



	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

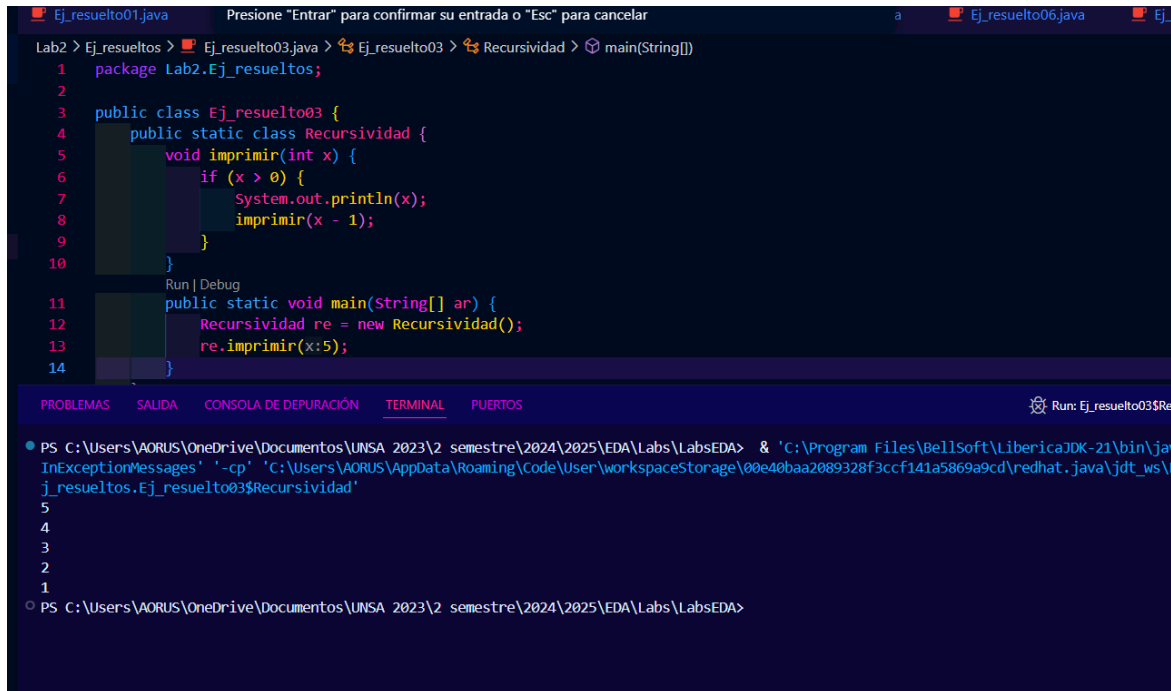
INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE DATOS Y ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	TÉCNICAS Y DISEÑO DE ALGORITMOS				
NÚMERO DE PRÁCTICA:	02	AÑO LECTIVO:	2025 – A	NRO. SEMESTRE:	Tercero III
FECHA DE PRESENTACIÓN	17/05/2025	HORA DE PRESENTACIÓN	11:59		
INTEGRANTE (s): Davila Flores Mathias Dario				NOTA:	
DOCENTE(s): <ul style="list-style-type: none"> <li>Mg. Ing. Rene Alonso Nieto Valencia.</li> <li>ENLACE GITHUB: <a href="https://github.com/mathiasddf/LabsEDA">https://github.com/mathiasddf/LabsEDA</a></li> </ul>					

SOLUCIÓN Y RESULTADOS
<p><b>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</b></p> <p><b>a. Ejercicios Resueltos:</b></p> <p><b>i. Implementación de un método recursivo:</b></p> <p>Cuando una función recursiva se llama a sí misma, debe tener un caso base que detenga las llamadas recursivas, para evitar que siga llamándose indefinidamente.</p> <p>Si el caso base no está bien definido, o nunca se cumple, la recursión nunca termina y el programa lanza un error de StackOverflowError.</p>

```
Lab2 > Ej_resultos > Ej_resuelto02.java > ...  
1 package Lab2.Ej_resultos;  
  
2  
3 public class Ej_resuelto02 {  
4     public static class Recursividad {  
5         void imprimir(int x) {  
6             System.out.println(x);  
7             imprimir(x - 1);  
8         }  
9     }  
10    public static void main(String[] ar) {  
11        Recursividad re = new Recursividad();  
12        re.imprimir(5);  
13    }  
14 }
```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 3</p>

**iii. Implementar un método recursivo que imprima en forma descendente de 5 a 1 de uno en uno.**

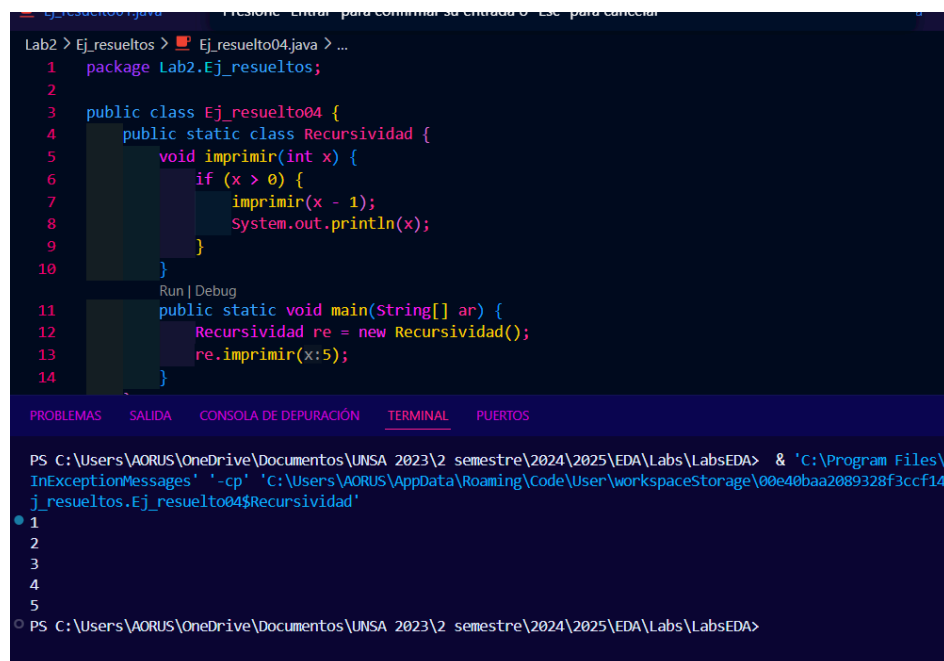


```

Ej_resuelto01.java  Presione "Entrar" para confirmar su entrada o "Esc" para cancelar
Lab2 > Ej_resueltos > Ej_resuelto03.java > Ej_resuelto03 > Recursividad > main(String[])
1 package Lab2.Ej_resueltos;
2
3 public class Ej_resuelto03 {
4     public static class Recursividad {
5         void imprimir(int x) {
6             if (x > 0) {
7                 System.out.println(x);
8                 imprimir(x - 1);
9             }
10        }
11    }
12    public static void main(String[] ar) {
13        Recursividad re = new Recursividad();
14        re.imprimir(x:5);
15    }
16 }
Run | Debug
PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK-21\bin\java.exe' -cp 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\redhat.java\jdt_ws\j_resueltos.Ej_resuelto03$Recursividad'
5
4
3
2
1
PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

**iv. Imprimir los números de 1 a 5 en pantalla utilizando recursividad.**

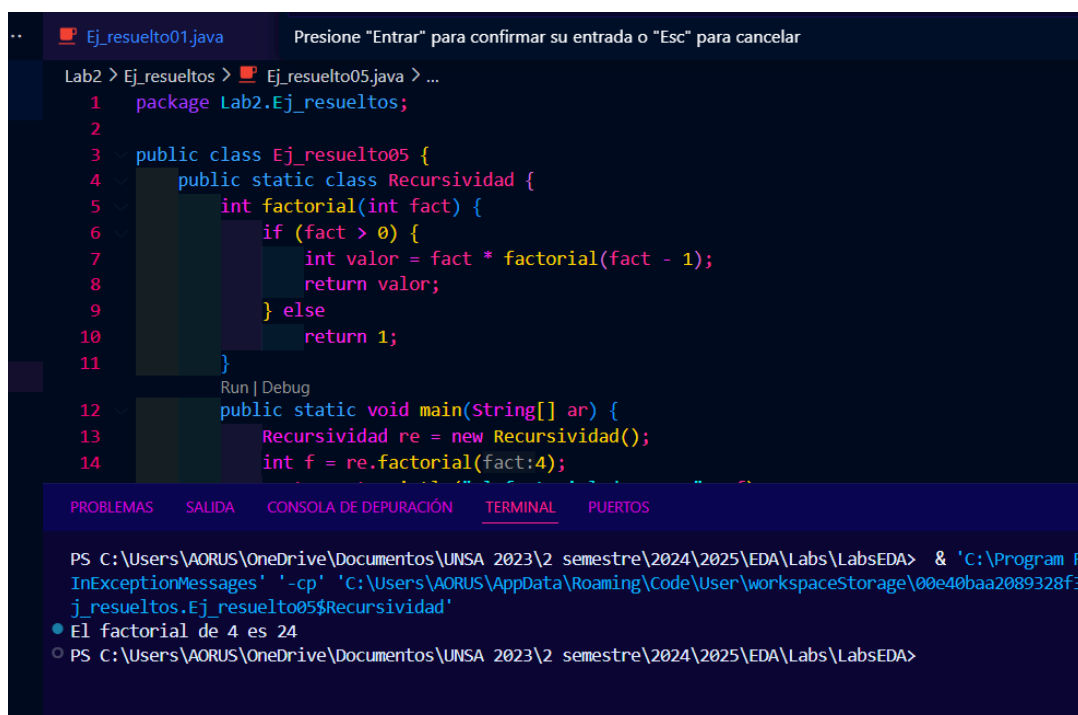


```

Ej_resuelto01.java  Presione "Entrar" para confirmar su entrada o "Esc" para cancelar
Lab2 > Ej_resueltos > Ej_resuelto04.java > ...
1 package Lab2.Ej_resueltos;
2
3 public class Ej_resuelto04 {
4     public static class Recursividad {
5         void imprimir(int x) {
6             if (x > 0) {
7                 imprimir(x - 1);
8                 System.out.println(x);
9             }
10        }
11    }
12    public static void main(String[] ar) {
13        Recursividad re = new Recursividad();
14        re.imprimir(x:5);
15    }
16 }
Run | Debug
PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK-21\bin\java.exe' -cp 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\redhat.java\jdt_ws\j_resueltos.Ej_resuelto04$Recursividad'
1
2
3
4
5
PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

- v. Obtener el factorial de un número. Recordar que el factorial de un número es el resultado que se obtiene de multiplicar dicho número por el anterior y así sucesivamente hasta llegar a uno. Ej. el factorial de 4 es  $4 * 3 * 2 * 1$  es decir 24.



```
.. Ej_resuelto01.java Presione "Entrar" para confirmar su entrada o "Esc" para cancelar
Lab2 > Ej_resueltos > Ej_resuelto05.java > ...
1 package Lab2.Ej_resueltos;
2
3 public class Ej_resuelto05 {
4     public static class Recursividad {
5         int factorial(int fact) {
6             if (fact > 0) {
7                 int valor = fact * factorial(fact - 1);
8                 return valor;
9             } else
10                return 1;
11        }
12        Run | Debug
13        public static void main(String[] ar) {
14            Recursividad re = new Recursividad();
15            int f = re.factorial(fact:4);
16        }
17    }
18
19 PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\AORUS\OneDrive\Documentos\UNSA_2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\Java\jdk-17\bin\java.exe' -cp 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f30\workspace\Lab2\Ej_resueltos\Ej_resuelto05$Recursividad'
● El factorial de 4 es 24
○ PS C:\Users\AORUS\OneDrive\Documentos\UNSA_2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>
```

**vi. Implementar un método recursivo para ordenar los elementos de un vector.**

```
Lab2 > Ej_resueltos > Ej_resuelto06.java > Ej_resuelto06 > Recursividad > main(String[])
1 package Lab2.Ej_resueltos;
2
3 public class Ej_resuelto06 {
4     public static class Recursividad {
5         static int[] vec = { 312, 614, 88, 22, 54 };
6         void ordenar(int[] v, int cant) {
7             if (cant > 1) {
8                 for (int f = 0; f < cant - 1; f++)
9                     if (v[f] > v[f + 1]) {
10                         int aux = v[f];
11                         v[f] = v[f + 1];
12                         v[f + 1] = aux;
13                     }
14                 ordenar(v, cant - 1);
15             }
16         }
17         void imprimir() {
18             for (int f = 0; f < vec.length; f++)
19                 System.out.print(vec[f] + " ");
20             System.out.println(x:"\n");
21         }
22     }
23     public static void main(String[] ar) {
24         Recursividad r = new Recursividad();
25
26         r.imprimir();
27     }
28 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** PUERTOS Run: E

```
● PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK
InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\redhat.ja
j_resueltos.Ej_resuelto06$Recursividad'
312 614 88 22 54

22 54 88 312 614

○ PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>
```

## b. Ejercicios Propuestos:

### i. Invertir vector de enteros, permite ingresar tamaño y captura de valores del arreglo, el método invertirArray calcula y muestra el resultado:

El programa invierte un arreglo intercambiando sus elementos desde los extremos hacia el centro, usando una variable temporal para hacer los cambios sin perder datos. Así, logra invertir el orden sin crear un arreglo nuevo y con un recorrido eficiente hasta la mitad del arreglo. Luego muestra el arreglo invertido.

```

6  Ej_propuestos.java -> Asistencia -> Ej_propuestos01 -> main(String[])
7
8  public class Ej_propuesto01 {
9
10     public static int[] invertirArray(int[] A) {
11         int n = A.length;
12         for (int i = 0; i < n / 2; i++) {
13             int temp = A[i];
14             A[i] = A[n - 1 - i];
15             A[n - 1 - i] = temp;
16         }
17
18         // Mostrar resultado
19         System.out.print(s:"Arreglo invertido: ");
20         for (int val : A) {
21             System.out.print(val + " ");
22         }
23         System.out.println();
24
25         return A;
26     }
27
28     PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
29
30     InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\redhat.java
31     j_propuestos.Ej_propuesto01'
32     Ingrese tamaño del arreglo: 5
33     Ingrese los valores del arreglo:
34     6
35     8
36     2
37     3
38     5
39     Arreglo: 6 8 2 3 5
40     Arreglo invertido: 5 3 2 8 6
41     PS C:\Users\AORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

### ii. Rotar a la Izquierda, permite ingresar tamaño y captura de valores del arreglo, el método rotarIzquierdaArray calcula y muestra el resultado:

El programa rota un arreglo hacia la izquierda d posiciones creando un nuevo arreglo. Copia primero los elementos desde d hasta el final al inicio del nuevo arreglo, y luego los primeros d elementos al final, logrando así el desplazamiento circular de los valores. Finalmente muestra y devuelve el arreglo rotado.

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 7

```

8
9 public class Ej_propuesto02 {
10     public static int[] rotarIzquierdaArray(int[] A, int d) {
11         int n = A.length;
12         d = d % n; // normalizar d
13         int[] Ainvertido = new int[n];
14
15         for (int i = 0; i < n - d; i++) {
16             Ainvertido[i] = A[i + d];
17         }
18         for (int i = n - d; i < n; i++) {
19             Ainvertido[i] = A[i - (n - d)];
20         }
21         // Mostrar resultado
22         System.out.print(s:"Arreglo rotado a la izquierda: ");
23         for (int val : Ainvertido) {
24             System.out.print(val + " ");
25         }
26         System.out.println();
27         return Ainvertido;
28     }
29     public static void main(String[] args) {
30
31         Run | Debug
32
33         PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
34
35         PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\Lib
36         InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\r
37         j_propuestos.Ej_propuesto02'
38         Ingrese tamaño del arreglo: 5
39         Ingrese los valores del arreglo:
40         3 9 6 8 4
41         Ingrese número de posiciones a rotar (d): 4
42         Arreglo rotado a la izquierda: 4 3 9 6 8
43         PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

iii. Triángulo recursivo 1. El método `trianguloRecursivo1` calcula y muestra el resultado.

- Si  $b = 5$
- Salida:

```

*
**
***
****
*****

```

El programa usa recursión para imprimir un triángulo de asteriscos con tantas líneas como indica el parámetro base. Cada llamada imprime una línea con un número de asteriscos igual al nivel actual, construyendo el triángulo de arriba hacia abajo.

```

Lab2 > Ej_propuestos > Ej_propuesto03.java > Ej_propuesto03 > main(String[])
6      *
7      **
8      ***
9      ****
10     *****
11     */
12
13     public class Ej_propuesto03 {
14         public static void trianguloRecursivo1(int base) {
15             if (base > 0) {
16                 trianguloRecursivo1(base - 1); // llamada recursiva
17                 for (int i = 0; i < base; i++) {
18                     System.out.print(s:"");
19                 }
20                 System.out.println();
21             }
22         }
23
24         public static void main(String[] args) {
25             Scanner sc = new Scanner(System.in);
26             System.out.println(x:"Indicar numero de pisos del grafico (base): ");
27             int base = sc.nextInt();
28             trianguloRecursivo1(base);
29             sc.close();
30
31         }
32     }
33
34     Run | Debug
35
36     public static void main(String[] args) {
37         Scanner sc = new Scanner(System.in);
38         System.out.println(x:"Indicar numero de pisos del grafico (base): ");
39         int base = sc.nextInt();
40         trianguloRecursivo1(base);
41         sc.close();
42     }
43
44     PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
45
46     PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\Java\jdk-11.0.10\bin\java.exe' -cp 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f\j_propuestos\Ej_propuesto03'
47     Indicar numero de pisos del grafico (base):
48     4
49     *
50     **
51     ***
52     ****
53     *****
54
55     PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

iv. **Triángulo recursivo 2.** El método `trianguloRecursivo2` calcula y muestra el resultado.

- Si  $b = 5$
- Salida:
 

```

                *
               **
              ***
             ****
            *****
            
```

El programa imprime un triángulo de asteriscos usando recursión. En cada llamada, imprime espacios y asteriscos según el nivel actual, y se llama a sí misma aumentando ese nivel hasta completar la base indicada por el usuario. Así se forma un triángulo ascendente alineado a la derecha.



```

12
13 public class Ej_propuesto04 {
14     public static void trianguloRecursivo2(int base, int actual) {
15         if (actual > base) return;
16         // Imprimir espacios
17         for (int i = 0; i < base - actual; i++) {
18             System.out.print(s:" ");
19         }
20         // Imprimir asteriscos
21         for (int i = 0; i < actual; i++) {
22             System.out.print(s:"*");
23         }
24         System.out.println();
25         trianguloRecursivo2(base, actual + 1);
26     }
27     public static void main(String[] args) {
28         Scanner sc = new Scanner(System.in);
29         System.out.println(x:"Indicar numero de pisos del grafico (base): ");
30         int base = sc.nextInt();
31         trianguloRecursivo2(base, actual:1);
32     }
33 }

```

Run | Debug

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\Bej InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5 j\_propuestos.Ej\_propuesto04'

Indicar numero de pisos del grafico (base):

3

```

*
**
***
****
*****

```

PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

v. Triángulo recursivo 3. El método `trianguloRecursivo3` calcula y muestra el resultado.}

- Si  $b = 5$
- Salida:
 

```

*
**
***
****
*****

```

El programa imprime recursivamente un triángulo de asteriscos centrado, donde en cada línea imprime espacios para alinear y luego los asteriscos con espacios entre ellos. La recursión termina cuando se alcanzan todas las líneas indicadas, construyendo así una pirámide desde la primera hasta la última línea.

```

10  * * * * *
11  */
12
13  public class Ej_propuesto05 {
14      public static void trianguloRecursivo3(int totalLineas, int lineaActual) {
15          if (lineaActual > totalLineas) {
16              return; // caso base: ya imprimimos todas las líneas
17          }
18          // imprimir espacios para centrar
19          for (int i = 0; i < totalLineas - lineaActual; i++) {
20              System.out.print(s: " ");
21          }
22          // imprimir asteriscos con espacios entre ellos
23          for (int i = 0; i < lineaActual; i++) {
24              System.out.print(s: "*");
25              if (i < lineaActual - 1) {
26                  System.out.print(s: " ");
27              }
28          }
29          System.out.println();
30          trianguloRecursivo3(totalLineas, lineaActual + 1);
31      }
32  }

```

Run | Debug

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\Be  
InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a  
j\_propuestos.Ej\_propuesto05'  
Ingrese la cantidad de pisos para la pirámide: 6  
\*  
\* \*  
\* \* \*  
\* \* \* \*  
\* \* \* \* \*  
\* \* \* \* \*  
\* \* \* \* \*

PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

vi. Cuadrado recursivo. El método `cuadradoRecursivo` calcula y muestra el resultado.

- Si  $b = 5$       \*\*\*\*\*
- Salida:        \*       \*
- \*       \*
- \*       \*
- \*\*\*\*\*

El programa imprime un cuadrado hueco de asteriscos de tamaño `base` usando recursión. En cada llamada, dibuja la línea actual: llena de asteriscos si es la primera o última, o con asteriscos en los bordes y espacios en medio si es intermedia. La recursión avanza hasta imprimir todas las líneas del cuadrado.

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 11</p>

```

13
14 public class Ej_propuesto06 {
15     public static void cuadradoRecurso(int base, int lineaActual) {
16         if (lineaActual > base) {
17             return; // caso base: terminamos
18         }
19         if (lineaActual == 1 || lineaActual == base) {
20             // línea llena de asteriscos
21             for (int i = 0; i < base; i++) {
22                 System.out.print(s:"*");
23             }
24         } else {
25             // línea intermedia: asterisco + espacios + asterisco
26             System.out.print(s:"*");
27             for (int i = 0; i < base - 2; i++) {
28                 System.out.print(s:" ");
29             }
30             System.out.print(s:"*");
31         }
32         System.out.println();
33         cuadradoRecurso(base, lineaActual + 1);
34     }
35     public static void main(String[] args) {

```

Run | Debug

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\Bel  
InExceptionMessages' -cp 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5  
j\_propuestos.Ej\_propuesto06'

● Ingrese el tamaño de la base del cuadrado: 5

```

*****
*   *
*   *
*   *
*****

```

○ PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> |

## II. SOLUCIÓN DEL CUESTIONARIO

- a. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.
- La ausencia de funciones o métodos adicionales hizo que el código principal se volviera muy extenso y poco organizado, dificultando la localización de errores.
  - No se implementaron validaciones para entradas de usuario, lo que provocaba que el programa fallara o se comportara de forma inesperada ante datos inválidos.
  - La lógica concentrada únicamente en el método principal limitó la reutilización del código y complicó la escalabilidad de los programas.

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 12</p>

#### **b. Diferencias entre algoritmos de secuencialidad, decisión e iteración.**

- La secuencialidad siempre sigue un camino fijo, mientras que la decisión y la iteración introducen variabilidad en el flujo del programa.
- La decisión evalúa condiciones para tomar diferentes rutas, pero cada ruta se ejecuta solo una vez por evaluación.
- La iteración puede ejecutar repetidamente un mismo bloque, lo que permite procesar datos de forma repetitiva o hasta cumplir un criterio.
- La secuencialidad no tiene condiciones ni ciclos; la decisión tiene condiciones sin repetición; la iteración combina condiciones con repetición.
- En términos de complejidad, la secuencialidad es la más simple, la decisión añade lógica condicional y la iteración añade repetición, aumentando la complejidad del algoritmo.
- La iteración puede generar bucles infinitos si no se maneja correctamente la condición, mientras que la decisión no tiene este riesgo.
- La decisión y la iteración permiten implementar lógica más dinámica y flexible que la secuencialidad.

#### **c. Qué son las clases y métodos genéricos**

Las clases genéricas son plantillas que permiten definir una clase con uno o más parámetros de tipo, los cuales se especifican al crear una instancia de la clase. Esto permite que una misma clase pueda trabajar con diferentes tipos de datos sin necesidad de duplicar el código para cada tipo específico, manteniendo la seguridad de tipos en tiempo de compilación.

Los métodos genéricos son aquellos que tienen uno o más parámetros de tipo propios, permitiendo que el método opere con distintos tipos de datos de manera flexible y segura. Estos métodos pueden estar dentro de clases genéricas o no, y facilitan la reutilización del código al evitar la necesidad de escribir múltiples versiones del mismo método para diferentes tipos.

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 13

### III. CONCLUSIONES

- Los ejercicios permitieron practicar y fortalecer el uso de estructuras básicas de programación como secuencialidad, decisiones e iteraciones, fundamentales para resolver problemas simples.
- La falta de validación de entradas en la mayoría de los ejercicios mostró la necesidad de implementar controles para evitar errores o comportamientos inesperados durante la ejecución.
- El desarrollo centrado en métodos principales sin dividir la lógica en funciones específicas limitó la reutilización y escalabilidad de los programas.

### RETROALIMENTACIÓN GENERAL

- Buen manejo de estructuras básicas (secuencialidad, decisión, iteración) para resolver problemas simples.
- Falta de comentarios y documentación interna que dificulta la comprensión rápida del código.
- Ausencia de validación de entradas y manejo de errores, lo que puede causar fallos inesperados.
- Recomendación: mejorar documentación, dividir la lógica en métodos, validar entradas.

### REFERENCIAS Y BIBLIOGRAFÍA

- Deitel, P., & Deitel, H. (2017). *Java: How to Program* (11th Edition). Pearson.
- Bloch, J. (2018). *Effective Java* (3rd Edition). Addison-Wesley.
- Weiss M., *Data Structures and Algorithms Analysis in Java*, 2012, Addison-Wesley
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd Edition). MIT Press. Capítulos sobre estructuras de control y algoritmos.