UNIVERSIDAD NACIONAL DE GENERAL SARMIENTO

LICENCIATURA EN SISTEMAS

PROGRAMACIÓN III

TRABAJO PRÁCTICO 1: 2048

FECHA DE ENTREGA: 16/04/2024

PROFESORES:

PATRICIA BAGNES

IGNACIO SOTELO

ALUMNOS:

FRANCISCO DOMINGUEZ

MELINA GOMEZ

MICAELA BARBARA PALOMEQUE

(micaelapalomeque07@gmail.com)

2024

INTRODUCCIÓN

El presente informe, muestra el desarrollo del conocido juego de lógica llamado "2048", utilizando el lenguaje de programación Java y la herramienta de diseño de interfaces WindowBuilder.

El juego "2048" es un rompecabezas numérico que desafía al jugador a combinar celdas con números para alcanzar el valor de 2048 en una cuadrícula de 4x4. La mecánica del juego implica mover el tablero hacia arriba, abajo, izquierda o derecha, combinando las celdas del mismo valor para crear una nueva con el doble del valor original. El objetivo es alcanzar la cifra 2048 y continuar combinando para lograr la puntuación más alta posible.

En este documento, se detallará el proceso de desarrollo del juego "2048", incluida la creación de la interfaz gráfica de usuario utilizando WindowBuilder, la implementación de la lógica del juego en Java y las funcionalidades adicionales agregadas para mejorar la experiencia del usuario.

A lo largo del informe, se analizarán los desafíos encontrados durante el desarrollo, las decisiones de diseño tomadas y las lecciones aprendidas en el proceso.

IMPLEMENTACIÓN

Lógica:

La lógica del juego 2048 se basa en la capacidad del jugador para combinar números en una cuadrícula de 4x4 y alcanzar el valor objetivo de 2048. Esta implementación del juego utiliza un enfoque algorítmico para manejar los movimientos del jugador y la generación de nuevos números en la cuadrícula.

Inicialmente el juego debe empezar con dos números ubicados aleatoriamente (2 o 4) en posiciones vacías de la cuadrícula. Luego a medida que el usuario se mueva en el tablero con las teclas direccionales o las teclas gamer “a, s, d y w”, se ira generando un numero nuevo en las celdas vacías, agregando dificultad al jugador.

Para lograr agregar números fueron claves los siguientes métodos, con ayuda de las funciones de Random:

-generar2o4(): Genera aleatoriamente un número 2 o 4 basado en una distribución uniforme.

```
private int generar2o4()
{
    Random r = new Random();
    int n = r.nextInt(2);
    return (n == 0) ? 2 : 4;
}
```

-generarPosicion(): Selecciona una posición vacía aleatoria en la cuadrícula donde se colocará el nuevo número.

```
private int[] generarPosicion()
{
    int[] posicion = new int[2];
    Random r = new Random();

    posicion[0] = r.nextInt(dimension); posicion[1] = r.nextInt(dimension);

    while(!esPosicionValida(posicion[0], posicion[1]))
    {
        posicion[0] = r.nextInt(dimension); posicion[1] = r.nextInt(dimension);
    }
    return posicion;
}
```

Estos métodos se utilizan en conjunto para agregar nuevos números en posiciones vacías después de cada movimiento válido del jugador.

Decidimos crear un método principal de movimiento que se encargara de mover los números en la cuadrícula hacia una dirección específica, como hacia la derecha, izquierda, arriba o abajo. Comienza creando una copia del tablero actual para no modificar el estado original. Luego, itera sobre cada fila o columna de la cuadrícula y realiza una serie de operaciones para desplazar los números hacia la dirección deseada. Las operaciones incluyen permutaciones para reorganizar los números y sumas para combinar números adyacentes del mismo valor, para finalmente, devolver el nuevo tablero con los movimientos aplicados.

```
private int[][] mover(int[][] tab)
{
    int[][] nuevoTablero = new int[dimension][dimension];

    for (int i = 0; i < dimension; i++)
    {
        for (int j = 0; j < dimension; j++)
        {
            nuevoTablero[i][j] = tab[i][j];
        }
    }

    for (int i = 0; i < dimension; i++)
    {
        while (!estaAlaDerecha(nuevoTablero[i]))
        {
            realizarPermutaciones(nuevoTablero[i]);
        }
        realizarSuma(nuevoTablero[i]);
        realizarPermutaciones(nuevoTablero[i]);
    }

    return nuevoTablero;
}
```

Los movimientos direccionales se basarán en el movimiento principal utilizando una serie de funciones auxiliares, esto nos permitió aprovechar un solo movimiento para los demás, disminuyendo la cantidad de lineas de nuestro código y facilitando su comprensión:

-trasponerMatriz(int[][] matriz): Transpone una matriz dada, intercambiando filas por columnas.

-rotarDerechaMatriz(int[][] matriz): Rota una matriz 90 grados hacia la derecha.

-rotarIzquierdaMatriz(int[][] matriz): Rota una matriz 90 grados hacia la izquierda.

-invertirMatriz(int[][] matriz): Invierte el orden de las columnas de una matriz.

-sonMatricesIguales(int[][] matriz1, int[][] matriz2): Verifica si dos matrices son idénticas en contenido.

Estas funciones auxiliares se utilizan dentro del método principal mover(), el cual realiza las permutaciones correspondientes al movimiento hacia la derecha del usuario; y los métodos de movimiento direccional para realizar las operaciones necesarias en la cuadrícula.

Adicionalmente, consideramos conveniente que la actualización de la matriz que representa nuestro tablero, alojada en una variable local “int[][] tablero”, se haga tras realizar los movimientos correspondientes, en lugar de actualizarse progresivamente. De este modo evitamos utilizar de manera irresponsable actualizarTablero().

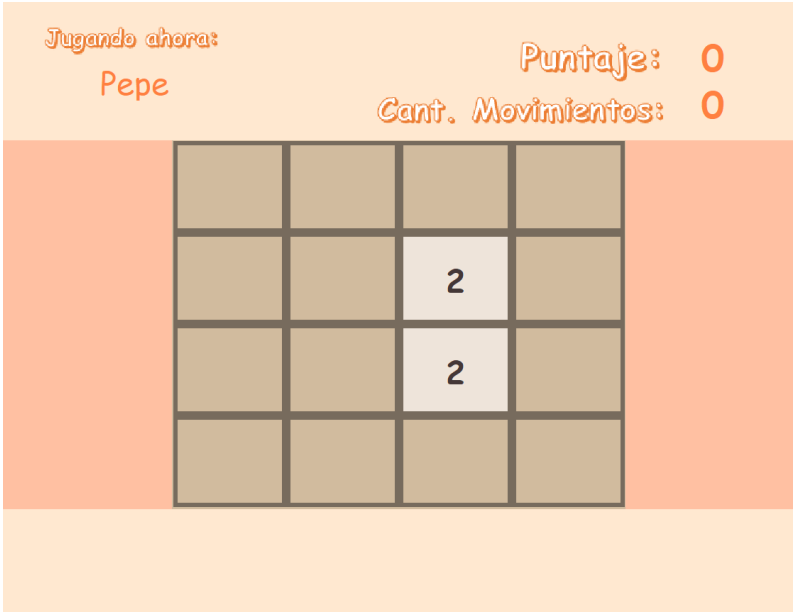
Interfaz gráfica:

Para construir la interfaz del usuario, nos enfocamos en la arquitectura “Forms and Controls”, utilizando componentes visuales predefinidos (como JPanel, JLabel, etc.) para construir dicha interfaz.

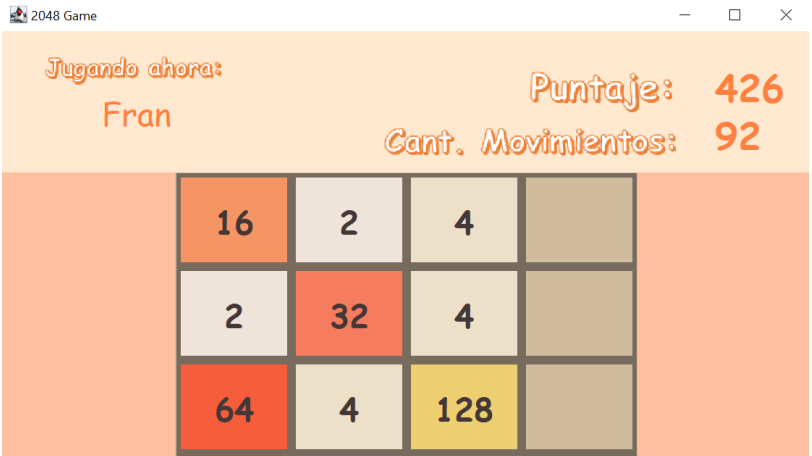
Para el manejo de eventos, se implementa la interfaz KeyListener para capturar los eventos de teclado y permitir que el usuario controle el juego, se crea la clase de nuestro juego y se consultara el estado de los movimientos, puntajes y si este gano o perdió, teniendo interacción directa constantemente.

Al comenzar el juego se mostrará una ventana PantallaMenu, donde el usuario deberá ingresar su nombre y este podrá dar comienzo a la partida.

Para esta ventana VentanaJuego2048, se decidió crear cinco paneles: uno superior para colocar el nombre del usuario, puntajes y cantidad de movimientos, uno para el tablero central (copia el mismo tablero que se crea en el juego), dos paneles laterales y otro inferior.



Se ha elegido una paleta de colores cálidos y suaves para crear una atmósfera acogedora y agradable. El uso de bordes y espaciado adecuado entre los elementos visuales contribuye a una apariencia ordenada y fácil de entender. Además, cada celda del tablero muestra el valor del número correspondiente y se colorea según su valor, lo que facilita la identificación de los números en el juego.



Cuando el jugador gana o pierde se muestran las ventanas correspondientes, mostrándole al usuario su nombre y puntuación:

Nota: Se uso un valor bajo para poder demostrar la VentanaGano, el juego se gana cuando el usuario llegue a la cifra 2048

CONCLUSIÓN

En conclusión, el desarrollo de este juego “2048” utilizando WindowBuilder y Java ha sido una experiencia enriquecedora que nos ha permitido aplicar conceptos de programación orientada a objetos, manejo de eventos, diseño de interfaces gráficas y lógica de programación aprendidos en la materia. A lo largo del proyecto, hemos logrado implementar con éxito las reglas del juego, proporcionando al jugador una experiencia interactiva y entretenida.

Mediante el uso de WindowBuilder, pudimos diseñar y estructurar la interfaz gráfica de manera intuitiva y sencilla, aprovechando los componentes predefinidos de Swing para crear una interfaz atractiva y funcional. La distribución de los elementos en paneles y la atención al diseño visual contribuyeron a una experiencia de usuario satisfactoria.

En el aspecto de la lógica del juego, implementamos algoritmos para los movimientos de las celdas, la generación aleatoria de números y la detección de condiciones de victoria o derrota, siendo los movimientos los que más dificultades nos presentaron.

En resumen, este proyecto no solo nos ha permitido desarrollar un juego funcional de 2048, sino que también nos ha proporcionado una experiencia práctica invaluable en el diseño y desarrollo de aplicaciones Java con interfaces gráficas. A través de este proceso, hemos fortalecido nuestras habilidades de programación y adquirido un mayor conocimiento sobre el desarrollo de interfaces gráficas, siendo antes un concepto poco conocido para nosotros.