

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

Ejercicio 2: Implementación de un método recursivo que reciba un parámetro de tipo entero y luego llame en forma recursiva con el valor del parámetro menos 1.

```

1 public class Recursividad2 {
2     void imprimir(int x) {
3         System.out.println(x);
4         imprimir(x - 1);
5     }
6
7     Run | Debug | Run main | Debug main
8     public static void main(String[] ar) {
9         Recursividad2 re = new Recursividad2();
10        re.imprimir(x:5);
11    }

```

Resultado:

```

-8718
-8719
-8720
-8721
-8722
-8723
-8724
-8725
-8726
-8727
-8728

```

Ejercicio 3: Implementar un método recursivo que imprima en forma descendente de 5 a 1 de uno en uno.

```

1 public class Recursividad3 {
2     void imprimir(int x) {
3         if (x > 0) {
4             System.out.println(x);
5             imprimir(x - 1);
6         }
7     }
8
9     Run | Debug | Run main | Debug main
10    public static void main(String[] ar) {
11        Recursividad3 re = new Recursividad3();
12        re.imprimir(x:5);
13    }

```

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

```
PS D:\UNSA\5º Semestre\EDA\Labotarioo> & 'C:\Program Files\
paceStorage\0dbceb9812766135e67feaea89c2c864\redhat.java\
5
4
3
2
1
```

Ejercicio 4: Imprimir los números de 1 a 5 en pantalla utilizando recursividad.

```
1 public class Recursividad4 {
2     void imprimir(int x) {
3         if (x > 0) {
4             imprimir(x - 1);
5             System.out.println(x);
6         }
7     }
8
9     Run | Debug | Run main | Debug main
10    public static void main(String[] ar) {
11        Recursividad4 re = new Recursividad4();
12        re.imprimir(x:5);
13    }
```

Resultado:

```
PS D:\UNSA\5º Semestre\EDA\Labotarioo> & 'C:\Program Files\
paceStorage\0dbceb9812766135e67feaea89c2c8
1
2
3
4
5
PS D:\UNSA\5º Semestre\EDA\Labotarioo>
```

Ejercicio 5: 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.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

```

1 public class Recursividad5 {
2     int factorial(int fact) {
3         if (fact > 0) {
4             int valor = fact * factorial(fact - 1);
5             return valor;
6         } else
7             return 1;
8     }
9 }

Run | Debug | Run main | Debug main
10 public static void main(String[] ar) {
11     Recursividad5 re = new Recursividad5();
12     int f = re.factorial(fact:4);
13     System.out.println("El factorial de 4 es " + f);
14 }
15 }

```

Resultado:

```

PS D:\UNSA\5° Semestre\EDA\Labotarioo> &
paceStorage\0dbceb9812766135e67feaea89c2c8
El factorial de 4 es 24
PS D:\UNSA\5° Semestre\EDA\Labotarioo>

```

Ejercicio 6. Implementar un método recursivo para ordenar los elementos de un vector.

```

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

```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 5</p>

Resultado:


```
PS D:\UNSA\5º Semestre\EDA\Labotarioo> d.; cd 'C:\Users\Usuario\AppData\Roaming\Code\User\workspace'
d'
312 614 88 22 54
22 54 88 312 614
```

Problemas Propuestos:

1-. Invertir un vector de enteros. Se debe permitir ingresar el tamaño del vector y capturar sus valores. El método `invertirArray` recibe el arreglo original, invierte el orden de sus elementos y devuelve el arreglo invertido.

```
1  import java.util.Scanner;
2  public class Propuesto1 {
3      // Método para invertir el arreglo
4      public static int[] invertirArray(int[] A) {
5          int n = A.length;
6          int[] Asalida = new int[n];
7          for (int i = 0; i < n; i++) {
8              Asalida[i] = A[n - 1 - i];
9          }
10         return Asalida;
11     }
12     Run | Debug | Run main | Debug main
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         System.out.print(s:"Ingrese tamaño del vector: ");
16         int N = sc.nextInt();
17         int[] A = new int[N];
18         System.out.println(x:"Ingrese los valores del arreglo:");
19         for (int i = 0; i < N; i++) {
20             A[i] = sc.nextInt();
21         }
22         int[] resultado = invertirArray(A);
23         System.out.println(x:"Arreglo invertido:");
24         for (int val : resultado) {
25             System.out.print(val + " ");
26         }
27         sc.close();
28     }
}
```

Resultados:

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 6</p>

```
PS D:\UNSA\5° Semestre\EDA\Labotarioo> & 'C:\P
b9812766135e67feaea89c2c864\redhat.java\jdt_ws\
Ingrese tamaño del vector: 5
Ingrese los valores del arreglo:
1
2
3
4
5
Arreglo invertido:
5 4 3 2 1
```

Explicación:


- El método invertirArray recibe un arreglo A y crea un nuevo arreglo Asalida de igual tamaño.
- Recorre el arreglo original desde el inicio, asignando a Asalida los valores desde el final hacia el inicio (Asalida[i] = A[n-1-i]).
- El main se encarga de capturar el tamaño y los valores del arreglo desde teclado, llama al método para invertirlo y luego imprime el arreglo invertido.
- Se utiliza un ciclo for para recorrer y mostrar el arreglo resultante.

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

Si d=2

A=[1 2 3 4 5] -> Ainvertido=[3 4 5 1 2]

```
1 public class Propuesto2 {
2     public static int[] rotarIzquierdaArray(int[] A, int d) {
3         int n = A.length;
4         int[] Ainvertido = new int[n];
5         d = d % n; // Por si d > n
6         for (int i = 0; i < n; i++) {
7             int nuevaPos = (i + n - d) % n;
8             Ainvertido[nuevaPos] = A[i];
9         }
10        return Ainvertido;
11    }
12    Run | Debug | Run main | Debug main
13    public static void main(String[] args) {
14        java.util.Scanner sc = new java.util.Scanner(System.in);
15        System.out.print(s:"Ingrese tamaño del arreglo: ");
16        int N = sc.nextInt();
17        int[] A = new int[N];
18        System.out.println(x:"Ingrese los valores del arreglo:");
19        for (int i = 0; i < N; i++) {
20            A[i] = sc.nextInt();
21        }
22        System.out.print(s:"Ingrese número de posiciones a rotar a la izquierda (d): ");
23        int d = sc.nextInt();
24        int[] resultado = rotarIzquierdaArray(A, d);
25        System.out.println(x:"Arreglo rotado a la izquierda:");
26        for (int val : resultado) {
27            System.out.print(val + " ");
28        }
29        sc.close();
30    }
31 }
```

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

Resultado:

```
PS D:\UNSA\5º Semestre\EDA\Labotarioo> d:; cd 'd:\UNSA\5º S
Users\Usuario\AppData\Roaming\Code\User\workspaceStorage\0db
'
Ingrese tamaño del arreglo: 5
Ingrese los valores del arreglo:
1
2
3
4
5
Ingrese número de posiciones a rotar a la izquierda (d): 2
Arreglo rotado a la izquierda:
3 4 5 1 2
```

Explicación:

- $d = d \% n$ asegura que la rotación no supere el tamaño del arreglo.
- Para cada índice i del arreglo original, calculo su nueva posición después de rotar a la izquierda d veces con $(i + n - d) \% n$.
- Creo un arreglo nuevo $A_{invertido}$ para almacenar los valores rotados.
- Finalmente, retorno el arreglo rotado.
- En main se capturan los datos y se muestra el resultado.



3-. Triangulo recursivo 1. El método trianguloRecursivo1 calcula y muestra el resultado.

- Si $b = 5$

- Salida:

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

```
1 public class Propuesto3 {
2     public void trianguloRecursivo1(int base) {
3         imprimirFilas(filaActual:1, base);
4     }
5     // Método auxiliar recursivo para imprimir filas
6     private void imprimirFilas(int filaActual, int base) {
7         if (filaActual > base) {
8             return; // Caso base: terminamos
9         }
10        imprimirAsteriscos(filaActual);
11        System.out.println();
12        imprimirFilas(filaActual + 1, base);
13    }
14    // Método auxiliar para imprimir asteriscos de forma iterativa en una fila
15    private void imprimirAsteriscos(int cantidad) {
16        for (int i = 0; i < cantidad; i++) {
```

	<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: 8

```

17      System.out.print(s:"*");
18  }
19  }
    Run | Debug | Run main | Debug main
20  public static void main(String[] args) {
21      Propuesto3 tr = new Propuesto3();
22      int b = 5; // Puedes cambiar este valor
23      tr.trianguloRecursivo1(b);
24  }
25  }

```

Resultado:

```

PS D:\UNSA\5º Semestre\EDA\Labotarioo> & 'C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Python\Python39\python.exe' C:\Users\user\AppData\Local\Temp\0dbceb9812766135e67feaea89c2c864\*.py
*
**
***
****
*****

```

Explicación:

- trianguloRecursivo1 inicia la impresión llamando a imprimirFilas desde la fila 1 hasta la base.
- imprimirFilas es recursivo: imprime los asteriscos correspondientes a la fila actual y luego llama a sí mismo con la siguiente fila.
- Cuando filaActual es mayor que base, termina la recursión.
- La función imprimirAsteriscos imprime el número de asteriscos requeridos para la fila actual con un simple ciclo for.

4.- Triangulo recursivo 2. El método trianguloRecursivo2 calcula y muestra el resultado.

- Si b = 5

- Salida:

```

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

```

```

1  public class Propuesto4 {
2      public void trianguloRecursivo2(int base) {
3          imprimirFilas(filaActual:1, base);
4      }
5      private void imprimirFilas(int filaActual, int base) {
6          if (filaActual > base) {
7              return;
8          }
9          imprimirEspacios(base - filaActual);

```



```
10     imprimirAsteriscos(filaActual);
11     System.out.println();
12     imprimirFilas(filaActual + 1, base);
13 }
14 private void imprimirEspacios(int cantidad) {
15     for (int i = 0; i < cantidad; i++) {
16         System.out.print(s: " ");
17     }
18 }
19 private void imprimirAsteriscos(int cantidad) {
20     for (int i = 0; i < cantidad; i++) {
21         System.out.print(s: "*");
22     }
23 }
24 Run | Debug | Run main | Debug main
25 public static void main(String[] args) {
26     Propuesto4 tr = new Propuesto4();
27     int b = 5;
28     tr.trianguloRecursivo2(b);
29 }
```

Resultado:



```
PS D:\UNSA\5º Semestre\EDA\Labotarioo> & '
paceStorage\0dbceb9812766135e67feaea89c2c86
*
**
***
****
*****
PS D:\UNSA\5º Semestre\EDA\Labotarioo>
```

Explicación:

- imprimirFilas controla la fila actual y recursivamente imprime hasta la base.
- Para alinear a la derecha, primero imprime base - filaActual espacios.
- Luego imprime filaActual asteriscos.
- Se usa recursión para avanzar en las filas.
- Cuando la fila actual supera la base, termina la recursión.

5-. Triangulo recursivo 3. El método trianguloRecursivo3 calcula y muestra el resultado.

- Si b = 5
- Salida:

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

*


```

1 public class Propuesto5 {
2     public void trianguloRecursivo3(int base) {
3         imprimirFila(filaActual:1, base);
4     }
5     private void imprimirFila(int filaActual, int base) {
6         if (filaActual > base)
7             return;
8         // Imprime espacios (base - filaActual)
9         imprimirEspacios(base - filaActual);
10        // Imprime asteriscos (2 * filaActual - 1)
11        imprimirAsteriscos(2 * filaActual - 1);
12        System.out.println();
13        imprimirFila(filaActual + 1, base);
14    }
15    private void imprimirEspacios(int cantidad) {
16        if (cantidad == 0)
17            return;
18        System.out.print(s:" ");
19        imprimirEspacios(cantidad - 1);
20    }
21    private void imprimirAsteriscos(int cantidad) {
22        if (cantidad == 0)
23            return;
24        System.out.print(s:"*");
25        imprimirAsteriscos(cantidad - 1);
26    }
27    Run | Debug | Run main | Debug main
28    public static void main(String[] args) {
29        Propuesto5 tr = new Propuesto5();
30        int b = 5;
31        tr.trianguloRecursivo3(b);
32    }

```

Resultados:

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

```
PS D:\UNSA\5º Semestre\EDA\Labotarioo> & 'C:\Proq
paceStorage\0dbceb9812766135e67feaea89c2c864\redh
*
***
*****
*****
*****
*****
*****
*****
PS D:\UNSA\5º Semestre\EDA\Labotarioo>
```

Explicacion:

- Para centrar el árbol, imprimimos base - filaActual espacios.
- Luego imprimimos $2 * \text{filaActual} - 1$ asteriscos, formando una pirámide con cantidad impar de asteriscos.
- Se usa recursión para imprimir cada fila y también para imprimir espacios y asteriscos sin ciclos.
- Al aumentar la fila, se reduce la cantidad de espacios y se incrementan los asteriscos, generando la forma de árbol.

6.- Cuadrado recursivo. El método cuadradoRecursivo calcula y muestra el resultado.

- Si $b = 5$

• Salida:

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

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

```

1 public class Propuesto6 {
2     public void cuadradoRecursoivo(int base) {
3         imprimirFila(filaActual:1, base);
4     }
5     // Recursión para imprimir fila por fila
6     private void imprimirFila(int filaActual, int base) {
7         if (filaActual > base)
8             return;
9         imprimirColumna(filaActual, columna:1, base);
10        System.out.println();
11        imprimirFila(filaActual + 1, base);
12    }
13    // Recursión para imprimir columnas de una fila
14    private void imprimirColumna(int fila, int columna, int base) {
15        if (columna > base)
16            return;
17        // Condición para imprimir asterisco en bordes
18        if (fila == 1 || fila == base || columna == 1 || columna == base) {
19            System.out.print(s:"*");
20        } else {
21            System.out.print(s:" ");
22        }
23        imprimirColumna(fila, columna + 1, base);
24    }
25    Run | Debug | Run main | Debug main
26    public static void main(String[] args) {
27        Propuesto6 fr = new Propuesto6();
28        int b = 5;
29        fr.cuadradoRecursoivo(b);
30    }

```

Explicación:

- imprimirFila controla la impresión fila a fila de forma recursiva.
- imprimirColumna imprime cada carácter en la fila, con recursión sobre las columnas.
- Se imprimen asteriscos en las filas y columnas externas (fila == 1 || fila == base || columna == 1 || columna == base).
- El interior se llena con espacios.
- Esto genera el cuadrado con borde sólido y espacio vacío interno.

II. SOLUCIÓN DEL CUESTIONARIO

Dificultades: Complejidad de la recursión, control de índices, falta de documentación clara, manejo de entradas y validación.

- Diferencias:Secuencialidad: ejecución paso a paso.

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

- Decisión: ramificación según condición (if, switch).
- Iteración: repetición con ciclos (for, while).

Clases y métodos genéricos: Plantillas que permiten definir clases o métodos para distintos tipos sin perder seguridad de tipos, mejorando reutilización y flexibilidad.

III. CONCLUSIONES

La recursión es una herramienta poderosa pero requiere un buen control de la lógica y condiciones base para evitar errores o ciclos infinitos.

Comprender los tipos básicos de algoritmos (secuenciales, decisiones e iterativos) es fundamental para diseñar soluciones eficientes y claras.

Las clases y métodos genéricos mejoran la reutilización del código y permiten manejar datos de distintos tipos con seguridad, optimizando el desarrollo y mantenimiento del software.

La práctica constante y el análisis de problemas desde diferentes enfoques (iterativo vs recursivo) fortalecen las habilidades para resolver problemas complejos en ingeniería de sistemas.

RETROALIMENTACIÓN GENERAL

REFERENCIAS Y BIBLIOGRAFÍA

- Weiss M., *Data Structures & Problem Solving Using Java*, 2010, Addison-Wesley.
- Weiss M., *Data Structures and Algorithms Analysis in Java*, 2012, Addison-Wesley.
- Cormen T., Leiserson C., Rivest R., Stein C., *Introduction to Algorithms*, 2022, The MIT Press