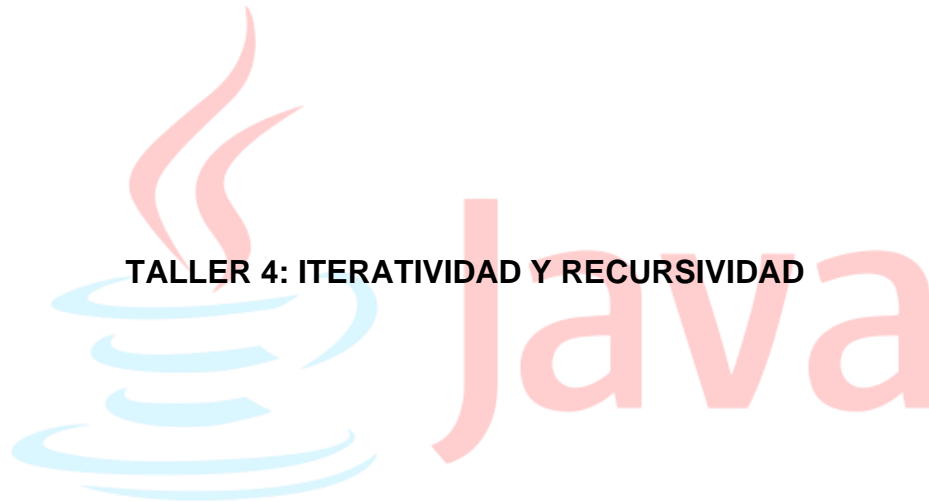


UNIVERSIDAD DE SAN BUENAVENTURA


ANALISIS Y DISEÑO DE ALGORITMOS



LUIS FELIPE VELASCO TAO

26 DE FEBRERO

2020

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

PARTE 1

1. ¿Qué es un algoritmo recursivo?

Un algoritmo recursivo es todo aquel que en su ejecución tiende a llamarse a si mismo, con el fin de cumplir con una condición que termine el ciclo de llamadas a si mismo. Este tipo de algoritmos se apoya en sentencias **If** para evaluar el límite de ejecución de sí mismo.


2. ¿Qué es un algoritmo iterativo?

Un algoritmo iterativo es todo aquel que en su ejecución trabaja de forma lineal, sin usarse o llamarse a sí mismo, apoyándose de estructuras de **bucle** (while, do – while, for, for – each) para su ejecución.

3. ¿Cuáles son las principales diferencias entre un algoritmo recursivo y un algoritmo iterativo?

- Los algoritmos recursivos necesitan un análisis mas grande para su construcción (uso de modelos matemáticos)
- Los algoritmos iterativos no requieren conocimientos avanzados.
- Los algoritmos recursivos controlan su ejecución mediante sentencias **If** o **Switch**, las cuales evalúan el cumplimiento de una condición para el fin de ejecución.
- Los algoritmos iterativos controlan su ejecución mediante el uso de estructuras de bucle (principalmente) en donde de forma iterativa (el uso de una variable que debe cambiar hasta cumplir una condición limitante) solucionan el problema por el que fueron diseñados.
- Los algoritmos recursivos para su ejecución utilizan mas recursos en su ejecución (espacio en memoria RAM).
- En la composición de algoritmos recursivos pueden estar presentes estructuras de bucle, estas no determinaran el fin de ejecución del estos.

4. ¿Cómo funciona un algoritmo recursivo? De un ejemplo.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------


1. El algoritmo recibe un conjunto de datos, de los cuales alguno o alguno de estos será el limitante de ejecución del algoritmo.
2. En la estructura del algoritmo, los datos limitantes se evaluarán en una/s sentencia/s **If**:
 - a. Si la o las condiciones no se cumplen, se ejecutará n líneas de código hasta que se vuelva a hacer la llamada del mismo algoritmo variando los datos limitantes del algoritmo.
 - b. Si la o las condiciones se cumplen, se ejecutarán o no n líneas de código para finalizar el algoritmo.

Ejemplo

```
// Se reciben los valores a, b, s y i, esta última será el
// dato limitante

public int algoritmoRecursivo (int a, int b, int s, int i) {
    // Se evalúa si el dato i se ha cumplido
    if (i == 3) {
        // Si la condición se cumple, se retornará el
        //siguiente resultado.
        return (a*b) *s;
    } else {
        // Si no se cumple la condición se operarán los
        //datos
        s=(a*b);
        // Se llamará a la función cambiando el dato
        //limitante
        return algoritmoRecursivo(a,b,s,i+1);
    }
}
```

5. ¿Cómo funciona un algoritmo iterativo? De un ejemplo.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------


1. Un algoritmo iterativo recibe, o no, n datos.
2. Los datos son evaluados las veces que sean necesarias
 - a. Si no se necesita operar mas de una vez, se operan si usar bucles
 - b. Si se necesita operar mas una vez, se evalúa cual estructura de bucle es la adecuada para manipular los datos.


Ejemplo


```
// Se reciben los datos a operar
public int algoritmoIterativo (int a, int b) {
    int s = 1;
    // Se requiere operar los datos 3 veces, por lo tanto
    // se usa un ciclo for para operar las variables
    for (int i = 0; i < 3; i++) {
        s *= (a*b);
    }
    return s;
}
```


6. Explique que tipos de recursividad existen, en que consiste cada tipo y de ejemplos de cada uno.

TIPO	EJEMPLO
<p>Recursividad simple: En este tipo de algoritmos solo se hace una llamada del algoritmo a si mismo, dando la posibilidad de traducir de forma fácil este</p>	<pre>private static int factorial(int num, int i, int r) { if (i != num) { return factorialRecursivo(num, i + 1, r * i); } else {</pre>

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

<p>tipo de algoritmos a una forma iterativa.</p>	<pre> return r * i; } }</pre>
<p>Recursividad múltiple: En este tipo de algoritmos, en su cuerpo, el algoritmo se llama si mismo varias veces en su ejecución.</p> 	<pre> TORRES DE HANOI public static void main(String[] args) { Scanner s = new Scanner (System.in); System.out.println("TORRES DE HANOI "); torresDeHanoi(4,1,2,3) } public static void torresDeHanoi(int d, int inicio, int auxiliar, int fin,) { if (d != 0) { torresDeHanoi(d - 1, inicio, fin, auxiliar); System.out.println("Mover Disco #" + d + " de " + inicio + " a " + fin +); torresDeHanoi(d - 1, auxiliar, inicio, fin); } } SERIE FIBONACCI public static void main(String[] args) { Scanner s = new Scanner (System.in); System.out.println("FIBONACCI = "+ fibonacci (s.nextInt())); } public static int fibonacci(int n){ if (n <= 2){ </pre>


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

	<pre> return 1; } else{ return fibonacci(n-1)+fibonacci(n-2); } } </pre>
<p>Recursividad anidada: En este tipo de algoritmos, al realizar el llamado de la misma función, esta se usa a si misma como parte de los parámetros de entrada.</p> 	<pre> public static void main(String[] args) { Scanner s = new Scanner (System.in); System.out.println("ACKERMAN = "+ackermann(s.nextInt(),s.nextInt())); } public static int ackermann(int m, int n) { if (m == 0) { return n + 1; } else if (n == 0) { return ackermann(m - 1, 1); } else { return ackermann(m - 1, ackermann(m, n - 1)); } } } </pre>
<p>Recursividad cruzada o indirecta: En estos algoritmos, se hace una llamada a si mismo al llamar a otra función, esto puede generar que entre los dos algoritmos se</p>	<pre> public static void main(String[] args) { Scanner s = new Scanner (System.in); System.out.println("PAR/INPAR = "+par(s.nextInt())); } public static String par(int num){ </pre>

hagan llamadas mutuas
para la solución del
problema.

```
if (num == 0){  
    return "PAR";  
} else {  
    return inpar(num-1);  
}  
  
private static String inpar(int num) {  
    if (num == 0){  
        return "INPAR";  
    } else {  
        return par(num-1);  
    }  
}
```



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------


PARTE 2

1. Algoritmo que calcule la factorial de un número.

```
public class Factorial {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int num = s.nextInt();
        System.out.println("Factorial iterativo = " +
            factorialIterativo(num) + "\n"
            + "Factorial Recursivo = " + factorialRecursivo(num,
2, 1));
    }
    private static int factorialIterativo(int num) {
        int r = 1;
        for (int i = 2; i <= num; i++) {
            r *= i;
        }
        return r;
    }
    private static int factorialRecursivo(int num, int i, int r) {
        if (i != num) {
            return factorialRecursivo(num, i + 1, r * i);
        } else {
            return r * i;
        }
    }
}
```

ITERATIVO


	NUM	I	R	I<=NUM
24	3	0	1	
25	3	2	1	false
26	3		2	
25	3	3	2	true
26	3		6	
28	3		6	

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos	TALLER 4: RECURSIVIDAD	2020 - 1
--	---	---	-----------------

RECURSIVO

	num	i	r	i != num
20	3	2	1	
32				true
33	3	3	2	
32				false
35			6	



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

2. Algoritmo que calcule un número de la serie Fibonacci.


```

public class Fibonacci {
    static Scanner s = new Scanner(System.in);
    public static void main(String[] args) {
        int num = s.nextInt();
        System.out.println("FIBONACCI ITERATIVA = " +
fibonacciIterativo(num) + "\n"
                        + "FIBONACCI RECURSIVO = " + fibonacciRecursivo(num,
0, 1, 0) + "\n")    }

    private static int fibonacciIterativo(int num) {
        if (num == 0) {
            return 0;
        } else {
            int a = 0, b = 1, c = 0;
            for (int i = 1; i < num; i++) {
                c = a + b;
                a = b;
                b = c;
            }
            return b;
        }
    }

    private static int fibonacciRecursivo(int num, int a, int b, int
i) {
        if (num == 0) {
            return 0;
        } else if (i < num) {
            return fibonacciRecursivo(num, b, a + b, i + 1);
        } else {
            return a;
        }
    }
}


```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

ITERATIVO


	num	i	a	b	c	i < num+1	num == 0	num <= 2
20	6							
26	6						false	
28	6							false
31	6			0	1	0		
32	6	2				true		
33	6	2				1		
34-35	6	2	1	1				
32	6	3				true		
33	6	3				2		
34-35	6	3	1	2				
32	6	4				true		
33	6	4				3		
34-35	6	4	2	3				
32	6	5				true		
33	6	5				5		
34-35	6	5	3	5				
32	6	6				true		
33	6	6				8		
34-35	6	6	5	8				
32	6	7				false		
37								
RETURN 8								

RECURSIVO

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

	num	a	b	i	num == 0	i < num
20	6	0	1	0		
39					false	
41						true
42		1	1	1		
39					false	
41						true
42		1	2	2		
39					false	
41						true
42		2	3	3		
39					false	
41						true
42		3	5	4		
39					false	
41						true
42		5	8	5		
39					false	
41						true
42		8	13	6		
39					false	
41						false
44	RETURN					8

3. Algoritmo que permita invertir un número. Ejemplo: Entrada: 56789 Salida:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

98765.

```

public class InvertirNumero {

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        int num = s.nextInt();

        System.out.println("INVERTIR NUMERO ITERATIVO = 
"+invertirNumeroIterativo(String.valueOf(num))+""

            + "\nINVERTIR NUMERO RECURSIVO = " + 
invertirNumeroRecursivo(String.valueOf(num),

                "", String.valueOf(num).length() - 1));

    }

    private static String invertirNumeroIterativo(String num) {

        String numInv = "";

        for (int i = num.length() - 1; i >= 0; i--) {

            numInv += num.charAt(i);

        }

        return numInv;

    }

    private static String invertirNumeroRecursivo(String num, String
numInv, int i) {

        if (i == 0) {

            return numInv.concat(String.valueOf(num.charAt(i)));

        } else {


            return invertirNumeroRecursivo(num,
numInv.concat(String.valueOf(num.charAt(i))), i - 1);

        }

    }

}

```


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

ITERATIVO

	num	numInv	i	i >= 0
24	"56789"	""		
25			4	True
26		"9"		
25			3	True
26		"98"		
25			2	True
26		"987"		
25			1	True
26		"9876"		
25			0	True
26		"98765"		
25			-1	False
28	return			"98765"

RECURSIVO

	num	numInv	i	i == 0
20	"56789"	""	4	
32				False
35	"56789"	"9"	3	
32				False
35	"56789"	"98"	2	
32				False
35	"56789"	"987"	1	
32				False
35	"56789"	"9876"	0	
32				True
33	"56789"	"98765"		
	return			"98765"


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

4. Algoritmo que permita sumar los elementos de un vector.

```

public class SumarNumerosArreglo {
    static Scanner s = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.println("SUMA VALORES ARREGLO ITERATIVA = " +
sumarElementosVectorIterativo(llenarArreglo(),0)+"SUMA VALORES ARREGLO RECURSIVA =
" + sumarElementosVectorRecursivo(llenarArregloR(0,null), 0, 0));
    }
    private static int sumarElementosVectorIterativo(int[] elementos, int r) {
        for (int i = 0; i < elementos.length; i++) {
            r += elementos[i];
        }
        return r;
    }
    private static int sumarElementosVectorRecursivo(int[] elementos, int i,int r)
{
    if (i == elementos.length-1){
        return r += elementos[i];
    } else {
        return sumarElementosVectorRecursivo(elementos, i+1, r+elementos[i]);
    }
}
    private static int[] llenarArreglo() {
        int [] elementos = new int [5];
        for (int i = 0; i < elementos.length; i++) {
            elementos[i] = s.nextInt();
        }return elementos;
    }
    private static int[] llenarArregloR(int i, int[] nume) {
        if (i == 0) {
            int[] num = new int[5];
            num[0] = s.nextInt();
            return llenarArregloR(i + 1, num);
        } else if (i < 5) {
            nume[i] = s.nextInt();
            return llenarArregloR(i + 1, nume);
        } else {
            return nume;
        }
    }
}

```


 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos	TALLER 4: RECURSIVIDAD	2020 - 1
--	---	---	-----------------

ITERATIVO

	r	i	i < e.length	e[]	i	elementos
24	0				0	5
25		0	true		1	6
26	5			5	2	47
25		1	true		3	14
26	11			6	4	-5
25		2	true		length = 5	
26	58			47		
25		3	true			
26	72			14		
25		4	true			
26	67			-5		
25		5	false			
28	return			67		

RECURSIVO

	i	r	i == e.length-1	e[]
21	0	0		
32			false	
35	1	5		5
32			false	
35	2	11		6
32			false	
35	3	58		47
32			false	
35	4	72		14
32			true	
33	RETURN	67		-5

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------


5. Algoritmo que muestre el número menor de un vector.

```

public class NumeroMenorArreglo {
    static Scanner s = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.println("NUMERO MENOR DE UN ARREGLO ITERATIVO = " +
        menorDelArregloIterativo(llenarArreglo()));
        System.out.println("NUMERO MENOR DE UN ARREGLO RECURSIVO = " +
        menorDelArregloRecursivo(llenarArregloR(0, null), 0, 0));
    }
    private static int menorDelArregloIterativo(int[] elementos) {
        int men = elementos[0];
        for (int i = 1; i < elementos.length; i++) {
            if (men > elementos[i]) {
                men = elementos[i];
            }
        }
        return men;
    }
    private static int menorDelArregloRecursivo(int[] elementos, int i, int men) {
        if (i == elementos.length) {
            return men;
        } else if (i == 0) {
            return menorDelArregloRecursivo(elementos, i + 1, elementos[i]);
        } else {
            if (men > elementos[i]) {
                return menorDelArregloRecursivo(elementos, i + 1, elementos[i]);
            }
            return menorDelArregloRecursivo(elementos, i + 1, men);
        }
    }
    private static int[] llenarArreglo() {
        int[] elementos = new int[5];
        for (int i = 0; i < elementos.length; i++) {
            elementos[i] = s.nextInt();
        }
        return elementos;
    }

    private static int[] llenarArregloR(int i, int[] nume) {
        if (i == 0) {
            int[] num = new int[5];
            num[0] = s.nextInt();
            return llenarArregloR(i + 1, num);
        } else if (i < 5) {
            nume[i] = s.nextInt();
            return llenarArregloR(i + 1, nume);
        } else {
            return nume;
        }
    }
}

```


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

ITERATIVO


	men	i	i < e.length	e[]	men > e[i]	i	elementos
24	5			5		0	5
25		1	true			1	6
26	5			6	false	2	-47
25		2	true			3	14
26	-47			-47	true	4	-5
27	-47			-47		length = 5	
25		3	true				
26	-47			14	false		
25		4	true				
26	-47			-5	false		
25		5	false				
	return			-47			

RECURSIVO

	i	men	e[]	i == e.length	i == 0	men > e[i]
20	0	0	0			
34	0			false		
36	0				true	
37	1	5	5			
34				false		
36	1				false	
39	1	5	6			false
42	2	5				
34	2			false		
36	2				false	
39	2	5	-47			true
40	3	-47				
34				false		
36					false	
39	3	-47	14			false
42	4	-47				
34	4			false		
36	4				false	
39	4	-47	-5		false	
42	5	-47				
34	5			true		
36	return		-47			


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

6. Algoritmo que permita sumar, restar, multiplicar, dividir y calcular la potencia



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

dos números naturales.

```


public class Operaciones {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int a = s.nextInt(), b = s.nextInt();
        operaciones(a, b);
        operacionesRecurisvas(a, b, 0);
    }

    private static void operacionesRecurisvas(int a, int b, int oper) {
        switch (oper) {
            case 4:
                System.out.println(a + "^2 = " + (a * a) + " " + b + "^2 = " + (b
* b));
                break;
            case 0:
                System.out.println(a + " + " + b + " = " + (a + b));
                operacionesRecurisvas(a, b, oper + 1);
                break;
            case 1:
                System.out.println("OPERACIONES RECURSIVAS\n" + a + " - " + b + " =
" + (a - b));
                operacionesRecurisvas(a, b, oper + 1);
                break;
            case 2:
                System.out.println(a + " x " + b + " = " + (a * b));
                operacionesRecurisvas(a, b, oper + 1);
                break;
            case 3:
                double div = (double) a / b;
                System.out.println(a + " / " + b + " = " + div);
                operacionesRecurisvas(a, b, oper + 1);
                break;
            default:
                break;
        }
    }

    private static void operaciones(int a, int b) {
        double div = (double) a / b;
        System.out.printf("OPERACIONES ITERATIVAS\n %d + %d = %d \n"
+ " %d - %d = %d \n"
+ " %d x %d = %d \n"
+ " %d / %d = %.2f \n"
+ " %d ^2 = %d      %d ^2 = %d \n",
a, b, (a + b), a, b, (a - b), a, b, (a * b), a, b, div, a, (a *
a), b, (b * b));}

```


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

ITERATIVO

	a	b	div	
	10	7		
51			1,428571429	
52	<p>"10 - 7 = 3 10 x 7 = 70 10 / 7 = 1.4285714285714286 10^2 = 100 7^2 = 49"</p>			

RECURSIVO

	a	b	oper	switch(oper)	div
	10	7	0		
				0	
	"10 + 7 = 17"				
	10	7		1	
				1	
	"10 - 7 = 3"				
	10	7		2	
				2	
	"10 x 7 = 70"				
	10	7		3	
				3	
					1,428571429
	"10 / 7 = 1,428571429"				
	10	7		4	
				4	
	"10^2 = 100 7^2=79"				


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

7. Algoritmo que permita buscar un elemento tipo **String** en una matriz de N*N

```

public class BuscarCadena {
    static Scanner s = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.println("BUSCAR ELEMENTO ITERATIVO = " +
            buscarElemento(llenarMatriz(s.nextInt()), s.next().toUpperCase())+"BUSCAR ELEMENTO
            RECURSIVO = " + buscarelementoRecursivo(llenarMatrizR(0, 0, s.nextInt(), null),
            s.next().toUpperCase(), 0, 0));
    }
    private static String[][] llenarMatriz(int n) {
        String[][] matriz = new String[n][n];
        for (String[] matriz1 : matriz) {
            for (int j = 0; j < matriz[0].length; j++) {
                matriz1[j] = s.next().toUpperCase();
            }
        }
        return matriz;
    }
    private static String[][] llenarMatrizR(int i, int j, int ext, String[][]
matriz) {
        if (i == 0 && j == 0) {
            String[][] m = new String[ext][ext];
            m[i][j] = s.next().toUpperCase();
            return llenarMatrizR(i, j + 1, ext, m);
        } else if (i != ext) {
            matriz[i][j] = s.next().toUpperCase();
            if (j == ext - 1) {
                return llenarMatrizR(i + 1, 0, ext, matriz);
            } else {
                return llenarMatrizR(i, j + 1, ext, matriz);
            }
        } else {
            return matriz;
        }
    }
    private static String buscarElemento(String[][] matriz, String elemento) {
        for (int i = 0; i < matriz.length; i++) {
            for (int j = 0; j < matriz[0].length; j++) {
                if (matriz[i][j].equals(elemento)) {
                    return elemento + " SE ENCUENTRA EN LA POSICIÓN [" + (i + 1) +
                        "]-[" + (j + 1) + "];"
                }
            }
        }
        return "NO SE ENCONTRO A " + elemento + " EN LA MATRIZ";
    }
}

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

```

private static String buscarelementoRecurso(String[][] matriz, String
elemento, int i, int j) {
    if (matriz[i][j].equals(elemento)) {
        return elemento + " SE ENCUENTRA EN LA POSICIÓN [" + (i + 1) + "]-[" +
(j + 1) + "];"
    } else if (i == matriz.length - 1 && j == matriz.length - 1) {
        return "NO SE ENCONTRO A " + elemento + " EN LA MATRIZ";
    } else {
        if (j == matriz.length - 1) {
            return buscarelementoRecurso(matriz, elemento, i + 1, 0);
        } else {
            return buscarelementoRecurso(matriz, elemento, i, j + 1);
        }
    }
}
}
}


```

ITERATIVO

	elemento	i	j	matriz[i][j]	i < m.lenght	j < m.lenght	e == m[i][j]	MATRIZ	"esfera"	"ventana"
55	"telefono"								"telefono"	"tarea"
56			0		true				(0:0)	(0:1)
57			0	0		true			(1:0)	(1:1)
58				"esfera"			false			
57			0	1		true				
58				"ventana"						
57			0	2		false				
56			1		true					
57			1	0		true				
	return			"telefono"			true			
	"TELEFONO SE ENCUENTRA EN LA POSICIÓN [2]-[1]"									

RECURSIVO

	elemento	i	j	matriz[i][j]	m[i][j]==e	i == m.lenght	j == m.lenght	MATRIZ	"esfera"	"ventana"
67	"telefono"		0	0	false				"telefono"	"tarea"
69			0	0		false			(0:0)	(0:1)
72			0	0			false		(1:0)	(1:1)
75			0	1						
67	telefono		0	1	"ventana"	false				
69			0	1		false				
72			0	1			true			
73			1	0						
67	"telefono"		1	0	"telefono"	true				
68	return									
	"TELEFONO SE ENCUENTRA EN LA POSICION [2][1]"									

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

PARTE 3

Ejercicio 1 - Palíndromo

Escriba un algoritmo recursivo que verifique si dado un **String** es palíndromo o no (Un Palíndromo es una palabra, frase, número y otra secuencia de caracteres los cuales se leen de la misma manera de derecha a izquierda o de izquierda a derecha. Como por ejemplo **madam** o **racecar**.


```
public class Palindromo {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String palabra = s.next();
        System.out.println(palindromo(palabra, "", palabra.length() - 1));
    }

    private static String palindromo(String palabra, String palabraInv, int i) {
        if (i == 0) {
            if (palabra.equals(palabraInv.concat(String.valueOf(palabra.charAt(i))))) {
                return palabra + " si es un palindromo";
            } else {
                return palabra + " no es un palindromo";
            }
        } else {
            return palindromo(palabra, palabraInv.concat(String.valueOf(palabra.charAt(i))),
                i - 1);
        }
    }
}
```

Ejercicio 2- Permutaciones

Desarrollar un algoritmo que muestre por consola todas las permutaciones que se pueden realizar con los caracteres de una cadena de cualquier longitud. La cadena la ingresa el usuario por consola o por línea de comandos y no tendrá caracteres repetidos.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

Por ejemplo, si la cadena ingresada fuese “ABC” entonces el programa debería arrojar las siguientes 6 líneas, en cualquier orden:

ABC

ACB

BAC

BCA

CAB

CBA

Y si la cadena fuese “ABCD” entonces la salida debería ser:

ABCD

ABDC

ACBD


ACDB

ADBC

ADCB

BACD



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

BADC

```
public class Permutacion {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String elemento = s.next();
        permutacion(elemento, "", elemento.length(), elemento.length());
    }

    private static void permutacion(String elemento, String permutado, int
letra, int ext) {
        if (letra == 0) {
            System.out.println(permutado);
        } else {
            for (int i = 0; i < ext; i++) {
                if (!permutado.contains(String.valueOf(elemento.charAt(i))))
                {
                    permutacion(elemento, permutado + elemento.charAt(i),
letra - 1, ext);
                }
            }
        }
    }
}
```

Ejercicio 3- Canción sin vocales


Hacer la lectura de un archivo plano en donde este un texto de una canción, usted debe leer la canción, eliminar con una función recursiva todas las vocales que tenga este texto y volver a escribir un archivo plano con la misma canción, pero sin ninguna vocal. Al final tendrá dos archivos planos, uno con la canción con vocales y otro con la canción sin vocales.

```
public class PuntoCancion {
    public static void main(String[] args) {
        leer();
        leerModificado();
    }
    public static void escribir(String cancion) {
        FileWriter fichero = null;
        PrintWriter pw = null;
        try {
            fichero = new
FileWriter("C:/Users/COMPURED/Documents/NetBeansProjects/iterativosRekursivos
/cancionModificada.txt", true);
            pw = new PrintWriter(fichero);
            pw.println(cancion);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (null != fichero) {
                    fichero.close();
                }
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
    }
    public static void leer() {
        File archivo = null;
        FileReader fr = null;
        BufferedReader br = null;
        try {
            archivo = new
File("C:/Users/COMPURED/Documents/NetBeansProjects/iterativosRekursivos/canci
onOriginal.txt");
            fr = new FileReader(archivo);
            br = new BufferedReader(fr);
            String cancion;
            while ((cancion = br.readLine()) != null) {
                quitarVocales(cancion, "", 0, cancion.length());
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (null != fr) {
                    fr.close();
                }
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
    }
}
```

```
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}

public static void leerModificado() {
    File archivo = null;
    FileReader fr = null;
    BufferedReader br = null;
    try {
        archivo = new
File("C:/Users/COMPURED/Documents/NetBeansProjects/iterativosRecursivos/canci
onModificada.txt");
        fr = new FileReader(archivo);
        br = new BufferedReader(fr);
        String cancion;
        while ((cancion = br.readLine()) != null) {
            System.out.println(cancion);
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            if (null != fr) {
                fr.close();
            }
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}

public static void quitarVocales(String cancion, String cancionMod, int i,
int exts) {
    if (i == exts) {
        escribir(cancionMod);
    } else {
        String aeiou = "aeiou";
        String letra = String.valueOf(cancion.charAt(i)).toLowerCase();
        if (!vocales(letra,aeiou,0,aeiou.length()-1)) {
            quitarVocales(cancion, cancionMod, i + 1, exts);
        } else{
            cancionMod +=letra;
            quitarVocales(cancion, cancionMod, i + 1, exts);
        }
    }
}
```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

```

private static boolean vocales(String letra, String aeiou, int i, int
exts) {
    if (String.valueOf(aeiou.charAt(i)).equals(letra)) {
        return false;
    } else if (i == exts &&
!String.valueOf(aeiou.charAt(i)).equals(letra)) {
        return true;
    } else if (i == exts &&
String.valueOf(aeiou.charAt(i)).equals(letra)) {
        return false;
    } else {
        return vocales(letra, aeiou, i + 1, exts);
    }
}

```

Ejercicio 4 - Palabras repetidas en un poema

Hacer la lectura de un archivo plano en donde este un texto de un poema (mínimo de 4 párrafos), usted debe leer el poema, recorrer el texto e identificar cuantas palabras repetidas hay de cada tipo, cuantos espacios tiene el texto, cuantas consonantes tiene el texto, cuantas vocales tiene el texto y cuantas letras en total.

Ejemplo:

```

Cantidad de vocales: 53
Cantidad de consonantes: 45
Cantidad de espacio: 15
Cantidad de letras: 98
Palabras repetidas:
    Hola: 4 veces
    De: 19 veces
    Para: 22 veces
    Semáforo: 1 vez
    Casa: 3 veces
    Historia:1 vez
    Siempre: 18 veces


```

```
public class PuntoPoema {

    static int v = 0, l = 0, e = 0;
    static ArrayList<claveValor> palabrasEncontradas = new ArrayList<>();
    static String aeiou = "aeiouáéíóú", sign = ",.¿?!|'";

    public static void main(String[] args) {
        leer();
        System.out.println("-----LISTA-----");
        contadorMuestra();
        System.out.println("Palabra repetidas:");
        listarPalabras(0);
    }

    public static void leer() {
        File archivo = null;
        FileReader fr = null;
        BufferedReader br = null;
        try {
            archivo = new
File("C:/Users/COMPURED/Documents/NetBeansProjects/iterativosRecursivos/poema
.txt");
            fr = new FileReader(archivo);
            br = new BufferedReader(fr);
            String verso;
            System.out.println("          Y TE BUSQUE POR PUEBLOS");
            while ((verso = br.readLine()) != null) {
                System.out.println(verso);
                contarVocalesConsonantesEspacioa(verso, 0, verso.length() -
1);
                palabras(verso, 0, verso.length() - 1, "");
            }
            System.out.println("          POR: Jose Marti");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (null != fr) {
                    fr.close();
                }
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
    }
}
```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------




```

    private static void palabras(String verso, int i, int exts, String
palabra) {
        if (i == exts) {
            addLista(palabra);
        } else {
            String letra = String.valueOf(verso.charAt(i)).toLowerCase();
            if (palabra.length() == 1){
                palabra = palabra.toUpperCase();
            }
            if (vocales(letra, aeiou, 0, aeiou.length() - 1) || letras(letra)
&& !espacios(letra)) {
                palabra += letra;
                palabras(verso, i + 1, exts, palabra);
            } else if (!palabra.equals("")) {
                addLista(palabra);
                palabras(verso, i + 1, exts, "");
            }
        }
    }

    public static void addLista(String palabra) {
        if (palabrasEncontradas.isEmpty()) {
            claveValor cl1 = new claveValor(palabra, 1);
            palabrasEncontradas.add(cl1);
        } else if (existsPalabra(0, palabra)) {
            int pos = ubicacionPalabra(0, palabra);
            claveValor cl = new claveValor(palabra,
palabrasEncontradas.get(pos).getNum() + 1);
            palabrasEncontradas.set(pos, cl);
        } else {
            claveValor cl1 = new claveValor(palabra, 1);
            palabrasEncontradas.add(cl1);
        }
    }

    private static void contarVocalesConsonantesEspacioa(String verso, int i,
int exts) {
        if (i == exts) {
            String letra = String.valueOf(verso.charAt(i)).toLowerCase();
            if (vocales(letra, aeiou, 0, aeiou.length() - 1)) {
                v++;
            } else if (espacios(letra)) {
                e++;
            } else if (letras(letra)) {
                l++;
            }
        } else {
            String letra = String.valueOf(verso.charAt(i)).toLowerCase();
            if (vocales(letra, aeiou, 0, aeiou.length() - 1)) {

```

```

        e++;
        contarVocalesConsonantesEspacioa(verso, i + 1, exts);
    } else if (letras(letra)) {
        l++;
        contarVocalesConsonantesEspacioa(verso, i + 1, exts);
    } else {
        contarVocalesConsonantesEspacioa(verso, i + 1, exts);
    }
}
}
private static boolean vocales(String letra, String aeiou, int i, int
exts) {
    if (String.valueOf(aeiou.charAt(i)).equals(letra)) {
        return true;
    } else if (i == exts &&
!String.valueOf(aeiou.charAt(i)).equals(letra)) {
        return false;
    } else if (i == exts &&
String.valueOf(aeiou.charAt(i)).equals(letra)) {
        return true;
    } else {
        return vocales(letra, aeiou, i + 1, exts);
    }
}

private static boolean espacios(String letra) {
    return letra.equals(" ");
}


private static boolean letras(String letra) {
    return !signos(letra, sign, 0, sign.length() - 1);
}
private static boolean signos(String letra, String sign, int i, int exts)
{
    if (String.valueOf(sign.charAt(i)).equals(letra)) {
        return true;
    } else if (i == exts &&
!String.valueOf(sign.charAt(i)).equals(letra)) {
        return false;
    } else if (i == exts && String.valueOf(sign.charAt(i)).equals(letra))
{
        return true;
    } else {
        return signos(letra, sign, i + 1, exts);
    }
}
private static void contadorMuestra() {
    System.out.println("-VOCALES: " + v + "\n"
        + "-CONSONANTES: " + l + "\n"

```

```
private static void listarPalabras(int i) {
    if (i == 0){
        Collections.sort(palabrasEncontradas);
    }
    if (i == palabrasEncontradas.size() - 1) {
        System.out.println(" - " +
palabrasEncontradas.get(i).toString());
    } else {
        System.out.println(" - " +
palabrasEncontradas.get(i).toString());
        listarPalabras(i + 1);
    }
}

private static boolean existsPalabra(int i, String palabra) {
    if (palabrasEncontradas.isEmpty()){
        return false;
    } else if (i == palabrasEncontradas.size() - 1 &&
palabrasEncontradas.get(i).getPalabra().equals(palabra)) {
        return true;
    } else if (i == palabrasEncontradas.size() - 1 &&
!palabrasEncontradas.get(i).getPalabra().equals(palabra)) {
        return false;
    } else if (palabrasEncontradas.get(i).getPalabra().equals(palabra)) {
        return true;
    } else {
        return existsPalabra(i + 1, palabra);
    }
}

private static int ubicacionPalabra(int i, String palabra) {
    if (palabrasEncontradas.isEmpty()){
        return -2;
    } else if (i == palabrasEncontradas.size() - 1 &&
palabrasEncontradas.get(i).getPalabra().equals(palabra)) {
        return i;
    } else if (i == palabrasEncontradas.size() - 1 &&
!palabrasEncontradas.get(i).getPalabra().equals(palabra)) {
        return -1;
    } else if (palabrasEncontradas.get(i).getPalabra().equals(palabra)) {
        return i;
    } else {
        return ubicacionPalabra(i + 1, palabra);
    }
}
```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

```

public static class Palabra implements Comparable<Palabra>{
    String palabra;
    int num;
    public Palabra(String palabra, int num) {
        this.palabra = palabra;
        this.num = num;
    }

    public String getPalabra() {
        return palabra;
    }

    public int getNum() {
        return num;
    }

    public void setNum(int num) {
        this.num = num;
    }


    @Override
    public String toString() {
        if (num == 1) {
            return palabra + ": " + num + " vez";
        } else {
            return palabra + ": " + num + " veces";
        }
    }

    @Override
    public int compareTo(Palabra t) {
        if (num > t.num){
            return -1;
        } else if (num < t.num) {
            return 1;
        } else {
            return 0;
        }
    }
}

```

Ejercicio 5 – El juego de los discos: La Torre de Hanoi

El juego, en su forma más tradicional, consiste en tres varillas verticales. En una de las varillas se apila un número indeterminado de discos (elaborados de madera) que

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

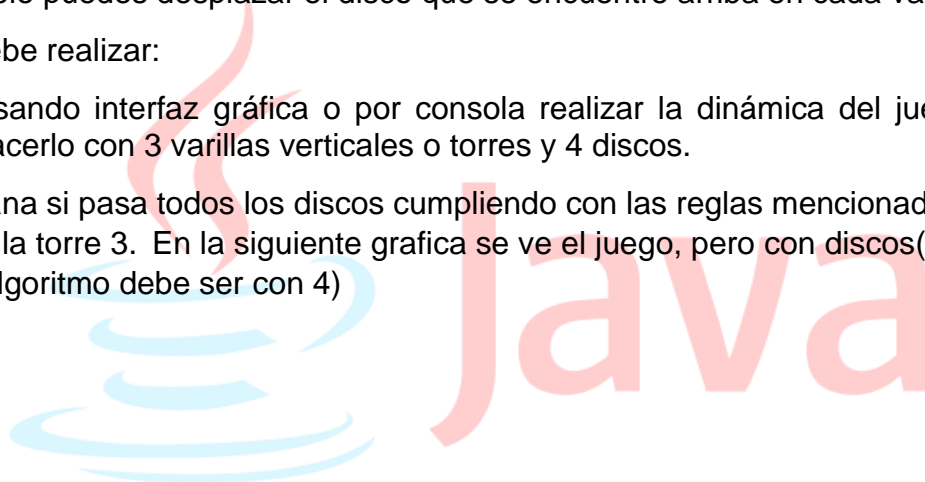
determinará la complejidad de la solución, por regla general se consideran ocho discos. Los discos se apilan sobre una varilla en tamaño decreciente. No hay dos discos iguales, y todos ellos están apilados de mayor a menor radio en una de las varillas, quedando las otras dos varillas vacantes. El juego consiste en pasar todos los discos de la varilla ocupada (es decir la que posee la torre) a una de las otras varillas vacantes. Para realizar este objetivo, es necesario seguir tres simples reglas:


- Sólo se puede mover un disco cada vez.
- Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.
- Sólo puedes desplazar el disco que se encuentre arriba en cada varilla.

Usted debe realizar:

- Usando interfaz gráfica o por consola realizar la dinámica del juego para hacerlo con 3 varillas verticales o torres y 4 discos.

Usted gana si pasa todos los discos cumpliendo con las reglas mencionadas de la torre 1 a la torre 3. En la siguiente grafica se ve el juego, pero con discos (recuerde que su algoritmo debe ser con 4)



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Análisis y Diseño de Algoritmos</p>	<p>TALLER 4: RECURSIVIDAD</p>	<p>2020 - 1</p>
---	---	-----------------------------------	-----------------

```

public class TorresDeHanoi {
    static String disco1 = " ( 1 ) ", disco2 = " ( 2 ) ", disco3 = "
( 3 ) ", disco4 = "( 4 )", vacio = " | ";
    static String[] columna1 = {disco1, disco2, disco3, disco4};
    static final String[] discos = {disco1, disco2, disco3, disco4};
    static String[] columna2 = {vacio, vacio, vacio, vacio};
    static String[] columna3 = {vacio, vacio, vacio, vacio};
    static Scanner s = new Scanner(System.in);

    public static void main(String args[]) {
        //torresDeHanoi(4, 1, 2, 3, 0);
        System.out.println("-----TORRES DE HANOI-----");
        dibujar(0);
        System.out.println("INGRESE EL DISCO, LA COLUMNA DE ORIGEN Y LA
COLUMNA DESTINO:");
        juego(s.nextInt(), s.nextInt(), s.nextInt(), 1);
    }
    public static void juego(int d, int o, int dest, int i) {
        if ((d == 0 && o == 0 && dest == 0) || validarColumna(0)) {
            System.out.println("FIN DEL JUEGO \n total de movimientos
realizados = " + i);
        } else {
            moverDiscos(d, o, dest);
            dibujar(0);
            System.out.println("INGRESE EL DISCO, LA COLUMNA DE ORIGEN Y LA
COLUMNA DESTINO:");
            juego(s.nextInt(), s.nextInt(), s.nextInt(), i + 1);
        }
    }
    public static void torresDeHanoi(int d, int inicio, int auxiliar, int
fin, int i) {
        if (d != 0) {
            torresDeHanoi(d - 1, inicio, fin, auxiliar, i + 1);
            System.out.println("Mover Disco #" + d + " de " + inicio + " a "
+ fin + " I = " + i);
            moverDiscos(d, inicio, fin);
            dibujar(0);
            torresDeHanoi(d - 1, auxiliar, inicio, fin, i + 2);
        }
    }
    private static void dibujar(int i) {
        if (i == 3) {
            System.out.println(" " + columna1[i] + " " +
columna2[i] + " " + columna3[i]);
        } else {
            System.out.println(" " + columna1[i] + " " +
columna2[i] + " " + columna3[i]);
            dibujar(i + 1);
        }
    }
}

```

```
private static void moverDiscos(int d, int inicio, int fin) {
    String[] start = hallarColumna(inicio), end = hallarColumna(fin);
    String disco = hallarDisco(d - 1, 0);
    if (start != null && end != null && disco != null && inicio != fin) {
        int pos = posicionDisco(disco, start, 3);
        if (pos > -1) {
            if (validarPos(pos, start)) {
                int posLibre = posicionLibre(end, 3);
                if (validarPosDestino(end, posLibre, d)) {
                    start[pos] = vacio;
                    end[posLibre] = disco;
                    ActualizarColumna(inicio, start);
                    ActualizarColumna(fin, end);
                } else {
                    System.out.println("NO SE PUEDE UBICAR EL DISCO " +
disco + " EN LA COLUMNA " + fin + " POS LIBRE = " + posLibre);
                }
            } else {
                System.out.println("NO SE PUEDE MOVER EL DISCO " + disco
+ " DE SU POSICION");
            }
        } else {
            System.out.println("NO SE ENCONTRO EL DISCO " + disco + " EN
LA COLUMNA " + inicio);
        }
    } else {
        System.out.println("PROBLEMA CON LOS DATOS INGRESADOS :\n"
+ "- Disco seleccionado debe ser 1,2,3,4\n"
+ "- Columnas origen debe ser diferente a la columna
destino\n"
+ "- Columna origen y destino deben ser 1,2,3");
    }
}

public static String[] hallarColumna(int i) {
    switch (i) {
        case 1:
            return columna1;
        case 2:
            return columna2;
        case 3:
            return columna3;
    }
    return null;
}
```

```
public static void ActualizarColumna(int i, String[] c) {
    switch (i) {
        case 1:
            columna1 = c;
            break;
        case 2:
            columna2 = c;
            break;
        case 3:
            columna3 = c;
            break;
    }
}

private static int posicionLibre(String[] end, int i) {
    if (i < 0) {
        return -1;
    } else if (" | ".equals(end[i])) {
        return i;
    } else {
        return posicionLibre(end, i - 1);
    }
}

private static String hallarDisco(int d, int i) {
    if (i < 0 || i > 3) {
        return null;
    } else if (d == 3) {
        return discos[3];
    } else if (d == i) {
        return discos[i];
    } else {
        return hallarDisco(d, i + 1);
    }
}

private static int posicionDisco(String disco, String[] c, int i) {
    if (i < 0) {
        return -1;
    } else if (c[i].equals(disco)) {
        return i;
    } else {
        return posicionDisco(disco, c, i - 1);
    }
}
```



```
private static int pesoDisco(String disco, int i) {
    if (disco.equals(disco1)) {
        return 1;
    } else if (disco.equals(disco2)) {
        return 2;
    } else if (disco.equals(disco3)) {
        return 3;
    } else {
        return 4;
    }
}

private static boolean validarPos(int pos, String[] columna) {
    if (pos != 0) {
        return columna[pos - 1].equals(vacio);
    } else {
        return true;
    }
}

private static boolean validarPosDestino(String[] end, int posLibre, int
disco) {
    switch (posLibre) {
        case -1:
            return false;
        case 3:
            return true;
        default:
            return pesoDisco(end[posLibre + 1], 3) > disco;
    }
}

private static boolean validarColumna(int i) {
    if (i == 3) {
        return discos[3].equals(columna3[3]);
    } else if (discos[i].equals(columna3[i])) {
        return validarColumna(i + 1);
    } else {
        return false;
    }
}
}
```