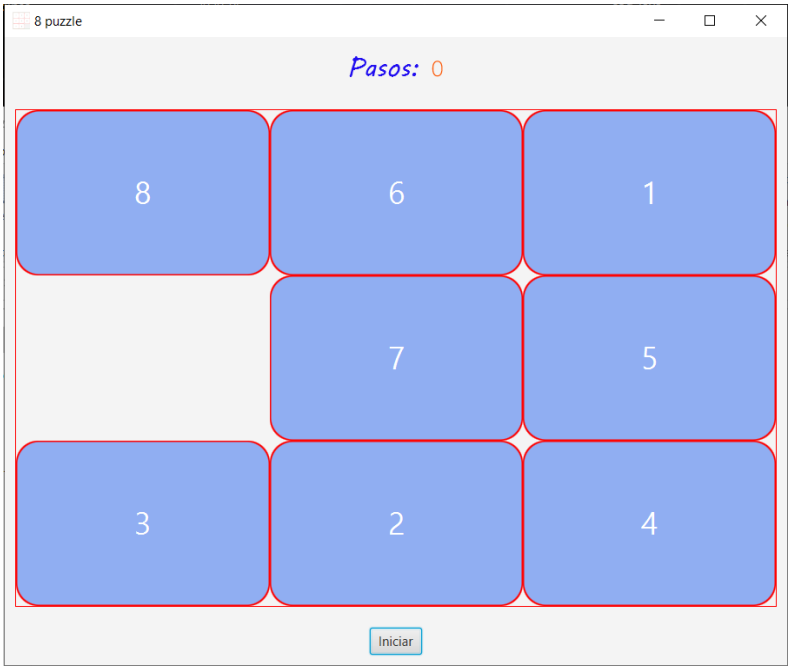
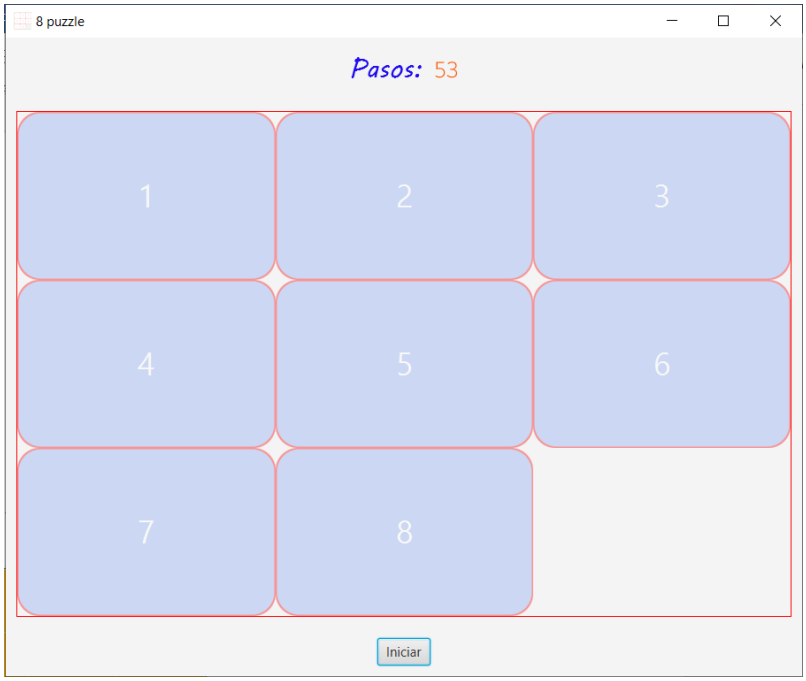
8PUZZLE

El 8 puzzle es un juego en el que tenemos un grid de 3x3 celdas en el cual hay depositados 8 botones o fichas, cada una de ellas con un número entre el 1 y el 8 y sin valores repetidos. Dentro del grid hay un único hueco sobre el cual podremos desplazar solamente las fichas adyacentes.



El juego termina cuando tenemos la combinación de fichas ordenadas según la siguiente figura:



En este juego hay combinaciones del estado inicial que son imposible de resolver. La manera de comprobar si el estado inicial es resoluble es sencilla y solamente hay

que comprobar dos a dos todos los números de la lista por orden, si el segundo numero es menor que el primero estaremos ante una inversión. Un estado inicial es factible si la suma de todas las inversiones es par. Por ejemplo: 1, 2, 3, 4, 5, 6, 8, 7. en esta caso solamente esta la inversión (8,7), luego es impar e irresoluble.

El siguiente código java genera un vector de n enteros que cumple con un posible estado inicial del juego:

```
public class Utils {

    static public int[] generarVectorAleatorio(int n) {
        int inversiones;
        int[] numeros = new int[n];
        int[] resultado = new int[n];
        Random rnd = new Random();

        do {
            //se rellena una matriz ordenada (1..n)
            for (int i = 0; i < n; i++) {
                numeros[i] = i + 1;
            }
            //se se genera el vector aleatorio sin elementos repetidos)
            int k = n;
            int res;
            for (int i = 0; i < n; i++) {
                res = rnd.nextInt(k);
                resultado[i] = numeros[res];
                numeros[res] = numeros[k - 1];
                k--;
            }
            // calculamos el numero de inversiones del vector resultante
            inversiones = 0;
            for (int i = 0; i < n - 1; i++) {
                for (int j = i + 1; j < n; j++) {
                    if (resultado[i] > resultado[j]) {
                        inversiones++;
                    }
                }
            }
        } while (inversiones % 2 != 0);
         // el estado incial tiene solucion si el numero de inversiones es par

        return resultado;
    }
}
```

Recuerda que tienes que invocar a este método y genera un vector de 8 números, el hueco no esta incluido. Si el hueco esta en la última posición (2,2), podrías recorrer el bucle y asociar el texto a los botones y reposicionarlos en la celda de la siguiente manera: (recuerda que esto es solo una opción, tu decides!!!)

```
    private void reiniciarJuego(MouseEvent event) {
        int[] vectorInicial = generarVectorAleatorio(8);
        int i=0;
        for (Node node: gridTablero.getChildren())
        {
            ButtonCelda ficha = (ButtonCelda) node;
            int col=i%3;
            int row= i/3;
            ficha.setText(Integer.toString(vectorInicial[i]));
            gridTablero.setConstraints(ficha,col,row);
            i++;
        }
        hueco_X= 2;
        hueco_Y= 2;
```

……

TRABAJO A REALIZAR:

Implementar en JavaFX el 8 puzzle con la gestión de eventos necesario para poder desplazar las fichas del juego con el ratón. Hay que hacer uso de los principios de diseño que se estudian en clase de teoría así como el patrón Modelo-Vista-Controlador

Para una implementación sencilla se recomienda que el tablero de juego sea un GridPane y que las fichas sean del tipo Button. Se aceptan implementaciones avanzadas.

Debes de utilizar un contador del tipo IntegerProperty para seguir el número de pasos y enlazarlo con el texto de la interfaz

ANEXO 1

Cuando se introducen los Button en un GridPane desde SceneBuilder normalmente se añade al fichero FXML la posición de la fila o de la columna pero no las dos. Para poder implementar el juego vamos a tener que invocar a las funciones del GridaPane: fila = miGridpane.getRowIndex(miButton); columna= miGridpane.getColumnIndex(miButton); y estos métodos fallaran si no modificamos el fichero FXML y añadimos a cada button su columna y su fila. Para ello al editar el fichero deberemos de comprobar que para cada button tengamos definidos el siguiente texto:

```
styleClass="Button" text="1" GridPane.columnIndex="0" GridPane.rowIndex="0">
```

y si no lo esta lo añadiremos (debe de existir el GridPane.columnIndex y el GridPane.rowIndex para todos los button dentro del grid).
Una solución alternativa es reposicionar los botones en el grid al inicio de la aplicación, en el método initialize() de la clase controladora podemos ejecutar un bucle en el que reposicionamos mediante el método gridpane.setConstraints(botón, columna, fila)

ANEXO 2

La solución mas sencilla pasa por añadir un evento de clic del ratón sobre el botón y si este es adyacente al hueco los movemos de celda en el grid al hueco y actualizamos la posición el hueco.

También podemos arrastrar el botón con el ratón. Para desplazar el botón sobre el grid cuando hacemos un drag con el ratón lo que tenemos que hacer es una traslación del punto de pintado del botón en el eje de las X y otro en el de las Y. Para ello tenemos que invocar a los métodos:

```
but.setTranslateX(newXPosition);
but.setTranslateY(newYPosition);
```

La nueva posición deberá de ser la diferencia entre la posición actual del ratón y la posición del ratón al inicio del proceso de drag. Cuando se suelta el botón hay que calcular si se suelta en el hueco y cambiarlos de celda en el grid, en esta situación hay que volver a trasladar el botón a la posición (0,0)