Guía código Búsqueda Binaria

Nombre: **Juan David Jiménez Romero** **NRC:** **16137**

Código BusquedaBinariaPresentacion:

La primera línea nos dice el paquete en el cual se encuentra el programa

package busquedabinariapresentacion;

import javax.swing.SwingUtilities; esta línea nos dice que llama a la librería que nos permite hacer uso de la utilidades de la librería Swing

clase principal de la BusquedaBinariaPresientacion

public class BusquedaBinariaPresentacion esta línea nos dice el inicio de la clase publica de BusquedaBinaria

esta línea es del metodo statico void que realiza la funcion de ejecutar el programa

public static void main(String[] args) inicio del metodo main que ejecuta el programa

SwingUtilities.invokeLater(new Runnable() nos dice que con la libreria SwingUtilities crea un nuevo ejecutable

@Override que sale por el uso del método run()

public void run() esta línea es el método que con ayuda de la librería ejecuta lo siguiente

new Ordenarxd() esta línea esto dice que crea una ventana con el contenido de la clase Ordenarxd()

finalización de la clase principal

Codigo de la BusquedaBinariaPresentacion:

```
//paquete en el cual se encuentra el programa

package busquedabinariapresentacion;

import javax.swing.SwingUtilities;//librería que nos permite hacer uso de la utilidades de la librería Swing


/**

 *

 * @author jrome

 */

//clase principal de la BusquedaBinariaPresientacion

public class BusquedaBinariaPresentacion { //esta es la clase publica de BusquedaBinaria

    //esta es el metodo statico void que realiza la funcion de ejecutar el programa
```

public static void main(String[] args) {//inicio del metodo main que ejecuta el programa

SwingUtilities.invokeLater(new Runnable(){//nos dice que con la libreria SwingUtilities crea un nuevo ejecutable

@Override

public void run(){//metodo que con ayuda de la libreria ejecuta lo siguiente

new Ordenarxd();//esto dice que crea una ventana con el contenido de la clase Ordenarxd()

}//fin del metodo run()

});//fin del SwuingUtilities

}//fin del metodo main

}//fin de la clase BusquedaBinariaPresentacion

Clase Radix:

Se empieza la clase con la línea package ordenamientoradix nos indica que está en el paquete ordenaminetoRadix

Por consiguiente las llamadas de las librerías import java.util.LinkedList nos dice que usa la librería de listasEnlazadas que nos permite ocupar las funciones de estas

Y la línea import java.util.Queue; que nos dice que usa la librería de colas y nos deja usar los métodos únicos para este tipos de TDA

public class ordenRadix esta línea da inicio de la clase ordenRadix

public static int[] ordenacionRadix(int[] vec) es el inicio del método estático entero ordenación Radix que contiene el valor vec que es un array

int rep = 1; esta línea nos dice que se declara la variable rep de tipo entero y que inicialize con el valor de 1

int numBytes = 4 esta línea declara una variable de numBytes que inicializa con el numero 4, número de bytes a desplazar

int numColas = (int) Math.pow(2, numBytes);

Queue[] cola = new LinkedList[numColas] esta linea se encarga de la creación de la cola con el nombre cola que tiene como tamaño la variable numColas

for (int i = 0; i < numColas; i++) esta línea es un for que nos dice que recorre la cola donde cada repetición realiza dentro de la cola una lista enlazada

cola[i] = new LinkedList();

} aquí se termina el for

for (int i = 0; i < rep; i++) { //for que recorre dependiendo de la variable rep de repeticiones que se realizara al programa

// En esta parte recorre el vector para guardar cada valor en la cola

```
        for (int numero : vec) {
            // Busca el mayor número del vector
            if (i == 0) { //Si i es igual a 0 se realizara la siguiente función
                if (numero > rep) {//si número es mayor a rep este realizara lo siguiente
                    rep = numero; // rep toma el valor de numero
            // Calcula en que cola debe ir cada número
            int numCola = (numero >> div) & 0xf;
            cola[numCola].add(numero);
        }
        div = div + numBytes;
        // Recorre cada cola para colocar cada elemento en el vector
        int j = 0 esta línea se declara una variable j de tipo entero que inicializa en 0
        for (Queue c : cola) for para pasar los datos a la cola cola a la cola c
            while (!c.isEmpty()) esta linea nos dice que la variable c mientras este no este vacia realizara lo siguiente
                vec[j++] = (int) c.remove() esta línea recorre el array vec y le va  asignando los datos de la cola c y remueve el primer valor de este haciendo que el segundo valor de la cola c pase a ser primera hasta que esta se quede vacia
        // La primera vez se actualiza el número de veces que debe ejecutar el proceso
        if (i == 0)
            rep = (int) (Math.log(rep) / Math.log(numColas)) + 1;
    return vec regresa el valor del array vec de tipo entero
  public static void imprimirVector(int vec[]) este metodo sirve para imprimir en consola los datos del array
     for(int i=0;i<vec.length;i++) esta linea dice que el for va a repetirse hasta que i deje de ser menor al tamaño del arreglo este para que imprima todos los valores q estén dentro del array
        System.out.print(vec[i]+" "); esta línea muestra en pantalla el valor dentro del array vec[i] y lo que este dentro de la posicion i
```

Código de la clase Radix:

```
package ordenamientoradix;//nos indica que esta en la libreria ordenaminetoRadix
```

```
import java.util.LinkedList;//dice que usa la libreria de listasEnlazadas que nos permite ocupar las funciones de estas

import java.util.Queue;//dice que usa la libreria de colas y nos deja usar los metodos unicos para este tipos de TDA


public class ordenRadix {//Inicio de la clase ordenRadix

    public static int[] ordenacionRadix(int[] vec) {//inicio del metodo statico entero ordenacion Radix que contiene el valor vec que es un array

        int rep = 1; // cantidad de repeticiones

        int numBytes = 4; // número de bytes a desplazar

        int numColas = (int) Math.pow(2, numBytes);


        // Creación de las colas

        Queue[] cola = new LinkedList[numColas];

        for (int i = 0; i < numColas; i++) {//for que nos dice que recorre la cola donde cada repeticion realiza dentro de la cola una lista enlazada

            cola[i] = new LinkedList();

        }

        int div = 0;//declara un valor denominado div de tipo entero y lo inicializa con 0

        for (int i = 0; i < rep; i++) {

            // En esta parte recorre el vector para guardar cada valor en la cola

            for (int numero : vec) {

                // Busca el mayor número del vector

                if (i == 0) {

                    if (numero > rep) {

                        rep = numero;

                    }

                }

                // Calcula en que cola debe ir cada número

                int numCola = (numero >> div) & 0xf;

                cola[numCola].add(numero);
```

```
            }
            div = div + numBytes;
            // Recorre cada cola para colocar cada elemento en el vector
            int j = 0;
            for (Queue c : cola) {
                while (!c.isEmpty()) {
                    vec[j++] = (int) c.remove();
                }
            }
            // La primera vez se actualiza el número de veces que debe ejecutar el proceso
            if (i == 0) {
                rep = (int) (Math.log(rep) / Math.log(numColas)) + 1;
            }
        }

        return vec;//regresa el valor de vec
    }//fin del metodo ordenacionRadix

    public static void imprimirVector(int vec[]){
        //for que repite hasta que i recorra el vector hasta que el vector no durante i no sea igual o mayor al tamaño del array vec[]
        for(int i=0;i<vec.length;i++){
            System.out.print(vec[i]+" ");//muestra en pantalla el valor dentro del array vec[i] y lo que este dentro de la posicion i
        }
    }//fin del metodo imprimirVector
}//fin de la clase ordenRadix
```

Clase Ordenarxd:

Esta linea nos dice que esta clase se encuentra en el paquete busquedabinariapresentacion

Importe de la libreria ActionEvent que nos sirve para dar haciones a los botones de la aplicacion

Importe de la libreria que nos permite hacer que los botones llamen a una accion

importe que nos permite usar todos los metodos de la clase Swing que como ejemplo estan el JButton y JTextArea

inicio de la clase Ordenarxd que extiende la clase JFrame que viene en java

declaracion de las variables que se van a usar como el JTextField para poner el texto

el JButton para añadir botones al programa

JTextArea para el mostrar los datos en una tabla

llamada a la clase Radix con una instancia radix

dato privado de tipo entero el cual es el array numbers

dato de tipo entero resultado

metodo Ordenarxd o el constructor de este

esta nos dice que la operacion por defecto es cerrar el programa

Esta linea nos sirve para poner el título del programa

El tamaño del programa donde 500 es el ancho y 300 es el alto

que el fondo sea nulo

constructor de los datos anteriormente mostrados donde se les añade un nombre

esta nos dice que los valores anteriormente van a tener el tamaño y estara en la ubicacion

constructor del array numbers

método que sirve para darle una función al botón addButton

método que sirve para realizar la función dentro de las llaves

llamada al método addNumero();

fin del método actionPerformed

fin del método addButton.addActionListener

método que sirve para darle una función al botón simularButton

método que sirve para realizar la función dentro de las llaves

llamada al método sumulateBubbleSort();

fin del método actionPerformed

fin del método simularButton.addActionListener

método que sirve para darle una función al boton buscaButton

método que sirve para realizar la función dentro de las llaves

aquí el valor entero xd va a tomar el valor ingresado en la ventana

resultado va a tomar el valor del método busquedaBinaria

método que muestran los resultados de la búsqueda

fin del método actionPerformed

fin del método buscaButton.addActionListener

add() significa que se añade lo que este dentro a la ventana principal

esta línea hace que se vuelva visible la ventana del programa

fin del constructor

método addNumero que nos permite añadir números al programa

indica que este intente realizar lo siguiente

hace que el valor number de tipo entero tome el valor de lo que el usuario ingrese dentro de la ventana

esta linea dice que el InputField estará vacío

si numbers esta vacío

numbers será igual a number

si no

se creará un nuevo array que tomará el valor del tamaño de numbers + 1

línea que pega los datos del array en la tabla

el array newArray toma el valor de number en la posición del tamaño del array numbers

numbers sera igual a newArray

fin del else

método que recarga la tabla de resultados

catch dice que si el programa tiene un error realize lo siguiente

aquí muestra en una ventana de error

fin del catch

fin del metodo addNumero()

metodo que representa el radixsort

si el array number no es nulo y el tamaño del array es mayor a 1 hacer lo siguiente

llama de la clase Radix el metodo ordenacionRadix y con el array numbers

metodo que actualiza la tabla

sino realizar lo siguiente

mensaje de error en caso de no cumplir lo anterior

fin del else

fin del metodo simulateBubbleSort()

Este método toma como parámetros un arreglo de elementos

ordenados y un valor a buscar, y devuelve la posición del valor en el arreglo o -1 si no se encuentra.

declaracion y inicializacion de las variables donde elementos[] es el array utilizado y x es el valor separado

l inicializa en 0 y r en la cantidad de elementos del array menos uno

l significa Left (izquierda) osea el valor de la posicion inicial del array

r significa Right (derecha) osea el valor de la posicion del tamaño del array menos 1

buque while (mientras) que nos dice que va a seguir haciendose mientras l sea menor o igual a r

se declara un valor entero "m" que sera igual a la operacion l + (r - l) / 2

un if que nos dice que devolvera verdadero si la posicion m del array sea igual a x

en caso de ser positivo el valor regresado sera m

if que nos dice que en caso de que el valor de la posicion de m del array sea menor que x

este hara que l sea igual al valor de m + 1

else (demas) que nos dice que si el ultimo if que se realizo tambien debe realizar

Que r tome el valor de m - 1

al cerrar el bucle while este retornara el valor de -1 y si ninguna de las otras funciones se cumplia se retorna

fin del metodo busqueda binaria

Metodo que actualiza la tabla de resultArea

un StringBuilder que crea una instancia sb

este dice que el string sb va a tener la forma base Numeros:

for que dice que el valor num va a tomar el valor de numbers

esta linea agrega a sb el num y que se va a separar " "

fin del for

se añade al resultArea el texto de sb.toString()

fin del metodo updateResultArea()

metodo que imprime el vector / array

for que recorre el array

imprime en consola el contenido del array

fin del for

fin del metodo imprimirVector()

metodo que muestra el resultBusqueda

if que verifica que resultado no sea igual a -1

muestra en consola que no se encontro el elemento buscado

sino

muestra que es en contro el elemento y la posicion en el que esta

fin del metodo resultBusqueda()

fin de la clase OrdenarxdCodigo de la clase Ordenarxd:

package busquedabinariapresentacion;// esta linea nos dice que esta clase se encuentra en el paquete busquedabinariapresentacion

import java.awt.event.ActionEvent;//Importe de la libreria ActionEvent que nos sirve para dar haciones a los botones de la aplicacion

import java.awt.event.ActionListener;//Importe de la libreria que nos permite hacer que los botones llamen a una accion

import javax.swing.*;//importe que nos permite usar todos los metodos de la clase Swing que como ejemplo estan el JButton y JTextArea

```
public class Ordenarxd extends JFrame {//inicio de la clase Ordenarxd que extiende la clase JFrame que viene en java

    private JTextField InputField;//declaracion de las variables que se van a usar como el JTextField para poner el texto

    private JButton addButton;//el JButton para añadir botones al programa

    private JButton simularButton;

    private JButton buscaButton;

    private JTextArea resultArea;//JTextArea para el mostrar los datos en una tabla

    Radix radix = new Radix();//llamada a la clase Radix con una instancia radix

    private int[] numbers;//dato privado de tipo entero el cual es el array numbers

    int resultado;//dato de tipo entero resultado


    public Ordenarxd(){//metodo Ordenarxd o el constructor de este

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//esta nos dice que la operacion por defecto es cerrar el programa

        setTitle("Ordenamiento Radix y Busqueda Binaria JuanJ");//Esta linea nos sirve para poner el titulo del programa

        setSize(500,300);//El tamaño del programa donde 500 es el ancho y 300 es el alto

        setLayout(null);//que el fondo sea nulo

        InputField = new JTextField();//constructor de los datos anteriormente mostrados donde se les añade un nombre

        addButton = new JButton("Añadir");

        simularButton = new JButton("Ordenar");

        buscaButton = new JButton("Busqueda");

        resultArea = new JTextArea();


        InputField.setBounds(20,20,150,30);//estas nos dice que los valores anteriormente van a tener el tamaño y estara en la ubicacion

        addButton.setBounds(180, 20, 90, 30);

        simularButton.setBounds(280, 20, 90, 30);

        buscaButton.setBounds(380, 20, 90,30);
```

```
        resultArea.setBounds(20, 70, 350, 150);

        resultArea.setEditable(false);

        numbers= new int[0];// constructor del array numbers

        addButton.addActionListener(new ActionListener() {//metodo que sirve para darle una funcion al boton addButton

            @Override

            public void actionPerformed(ActionEvent e) {//metodo que sirve para realizar la funcion dentro de las llaves

                addNumero();//llamada al metodo addNumero();

            }//fin del metodo actionPerformed

        });//fin del metodo addButton.addActionListener


        simularButton.addActionListener(new ActionListener() {//metodo que sirve para darle una funcion al boton simularButton

            @Override

            public void actionPerformed(ActionEvent e) {//metodo que sirve para realizar la funcion dentro de las llaves

                simulateBubbleSort();//llamada al metodo sumulateBubbleSort();

            }//fin del metodo actionPerformed

        });//fin del metodo simularButton.addActionListener


        buscaButton.addActionListener(new ActionListener() {//metodo que sirve para darle una funcion al boton buscaButton

            @Override

            public void actionPerformed(ActionEvent e) {//metodo que sirve para realizar la funcion dentro de las llaves

                int xd = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el número a buscar"));//aqui el valor entero xd va a tomar el valor ingresado en la ventana

                resultado = busquedaBinaria(numbers,xd);//resultado va a tomar el valor del metodo busquedaBinaria

                resultBusqueda();//metodo que muestran los resultados de la busqueda

            }//fin del metodo actionPerformed

        });//fin del metodo buscaButton.addActionListener
```

```
        add(resultArea);//add() significa que se añade lo que este dentro a la ventana principal
        add(InputField);
        add(addButton);
        add(simularButton);
        add(buscaButton);



        setVisible(true);//esta linea hace que se vuelva visible la ventana del programa
    }//fin del constructor


    private void addNumero(){//metodo addNumero que nos permite añadir numeros al programa
        try{//indica que este intente realizar lo siguiente
            int number= Integer.parseInt(InputField.getText());// hace que el valor number de tipo entero tome el valor de lo que el usuario ingrese dentro de la ventana
            InputField.setText("");//esta linea dice que el InputField estara vacio
            if(numbers==null){//si numbers esta vacio
                numbers = new int[]{number};//numbers sera igual a number
            }else{//si no
            int[] newArray = new int[numbers.length+1];//se creara un nuevo array que tomara el valor del tamaño de numbers + 1
            System.arraycopy(numbers, 0, newArray, 0, numbers.length);//linea que pega los datos del array en la tabla
            newArray[numbers.length]=number;//el array newArray toma el valor de number en la posicion del tamaño del array numbers
            numbers=newArray;//numbers sera igual a newArray
            }//fin del else
            updateResultArea();//metodo que recarga la tabla de resultados
            //
        }catch(NumberFormatException e){//catch dice que si el programa tiene un error realize lo siguiente
```

```
            JOptionPane.showMessageDialog(this, "Error al ingresar un
Numero","Error",JOptionPane.ERROR_MESSAGE);//aqui muestra en una ventana de error

        }//fin del catch

    }//fin del metodo addNumero()


    private void simulateBubbleSort(){//metodo que representa el radixsort

        if(numbers !=null&& numbers.length > 1){//si el array number no es nulo y el tamaño del
array es mayor a 1 hacer lo siguiente

            Radix.ordenacionRadix(numbers);//llama de la clase Radix el metodo ordenacionRadix y
con el array numbers

            updateResultArea();//metodo que actualiza la tabla

        }else{//sino realizar lo siguiente

            JOptionPane.showMessageDialog(this, "Agregue al menos dos numeros antes de simular el
ordenamiento.", "Advertencia", JOptionPane.WARNING_MESSAGE);//mensaje de error en caso
de no cumplir lo anterior

        }//fin del else

    }//fin del metodo simulateBubbleSort()


    public int busquedaBinaria(int elementos[], int x){//Este método toma como parámetros un
arreglo de elementos

          //ordenados y un valor a buscar, y devuelve la posición del valor en el arreglo o -1 si no se
encuentra.

        //declaracion y inicializacion de las variables donde elementos[] es el array utilizado y x es el
valor separado

        int l = 0, r = elementos.length - 1; //l inicializa en 0 y r en la cantidad de elementos del array
menos uno

                                //l significa Left (izquierda) osea el valor de la posicion inicial del array

                                //r significa Right (derecha) osea el valor de la posicion del tamaño del array
menos 1

        while (l <= r) {// buque while (mientras) que nos dice que va a seguir haciendose mientras l sea
menor o igual a r

            //se declara un valor entero "m" que sera igual a la operacion l + (r - l) / 2

            int m = l + (r - l) / 2;

            //un if que nos dice que devolvera verdadero si la posicion m del array sea igual a x
```

```
            if (elementos[m] == x)
                //en caso de ser positivo el valor regresado sera m
                return m;
            //if que nos dice que en caso de que el valor de la posicion de m del array sea menor que x
            if (elementos[m] < x)
                //este hara que l sea igual al valor de m + 1
                l = m + 1;
            // else (demas) que nos dice que si el ultimo if que se realizo tambien debe realizar
            else
                //Que r tome el valor de m - 1
                r = m - 1;
        }
        //al cerrar el bucle while este retornara el valor de -1 y si ninguna de las otras funciones se cumplia se retorna
        return -1;
    }//fin del metodo busqueda binaria

    private void updateResultArea(){//Metodo que actualiza la tabla de resultArea
        StringBuilder sb = new StringBuilder();//un StringBuilder que crea una instancia sb
        sb.append("Numeros: ");//este dice que el string sb va a tener la forma base Numeros:
        for(int num:numbers){//for que dice que el valor num va a tomar el valor de numbers
            sb.append(num).append(" ");//esta linea agrega a sb el num y que se va a separar " "
        }//fin del for
        resultArea.setText(sb.toString());//se añade al resultArea el texto de sb.toString()
    }//fin del metodo updateResultArea()

    public static void imprimirVector(int vec[]){//metodo que imprime el vector / array
        for(int i=0;i<vec.length;i++){//for que recorre el array
            System.out.print(vec[i]+" ");//imprime en consola el contenido del array
        }//fin del for
```

```
    }//fin del metodo imprimirVector()


    private void resultBusqueda(){//metodo que muestra el resultBusqueda

        if (resultado == -1)//if que verifica que resultado no sea igual a -1

            System.out.println("No se ha encontrado el elementos buscado");//muestra en consola que no se encontro el elemento buscado

        else//sino

            System.out.println("Elemento encontrado en la posicion: "+ resultado );//muestra que es en contro el elemento y la posicion en el que esta

    }//fin del metodo resultBusqueda()

}//fin de la clase Ordenarxd
```