 aleatorio se colocó en la posición (2, 2).

Si pulsamos de nuevo la tecla L el tablero quedaría:

```
2 0 2
2 0 0
2 0 0

Mueve los números en una dirección [r R]/[l L]/[u U]/[d D]:
```

Aquí el 2 aleatorio aparece en la posición (0,2)

Si pulsamos de nuevo la tecla L el tablero quedaría:

```
2 2 0
2 0 0
2 0 2

Mueve los números en una dirección [r R]/[l L]/[u U]/[d D]:
```

En este caso el nuevo 2 aparece en la posición (2,2).

4 Objetivo

Se trata de implementar las operaciones de creación del tablero y compactación. Se tomará el esqueleto recibido como proyecto de partida. El juego se implementa en la clase **Game2048**.

5 Fases del trabajo

Es muy importante que realices la tarea en el orden que se te indica.

5.1 PRIMERO. Implementar todas las pruebas

En la carpeta test del proyecto, están todas las clases de prueba y éstas contienen todos los métodos de prueba que hay que implementar.

¡Atención! **Ya están implementados** los métodos de test para *compactLeft* y para *isBoardFull*. Analiza su implementación y será una guía para implementar el resto.

La clase `ForTesting` contiene código útil para implementar todos los test. Se incluye un método estático `getSum` que puede ser usado para las pruebas de los métodos `next` y `restart`

5.2 SEGUNDO. Implementar los métodos solicitados en la clase Game2048
Para ello, se dan algunas indicaciones en el siguiente apartado.

6 Clase Game2048

La clase **Game2048** debe incluir las constantes necesarias (el tamaño mínimo permitido del tablero será 3X3 y el máximo 5x5) y la propiedad board como matriz de números enteros, con los siguientes métodos públicos:

1. Un **constructor sin parámetros** ***`public Game2048()`*** para crear un tablero de dimensión por defecto 3x3.

2. Un **constructor con un parámetro** ***`public Game2048 (int size)`*** que creará un tablero con filas y columnas igual al tamaño recibido. **En caso de recibir un tamaño incorrecto, se creará un tablero de tamaño por defecto 3x3.**

3. Un **constructor con un parámetro** ***`public Game2048 (int[][] board)`*** que recibe un tablero. Si el parámetro board es null o el tablero tiene un tamaño no válido o las filas del tablero tienen distinto tamaño se deberá lanzar una excepción `IllegalArgumentException` con el mensaje “Tablero inadecuado o null”. Es importante que, al recibir el tablero, éste no se asigne directamente al atributo board que tiene la clase, sino que **se haga una copia (se denomina copia defensiva)** para que desde fuera no se pueda modificar el tablero del juego.

4. Método ***`public int[][] getBoard()`***. Devuelve **una copia** de la matriz para que, desde la interfaz, pueda ser mostrada como se desee. También será utilizada en las pruebas.

5. Método ***`public void restart()`***. Permite restablecer el tablero asignando a todas las casillas un cero (se vacía el tablero) salvo una casilla que de manera aleatoria se le asigna un 2 para añadir el primer valor al tablero. Reutiliza el código siempre que puedas.

6. Método ***`public void next()`***. Añade un nuevo número 2 en una posición aleatoria del tablero que esté vacía (con valor 0). En caso de que no haya posiciones libres, el método no hace nada.

7. Método ***`public boolean isBoardFull()`***. Comprueba si el tablero está lleno. Esto ocurre cuando todas las celdas o posiciones del tablero tienen un número distinto de cero.

8. Método ***public void compactRight().*** Desplaza todos los valores positivos a la derecha en cada fila, dejando los huecos (valores = 0) a la izquierda.

9. Método ***public void compactLeft().*** Desplaza todos los valores positivos a la izquierda en cada fila, dejando los huecos (valores = 0) a la derecha.

10. Método ***public void compactUp().*** Desplaza todos los valores positivos hacia arriba en cada fila, dejando los huecos (valores = 0) abajo.

11. Método ***public void compactDown().*** Desplaza todos los valores positivos abajo en cada fila, dejando los huecos (valores = 0) arriba.

12. Método ***public String toString()***. Devuelve un String con los datos de la matriz en el formato especificado para ser mostrado por pantalla. Habrá 5 espacios para cada posición en la misma fila y un \n al final de cada fila.

    Ejemplo. Si el tablero contiene:

    2 2 0
    2 0 0
    2 0 2

    Se devuelve el String: "2 2 0\n2 0 0\n2 0 2\n"

13. **Generar la documentación con JavaDoc**. El código del proyecto debe incluir comentarios de documentación y comentarios de implementación.

Pseudocódigo de cómo compactar a la izquierda una fila (todos los ceros a la derecha)

Recorrer fila

- Para cada posición.
  - Si está vacía (valor 0)
    - Buscar primera posición a la derecha de ésta que no esté vacía (*submétodo que devuelve la posición*)
    - Si la encuentro
      - Intercambiar los valores de posiciones
    - Si no hay ninguna
      - Ya acabé, ya está compactado
  - Si no está vacía continuar con la siguiente