

ALGORITMOS FUNDAMENTALES

GRUPO: GR1

FECHA: 06/02/2019

TEMA: Trabajo Final

INTEGRANTES: -Rodríguez Mauricio

- Quijia Lizeth

- Vargas Erick

INFORME TRABAJO FINAL

1. OBJETIVO GENERAL:

- Implementar todos los algoritmos de ordenamiento, tratados en clase, para verificar la eficiencia de cada uno basado en el tiempo de ejecución.
- Implementar una aplicación basada en el uso de varios algoritmos tratados en el curso.

2. OBJETIVO ESPECÍFICO:

- Conocer el funcionamiento de todos los algoritmos de ordenamientos vistos en clase

3. INTRODUCCIÓN:

Algoritmos de Ordenamiento

Los algoritmos de ordenamiento nos permiten, como su nombre lo dice, ordenar. En este caso, nos servirán para ordenar vectores o matrices con valores asignados aleatoriamente.

Para poder ordenar una cantidad determinada de números almacenadas en un vector o matriz, existen distintos métodos (algoritmos) con distintas características y complejidad. Existe desde el método más simple, como el Método Burbuja, que son simples iteraciones, hasta el Quicksort (Método Rápido), que al estar optimizado usando recursión, su tiempo de ejecución es menor y es más efectivo.

Los que vamos a conocer son:

- MergeSort
- QuickSort
- Burbuja

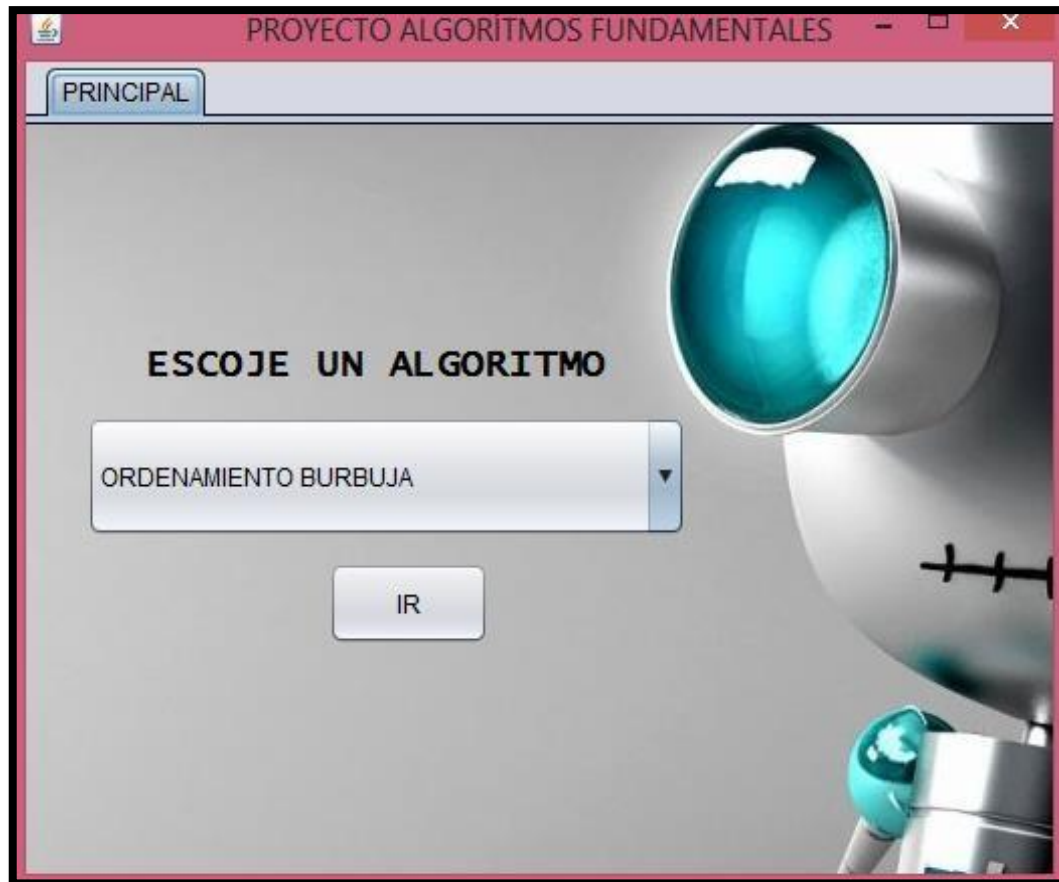
ESCUELA DE FORMACIÓN DE TECNÓLOGOS

- Búsqueda Binaria
- Comparación
- Insercion
- Selección

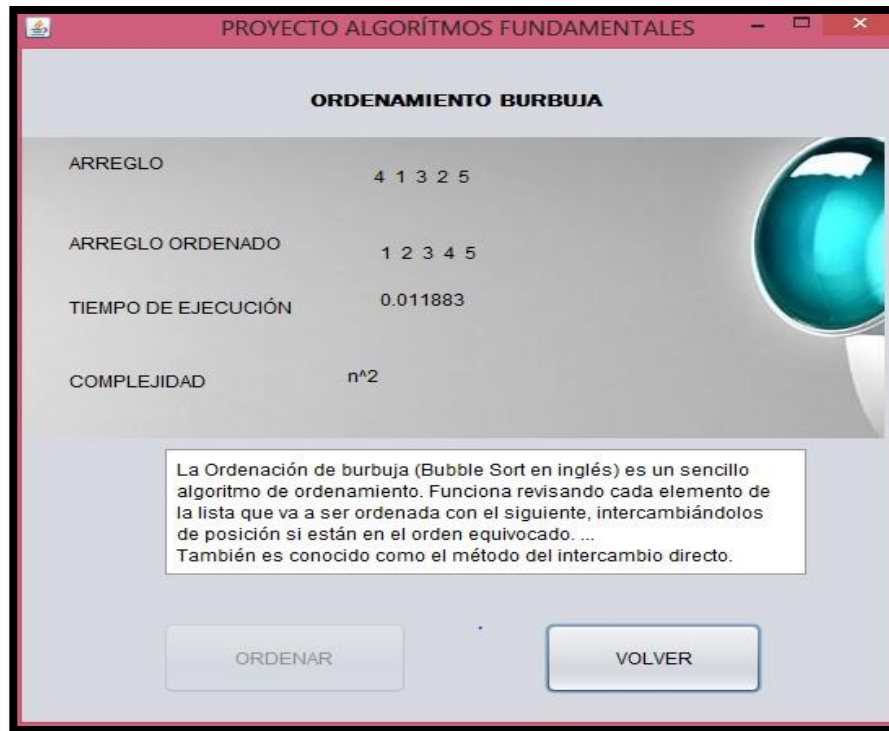
1. DESCRIPCIÓN DE ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA:

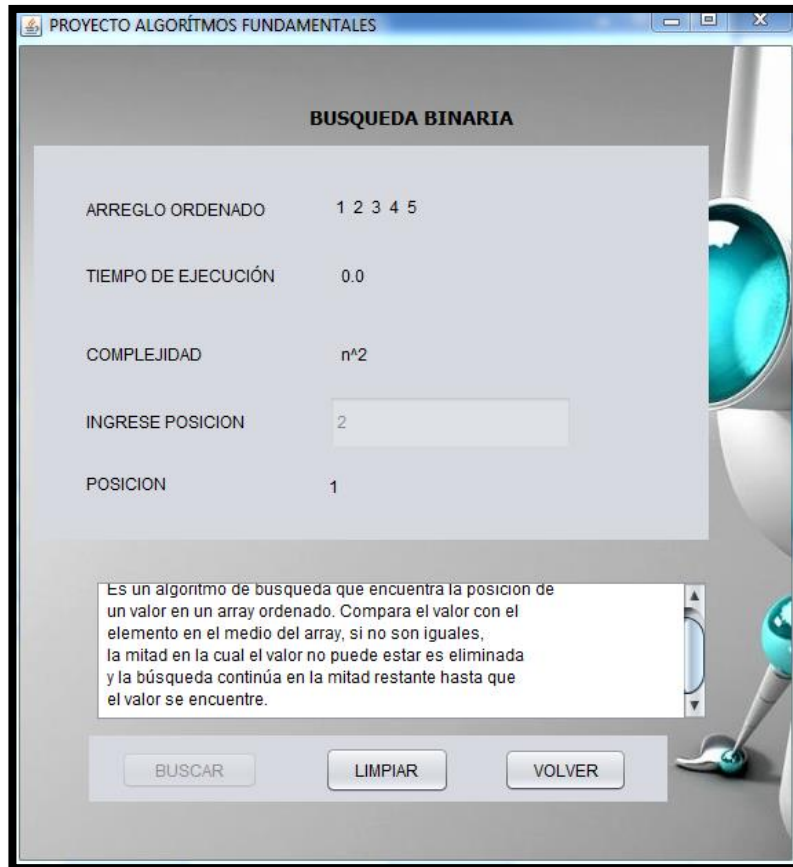
Creemos un menú donde podemos seleccionar cuáles algoritmos deseamos observar su funcionamiento. A continuación, se presentan las imágenes que muestran nuestro programa gráficamente.

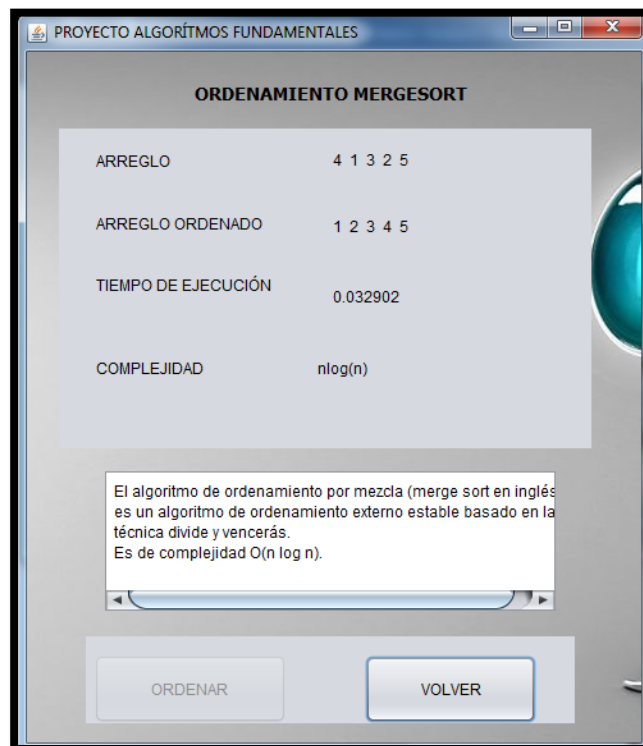
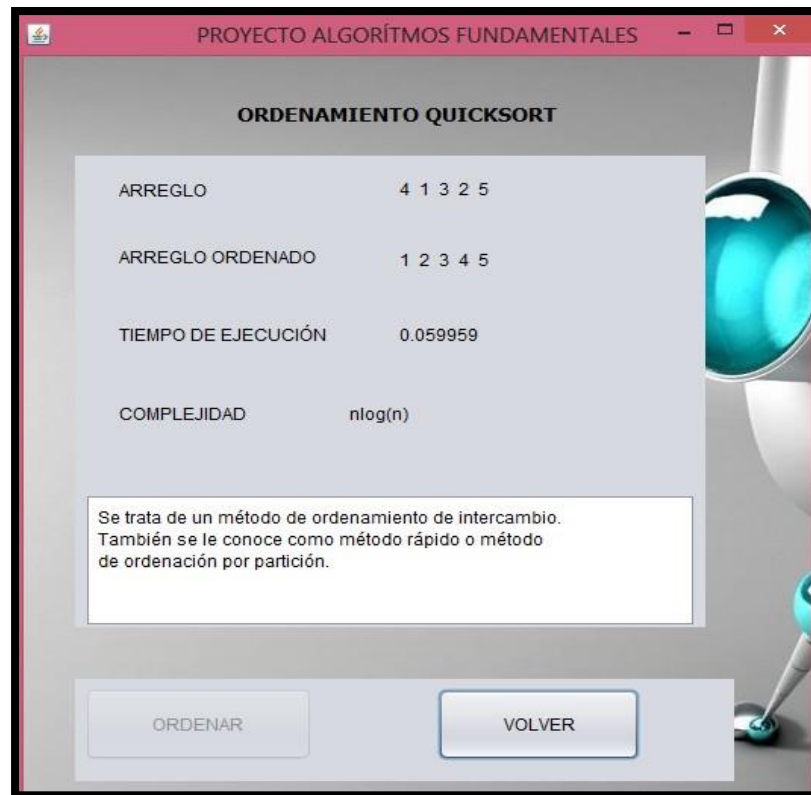
Pantalla de inicio:

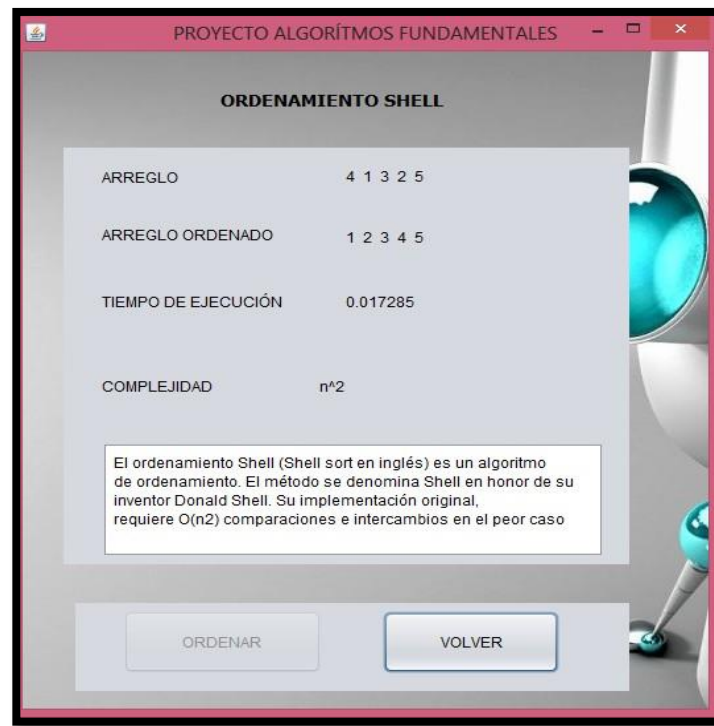


ESCUELA DE FORMACIÓN DE TECNÓLOGOS

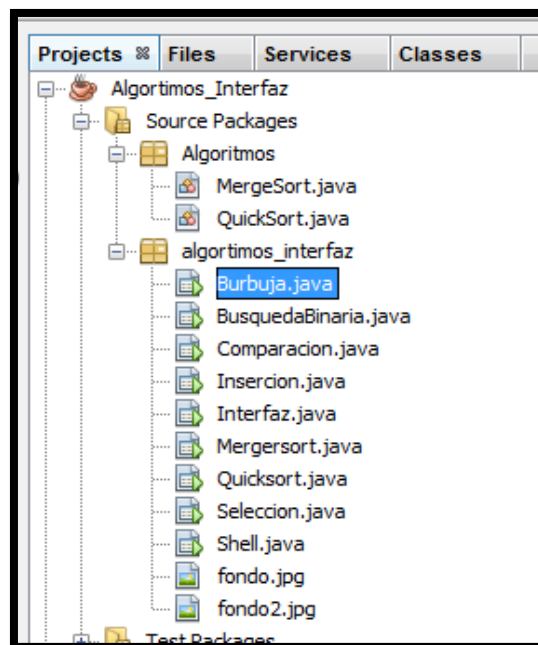






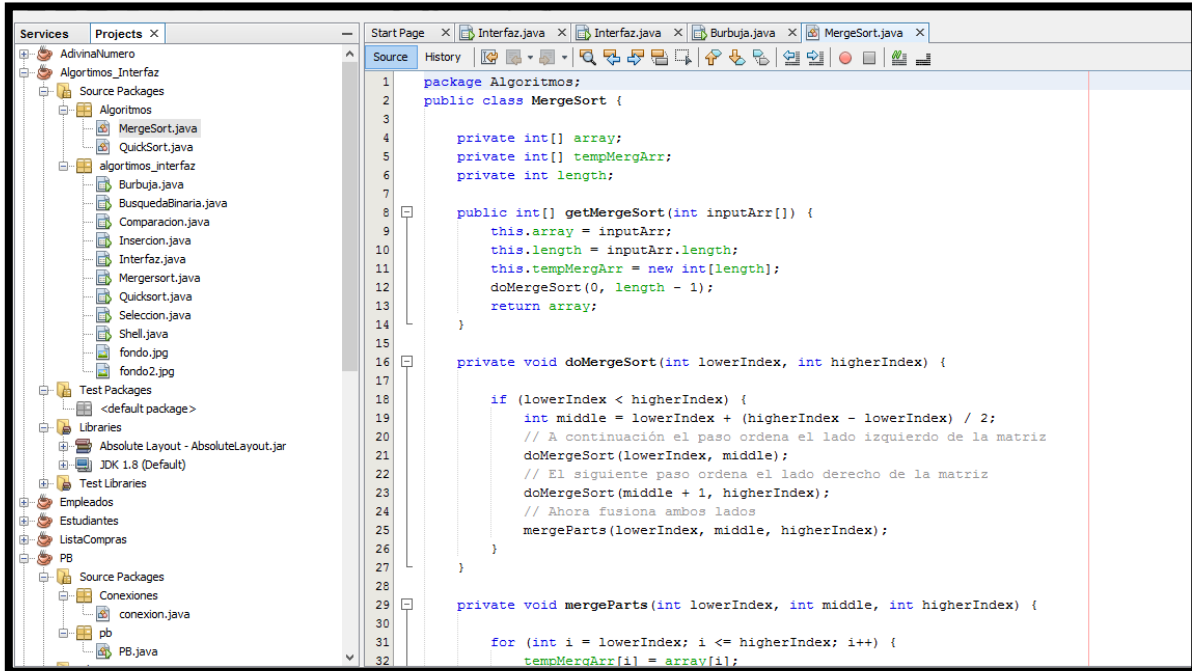


Dos algoritmos de ordenamiento necesitaron la creación de clases para su realización. Estos fueron el MergeSort y el QuickSort como lo observamos en la siguiente imagen.



Los códigos usados en cada algoritmo de ordenamiento:

MergeSort



```

1 package Algoritmos;
2 public class MergeSort {
3
4     private int[] array;
5     private int[] tempMergArr;
6     private int length;
7
8     public int[] getMergeSort(int inputArr[]) {
9         this.array = inputArr;
10        this.length = inputArr.length;
11        this.tempMergArr = new int[length];
12        doMergeSort(0, length - 1);
13        return array;
14    }
15
16    private void doMergeSort(int lowerIndex, int higherIndex) {
17
18        if (lowerIndex < higherIndex) {
19            int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
20            // A continuación el paso ordena el lado izquierdo de la matriz
21            doMergeSort(lowerIndex, middle);
22            // El siguiente paso ordena el lado derecho de la matriz
23            doMergeSort(middle + 1, higherIndex);
24            // Ahora fusiona ambos lados
25            mergeParts(lowerIndex, middle, higherIndex);
26        }
27    }
28
29    private void mergeParts(int lowerIndex, int middle, int higherIndex) {
30
31        for (int i = lowerIndex; i <= higherIndex; i++) {
32            tempMergArr[i] = array[i];

```

```

    private void mergeParts(int lowerIndex, int middle, int higherIndex) {

        for (int i = lowerIndex; i <= higherIndex; i++) {
            tempMergArr[i] = array[i];
        }
        int i = lowerIndex;    //índice inferior
        int j = middle + 1;    //índice de la mitad
        int k = lowerIndex;
        while (i <= middle && j <= higherIndex) {
            if (tempMergArr[i] <= tempMergArr[j]) {
                array[k] = tempMergArr[i];
                i++;
            } else {
                array[k] = tempMergArr[j];
                j++;
            }
            k++;
        }
        while (i <= middle) {
            array[k] = tempMergArr[i];
            k++;
            i++;
        }
    }
}

```



```

1 package algoritmos_interfaz;
2
3
4 import Algoritmos.MergeSort;
5
6 public class Mergersort extends javax.swing.JFrame {
7     int Arreglo[] = {4, 1, 3, 2, 5};
8     public Mergersort() {
9         initComponents();
10        this.setTitle("PROYECTO ALGORÍTMOS FUNDAMENTALES");
11        for (int i = 0; i < Arreglo.length; i++) {
12            String desordenado = Integer.toString(Arreglo[i]);
13            lbl1.setText(lbl1.getText() + desordenado + " ");
14        }
15    }
16 }
17

```

```

private void btnInsercionActionPerformed(java.awt.event.ActionEvent evt) {

    double tiempo = 0;
    MergeSort mer = new MergeSort();
    long time_start, time_end;
    time_start = System.nanoTime();

    mer.getMergeSort(Arreglo);

    time_end = System.nanoTime();
    tiempo = (time_end - time_start) / 1e6;

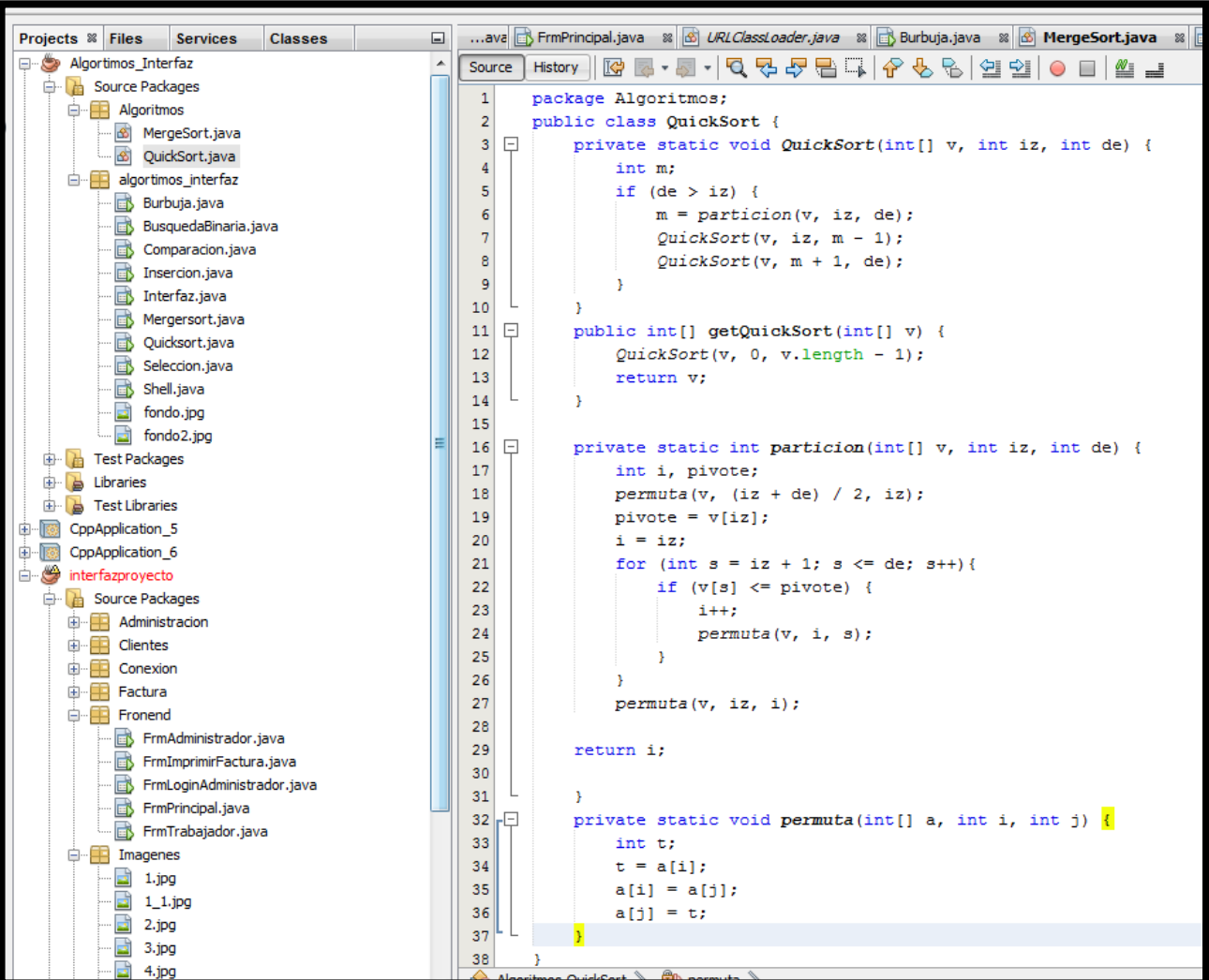
    for (int i = 0; i < Arreglo.length; i++) {
        String ordenado = Integer.toString(Arreglo[i]);
        lbl2.setText(lbl2.getText() + ordenado + " ");
    }
    lbl3.setText(Double.toString(tiempo));
    lbl4.setText("nlog(n)");
    btn1.setEnabled(true);
    btnInsercion.setEnabled(false);
}

private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    Interfaz reabrir = new Interfaz();
    reabrir.setVisible(true);
}

/**

```


QuickSort



```

1 package Algoritmos;
2 public class QuickSort {
3     private static void QuickSort(int[] v, int iz, int de) {
4         int m;
5         if (de > iz) {
6             m = particion(v, iz, de);
7             QuickSort(v, iz, m - 1);
8             QuickSort(v, m + 1, de);
9         }
10    }
11    public int[] getQuickSort(int[] v) {
12        QuickSort(v, 0, v.length - 1);
13        return v;
14    }
15
16    private static int particion(int[] v, int iz, int de) {
17        int i, pivote;
18        permuta(v, (iz + de) / 2, iz);
19        pivote = v[iz];
20        i = iz;
21        for (int s = iz + 1; s <= de; s++) {
22            if (v[s] <= pivote) {
23                i++;
24                permuta(v, i, s);
25            }
26        }
27        permuta(v, iz, i);
28
29        return i;
30    }
31
32    private static void permuta(int[] a, int i, int j) {
33        int t;
34        t = a[i];
35        a[i] = a[j];
36        a[j] = t;
37    }
38 }
  
```

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

```

L  */
   package algoritmos_interfaz;

   import Algoritmos.QuickSort;

   /**
    *
    * @author ADMIN-MINEDUC
    */
   public class Quicksort extends javax.swing.JFrame {

       int Arreglo[] = {4, 1, 3, 2, 5};

       public Quicksort() {
           initComponents();
           this.setTitle("PROYECTO ALGORÍTMOS FUNDAMENTALES");
           for (int i = 0; i < Arreglo.length; i++) {
               String desordenado = Integer.toString(Arreglo[i]);
               lbl1.setText(lbl1.getText() + desordenado + " ");
           }
       }
   }

```

```

private void btnInsercionActionPerformed(java.awt.event.ActionEvent evt) {

    double tiempo = 0;
    QuickSort qui = new QuickSort();
    long time_start, time_end;
    time_start = System.nanoTime();

    qui.getQuickSort(Arreglo);

    time_end = System.nanoTime();
    tiempo = (time_end - time_start) / 1e6;

    for (int i = 0; i < Arreglo.length; i++) {
        String ordenado = Integer.toString(Arreglo[i]);
        lbl2.setText(lbl2.getText() + ordenado + " ");
    }
    lbl3.setText(Double.toString(tiempo));
    lbl4.setText("nlog(n)");
    btn1.setEnabled(true);
    btnInsercion.setEnabled(false);
}

private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    Interfaz reabrir = new Interfaz();
    reabrir.setVisible(true);
}

```

Burbuja

```

1 package algortimos_interfaz;
2
3
4
5 public class Burbuja extends javax.swing.JFrame {
6     int Arreglo[]={4,1,3,2,5};
7
8     public Burbuja() {
9         initComponents();
10        this.setTitle("PROYECTO ALGORÍTMOS FUNDAMENTALES");
11        for (int i = 0; i < Arreglo.length; i++) {
12            String desordenado = Integer.toString(Arreglo[i]);
13            lbl1.setText(lbl1.getText() + desordenado + " ");
14        }
15    }
16

```

```

private void btnBurbujaActionPerformed(java.awt.event.ActionEvent evt) {

    double tiempo=0;
    long time_start, time_end;
    time_start = System.nanoTime();
    int aux = 0;
    for (int i = 0; i < Arreglo.length - 1; i++) {
        for (int j = 0; j < Arreglo.length - i - 1; j++) {
            if (Arreglo[j + 1] < Arreglo[j]) {
                aux = Arreglo[j + 1];
                Arreglo[j + 1] = Arreglo[j];
                Arreglo[j] = aux;
            }
        }
    }
    time_end = System.nanoTime();
    tiempo=(time_end - time_start)/1e6;

    for (int i = 0; i < Arreglo.length; i++) {
        String ordenado = Integer.toString(Arreglo[i]);
        lbl2.setText(lbl2.getText() + ordenado + " ");
    }
    lbl3.setText(Double.toString(tiempo));
    lbl4.setText("n^2");
    btn1.setEnabled(true);
    btnBurbuja.setEnabled(false);
}

private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    Interfaz reabrir= new Interfaz ();
    reabrir.setVisible(true);
}

```

Búsqueda Binaria

```
String Posicion = txt1.getText();

if (Posicion.equals("")) {
    JOptionPane.showMessageDialog(rootPane, "Debe ingresar un numero");
} else {
    try {
        double tiempo = 0;
        long time_start, time_end;
        time_start = System.nanoTime();

        int inf=0;
        int sup=5;
        int i=0;
        char band='F';
        int dato=0;
        int mitad = 0;
        dato = Integer.parseInt(txt1.getText());

        while((inf<=sup)&&(i<5)){
            mitad = (inf+sup)/2;
            if(Arreglo[mitad] == dato){
                band='V';
                break;
            }
            if(Arreglo[mitad]>dato){
                sup = mitad;
                mitad = (inf+sup)/2;
            }
            if(Arreglo[mitad]<dato){
                inf = mitad;
            }
        }
    }
}
```

```
226
227
228     if(band == 'V'){
229         lbl1.setText(""" + mitad);
230         txt1.setEnabled(false);
231         btnBusqueda.setEnabled(false);
232         lbl3.setText(Double.toString(tiempo));
233         lbl4.setText("n^2");
234     }
235     else{
236         lbl1.setText("NO EXISTE NUMERO EN EL ARREGLO");
237
238         txt1.setEnabled(false);
239         btnBusqueda.setEnabled(false);
240     }
241
242     time_end = System.nanoTime();
243     tiempo = (time_end - time_start) / 1e6;
244
245
246
247     }catch (NumberFormatException nfe) {
248         JOptionPane.showMessageDialog(rootPane, "Debe ingresar solo numeros");
249         txt1.setText("");
250         btnBusqueda.setEnabled(true);
251     }
252 }
253
254 btn2.setEnabled(true);
255
```

Inserción

```

5  |  /**
6  |  package algortimos_interfaz;
7  |
8  |  /**
9  |  *
10 |  * @author ADMIN-MINEDUC
11 |  */
12 |  public class Insercion extends javax.swing.JFrame {
13 |
14 |      int Arreglo[] = {4, 1, 3, 2, 5};
15 |
16 |
17 |      public Insercion() {
18 |          initComponents();
19 |          this.setTitle("PROYECTO ALGORÍTMOS FUNDAMENTALES");
20 |          for (int i = 0; i < Arreglo.length; i++) {
21 |              String desordenado = Integer.toString(Arreglo[i]);
22 |              lbl1.setText(lbl1.getText() + desordenado + " ");
23 |          }
24 |      }
25 |  }

```

```

private void btnInsercionActionPerformed(java.awt.event.ActionEvent evt) {
    lbl4.setText("n^2");
    double tiempo = 0;
    long time_start, time_end;
    time_start = System.nanoTime();
    int aux = 0;

    for (int i = 1; i < Arreglo.length; i++) {
        aux = Arreglo[i];
        int j = i - 1;
        while (j >= 0 && aux < Arreglo[j]) {
            Arreglo[j + 1] = Arreglo[j];
            j = j - 1;
        }
        Arreglo[j + 1] = aux;
    }
    time_end = System.nanoTime();
    tiempo = (time_end - time_start) / 1e6;

    for (int i = 0; i < Arreglo.length; i++) {
        String ordenado = Integer.toString(Arreglo[i]);
        lbl2.setText(lbl2.getText() + ordenado + " ");
    }
    lbl3.setText(Double.toString(tiempo));

    btn1.setEnabled(true);
    btnInsercion.setEnabled(false);
}

private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    Interfaz reabrir = new Interfaz();
    reabrir.setVisible(true);
}

```

Selección

```

5  /**
6   package algoritmos_interfaz;
7
8   /**
9   *
10  * @author ADMIN-MINEDUC
11  */
12  public class Seleccion extends javax.swing.JFrame {
13      int Arreglo[]={4,1,3,2,5};
14      public Seleccion() {
15          initComponents();
16          this.setTitle("PROYECTO ALGORÍTMOS FUNDAMENTALES");
17          for (int i = 0; i < Arreglo.length; i++) {
18              String desordenado = Integer.toString(Arreglo[i]);
19              lbl1.setText(lbl1.getText() + desordenado + " ");
20          }
21      }
22  }

```

```

private void btnSeleccionActionPerformed(java.awt.event.ActionEvent evt) {

    double tiempo=0;
    long time_start, time_end;
    time_start = System.nanoTime();

    int aux,minimo;
    for (int i =0; i <Arreglo.length; i++) {
        minimo= i;
        for (int j= i + 1; j< Arreglo.length; j++) {

            if (Arreglo[j] <Arreglo[minimo]) {
                minimo= j;
            }

        }
        aux = Arreglo[i];
        Arreglo[i] =Arreglo[minimo];
        Arreglo[minimo] = aux;
    }

    time_end = System.nanoTime();
    tiempo=(time_end - time_start)/1e6;

    for (int i = 0; i < Arreglo.length; i++) {
        String ordenado = Integer.toString(Arreglo[i]);
        lbl2.setText(lbl2.getText() + ordenado + " ");
    }

    lbl3.setText(Double.toString(tiempo));
    lbl4.setText("n^2");
    btn1.setEnabled(true);
    btnSeleccion.setEnabled(false);
}

private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    Interfaz reabrir= new Interfaz ();
    reabrir.setVisible(true);
}

```

Shell

```
private void btnInsercionActionPerformed(java.awt.event.ActionEvent evt) {

    double tiempo = 0;
    long time_start, time_end;
    time_start = System.nanoTime();

    int inta, i, aux;
    boolean band;
    inta = Arreglo.length;
    while(inta > 0){
        inta = inta / 2;
        band = true;
        while(band){
            band = false;
            i = 0;
            while ((i+inta) <=Arreglo.length-1){//2.1.1
                if (Arreglo[i] > Arreglo[i + inta]){
                    aux = Arreglo[i];
                    Arreglo[i] = Arreglo[i+inta];
                    Arreglo[i+inta] = aux;
                    band = true;
                }
                i = i +1;
            }
        }

        time_end = System.nanoTime();
        tiempo = (time_end - time_start) / 1e6;

        for (int j = 0; j < Arreglo.length; j++) {
            String ordenado = Integer.toString(Arreglo[j]);
            lbl2.setText(lbl2.getText() + ordenado + " ");
        }
    }
}
```

```
time_end = System.nanoTime();
tiempo = (time_end - time_start) / 1e6;

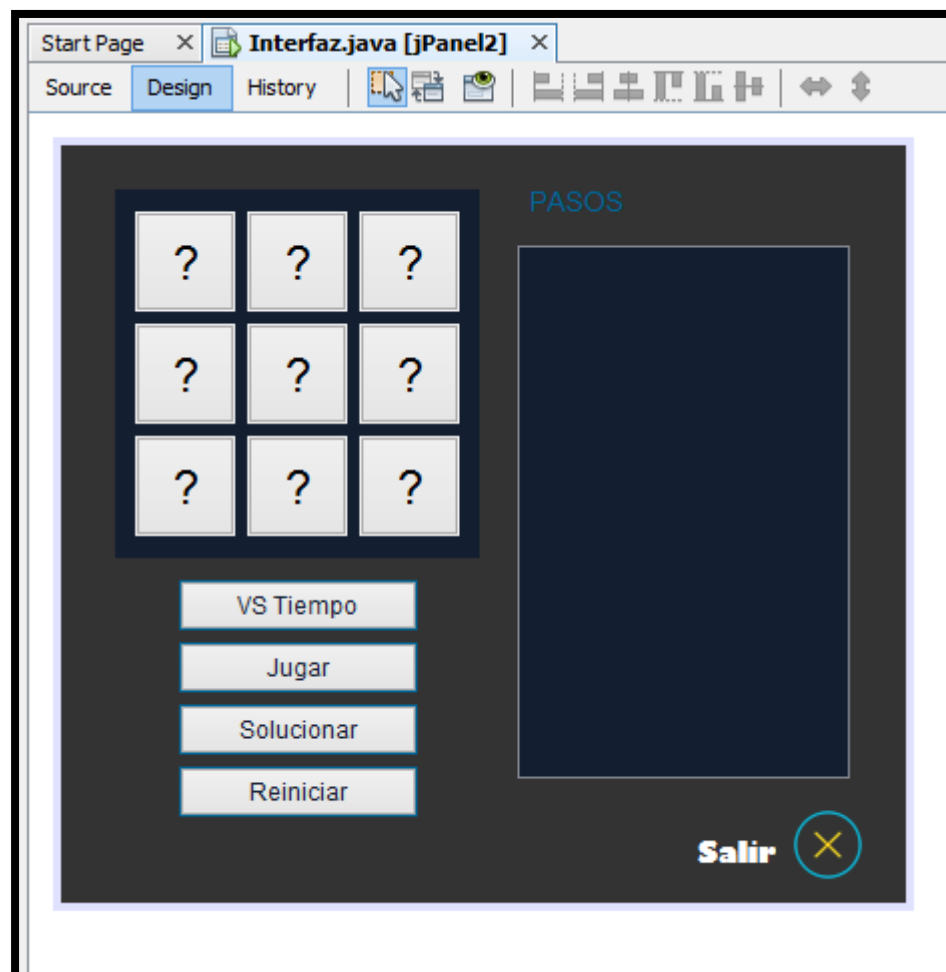
for (int j = 0; j < Arreglo.length; j++) {
    String ordenado = Integer.toString(Arreglo[j]);
    lbl2.setText(lbl2.getText() + ordenado + " ");
}
lbl3.setText(Double.toString(tiempo));
lbl4.setText("n^2");
btn1.setEnabled(true);
btnInsercion.setEnabled(false);
}

private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
    Interfaz reabrir = new Interfaz();
    reabrir.setVisible(true);
}
}
```

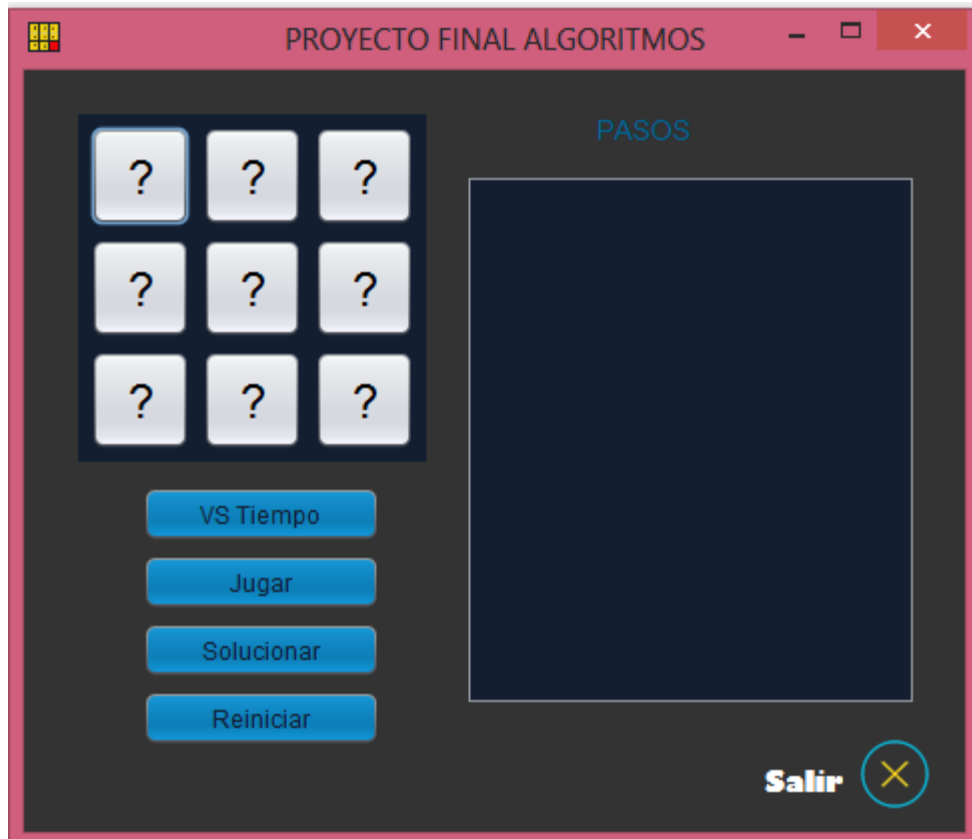

Para la implementación de la aplicación realizamos lo siguiente.

```
//Se Cargan todos los componentes del JFrame
public Interfaz() {
    initComponents();
    setIconImage(new ImageIcon(getClass().getResource("/Imagenes/ganar.jpg")).getImage());
    this.setTitle("PROYECTO FINAL ALGORITMOS");
    inicio=new int [3][3];
    pasos.setVisible(true);
}
```

Realizamos la interfaz en Netbeans colocando los nombres correspondientes.



Al correr el programa tenemos de la siguiente manera la interfaz.



- En el botón JUGAR: Procedemos a poner un color dependiendo el número y a definir diferentes en valores en los 6 casos.

```
// cuando doy clic en el boton jugar, realizo un switch con diferentes casos para el juego
case 1:

    //doy valores a cada cuadro de color
    inicio[0][0]=1;
    inicio[0][1]=2;
    inicio[0][2]=3;
    inicio[1][0]=4;
    inicio[1][1]=0;
    inicio[1][2]=6;
    inicio[2][0]=7;
    inicio[2][1]=5;
    inicio[2][2]=8;
```

```
//Establezco colores dependiendo el numero
b1.setText("1");
b1.setBackground(Color.blue);
b2.setText("2");
b2.setBackground(Color.pink);
b3.setText("3");
b3.setBackground(Color.white);
b4.setText("4");
b4.setBackground(Color.gray);
b5.setText("0");
b5.setBackground(Color.red);
b6.setText("6");
b6.setBackground(Color.orange);
b7.setText("7");
b7.setBackground(Color.yellow);
b8.setText("5");
b8.setBackground(Color.green);
b9.setText("8");
b9.setBackground(Color.black);

break;
```



- En el botón VS Tiempo

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //Estamos en el boton VS tiempo

    // Al momento de dar clic corre el cronometro
    if(!"?".equals(b1.getText()) && !"?".equals(b2.getText()) && !"?".equals(b3.getText()) && !"?".equals(b4.getText())) {
        t= new Timer(1000, (ActionEvent e) -> {
            tiempo--;
            time.setText(tiempo+" ");
            if(tiempo==0){
                //audio para cuando se agota el tiempo
                AudioClip perdersonido;
                perdersonido= java.applet.Applet.newAudioClip(getClass().getResource("/audios/over.wav"));
                perdersonido.play();
                JOptionPane.showMessageDialog(null, "TIEMPO AGOTADO");
                reiniciar();
                t.stop();
            }
        });
        t.start();
    }else{
        JOptionPane.showMessageDialog(null, "[PRIMERO DAR CLIC EN JUGAR]"); // en caso de que el usuario no
    }
}
```

```
//Verifica si todos los botones tienen el valor correspondiente
public void ganaste() {
    if("1".equals(b1.getText()) && "2".equals(b2.getText()) && "3".equals(b3.getText()) && "4".equals(b4.getText())) {
        if(t!=null){
            t.stop();
            tiempo=160;
            time.setText("");
        }
        JOptionPane.showMessageDialog(null, "GANASTE!");
        reiniciar();
    }
}
```



- Al dar clic en el botón SOLUCIONAR

```
private void seleccionarActionPerformed(java.awt.event.ActionEvent evt) {
//Estamos en el boton de solucionar
Nodo iniciar=new Nodo(inicio);
Nodo sol=buscarSolucion(iniciar, solución);
//Muestra los pasos
contador=0;

boolean llave=true;
try {
while(sol.padre!=null & llave==true){
imprimirEstado(sol.getEstado());
System.out.println("-----" + (++contador));
sol=sol.padre;
pasos.setText(Integer.toString(contador));
//Cuando llega a los 20000 sale del while
if(contador==30000){
llave=false;
borrartext();
}
}
} catch (Exception e) {
}
System.out.println("-----");
}
}
```



- Al dar clic en el botón REINICIAR.

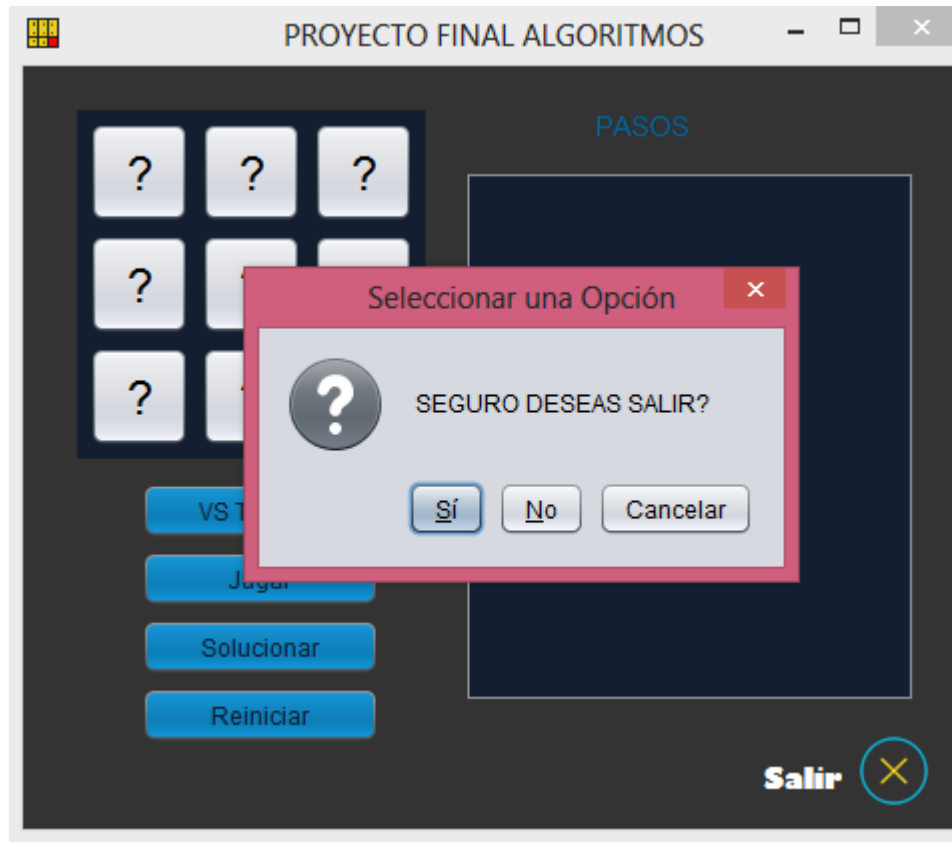
```
private void seleccionar2ActionPerformed(java.awt.event.ActionEvent evt) {  
    reiniciar();  
}
```

```
//Reacomoda todo en su lugar  
public void reiniciar() {  
    b1.setText("?");  
    b2.setText("?");  
    b3.setText("?");  
    b4.setText("?");  
    b5.setText("?");  
    b6.setText("?");  
    b7.setText("?");  
    b8.setText("?");  
    b9.setText("?");  
  
    tiempo=160;  
    time.setText("");  
  
    b1.setBackground(Color.getHSBColor(240,240,240));  
    b2.setBackground(Color.getHSBColor(240,240,240));  
    b3.setBackground(Color.getHSBColor(240,240,240));  
    b4.setBackground(Color.getHSBColor(240,240,240));  
    b5.setBackground(Color.getHSBColor(240,240,240));  
    b6.setBackground(Color.getHSBColor(240,240,240));  
    b7.setBackground(Color.getHSBColor(240,240,240));  
    b8.setBackground(Color.getHSBColor(240,240,240));  
    b9.setBackground(Color.getHSBColor(240,240,240));  
  
    text.setText("");  
    pasos.setText("");  
  
    for (int i=1; i<=10; i++) {
```




- Al dar clic en el botón Salir:

```
private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {
    //Boton de salir
    int res=JOptionPane.showConfirmDialog(null, "SEGURO DESEAS SALIR?");//aparece un mensaje para asegurars
    if(res==0){// si es verdad Entonces termina el programa
        System.exit(0);
    }else{//caso contrario no hago nada se queda ahi
    }
}
```



2. CONCLUSIÓN:

- Los métodos de ordenamiento de datos son muy útiles, ya que la forma de arreglar los registros de una tabla en algún orden secuencial de acuerdo a un criterio de ordenamiento, el cual puede ser numérico, alfabético o alfanumérico, ascendente o descendente; nos facilita las búsquedas de cantidades de registros en un moderado tiempo, en modelo de eficiencia. Mediante sus técnicas podemos colocar listas detrás de otras y luego ordenarlas, como también podemos comparar pares de valores de llaves, e intercambiarlos si no se encuentran en sus posiciones correctas.
- En la aplicación del juego mostrado en el informe se utilizó el algoritmo QuickSort, ya que presenta un pivote que es el cero "0", el cuál es la base de la solución pues mediante este número se sigue comparando e intercambiando hasta llegar a la solución. Un punto importante es que en este caso el pivote ya está definido y no cambia, siempre será el cero.
- En la aplicación también en ciertas partes utilizamos la programación dinámica, la cual es una técnica que permite la optimización de soluciones a problemas adaptan dándolos

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

a la metodología divide y vencerás, fraccionando el problema en subproblemas y solucionando a cada uno de ellos mediante el uso de la recursividad para luego combinar estas soluciones parciales para obtener la solución al problema.

3. REFERENCIAS:

- Lagos, F. (2007). Algoritmos de Ordenamiento. Recuperado el 06-02-2019 de https://blog.zerial.org/ficheros/Informe_Ordenamiento.pdf

