



	<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: 1</p>

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE DATOS Y ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	POO, HERENCIA, INTERFACES Y GENERICIDAD				
NÚMERO DE PRÁCTICA:	03	AÑO LECTIVO:	2025 – A	NRO. SEMESTRE:	Tercero III
FECHA DE PRESENTACIÓN	24/05/2025	HORA DE PRESENTACIÓN	11:59		
INTEGRANTE (s): Davila Flores Mathias Dario				NOTA:	
DOCENTE(s): <ul style="list-style-type: none"> Mg. Ing. Rene Alonso Nieto Valencia. ENLACE GITHUB: https://github.com/mathiasddf/LabsEDA 					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>a. Ejercicios Resueltos:</p> <p>i. ArrayList: Se demuestra cómo declarar y utilizar listas dinámicas (ArrayList) para almacenar elementos, así como consultar su tamaño, si está vacía y su código hash.</p>

	<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: 2</p>

```

Lab3 > Ej_resueltos > Ej_resuelto01.java > ...
1 package Lab3.Ej_resueltos;
2 import java.util.ArrayList;
3
4 public class Ej_resuelto01 {
5     Run | Debug
6     public static void main(String [] args){
7         ArrayList<String> alumnos = new ArrayList<String>();
8         ArrayList<Integer> notas = new ArrayList<Integer>(); The value of the local variable notas is not used
9         alumnos.add(e:"MARIA");
10        alumnos.add(e:"DIEGO");
11        alumnos.add(e:"RENE");
12        alumnos.add(e:"ALONSO");
13        System.out.println(alumnos.hashCode());
14        System.out.println(alumnos.isEmpty());
15        System.out.println(alumnos.size());
16    }
17
PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS Run: E

PS C:\Users\AORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK-21\bin\jav
InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\redhat.java\jdt_ws
j_resueltos.Ej_resuelto01'
1121431919
false
4
PS C:\Users\AORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

- ii. **Iterador:** Se recorre la lista de alumnos usando un Iterator, lo que permite acceder secuencialmente a cada elemento de forma segura.

```

Ej_resuelto01.java Presione "Entrar" para confirmar su entrada o "Esc" para cancelar Ej_resuelto06.java Ej_
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    Run | Debug
13    public static void main(String[] ar) {
14        Recursividad re = new Recursividad();
15        re.imprimir(x:5);
16    }
17
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS Run: Ej_resuelto03$Re

PS C:\Users\AORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK-21\bin\jav
InExceptionMessages' '-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\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

	<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: 3</p>

- iii. **Clase Animal en Java:** Se desarrolló la clase Animal con los atributos nombre y género, junto a su constructor y métodos get y set, permitiendo encapsular la información de cada objeto. Además, se aplicó el uso de listas (ArrayList) para almacenar múltiples animales y se demostró cómo recorrerlos e imprimir sus datos, reforzando el manejo de clases y colecciones en Java.

```

4
5 class Animal {
6     String nombre;
7     boolean genero;
8
9     // Constructor
10    public Animal(String nombre, boolean genero) {
11        super();
12        this.nombre = nombre;
13        this.genero = genero;
14    }
15
16    public String getNombre() {
17        return nombre;
18    }
19
20    public void setNombre(String nombre) {
21        this.nombre = nombre;
22    }
23
24    public boolean isGenero() {
25        return genero;
26    }
27
28    public void setGenero(boolean genero) {
29        this.genero = genero;
30    }
31 }
32

```

```

33 public class Ej_resuelto03 {
34     Run | Debug
35     public static void main(String[] args) {
36         ArrayList<Animal> mascotas = new ArrayList<Animal>();
37         List<Animal> mascotas2 = new ArrayList<Animal>();
38         // List<Animal> mascotas3 = new List<Animal>(); // <-- Esto generaría error (no se puede instanciar una interfaz directamente)
39
40         mascotas.add(new Animal(nombre:"Firulais", genero:true));
41         mascotas2.add(new Animal(nombre:"Mishi", genero:false));
42
43         System.out.println(x:"Lista mascotas:");
44         for (Animal a : mascotas) {
45             System.out.println("Nombre: " + a.getNombre() + ", Género: " + (a.isGenero() ? "Macho" : "Hembra"));
46         }
47
48         System.out.println(x:"\nLista mascotas2:");
49         for (Animal a : mascotas2) {
50             System.out.println("Nombre: " + a.getNombre() + ", Género: " + (a.isGenero() ? "Macho" : "Hembra"));
51         }
52     }
53 }

```

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS Run: Ej_resuelto03 + - □ □ □ □

```

PS C:\Users\AORUS\OneDrive\Documentos\UNSA_2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK-21\bin\java.exe' '-XX:+ShowCode
InExceptionMessages' '-cp' 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5869a9cd\redhat.java\jdk_ws\LabsEDA_4af3ebda\bin'
j_resueltos.Ej_resuelto03'
● Lista mascotas:
Nombre: Firulais, Género: Macho

Lista mascotas2:
Nombre: Mishi, Género: Hembra
○ PS C:\Users\AORUS\OneDrive\Documentos\UNSA_2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

	<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>

b. Ejercicios Propuestos:

- i. Listas, Implementar una Lista usando POO con clases y métodos genéricos siguiendo los estándares de Java. (Los métodos para una lista)
 - Puede ignorar los siguientes métodos:
 - hashCode()
 - iterator()
 - listIterator()
 - listIterator(int index)
 - retainAll(Collection<?> c)
 - toArray()
 - toArray(T[] a)
 - Implemente una clase Node<T> donde T es un tipo genérico, esta clase debe contener al menos dos propiedades.
 - T data: la información almacenada en el nodo Node<T> nextNode: una referencia al siguiente nodo
 - Implementar una clase List<T> esta clase debe contener al menos esta propiedad
 - Node<T> root: la referencia sobre el nodo inicial

```

Lab3 > Ej_propuestos > Ej_propuesto01 > Node.java > Node<T> > Node(T)
1  package Lab3.Ej_propuestos.Ej_propuesto01;
2
3  public class Node<T> {
4      private T data;                // Información del nodo
5      private Node<T> nextNode;      // Referencia al siguiente nodo
6      private Node<T> previousNode;  // Referencia al nodo anterior (si la lista es doble)
7      private boolean isDeleted;     // Marcado lógico para "eliminación suave"
8      private int position;          // Posición en la lista (rastreo)
9
10     public Node(T data) {
11         this.data = data;
12         this.nextNode = null;
13         this.previousNode = null;
14         this.isDeleted = false;
15         this.position = -1;
16     }

```

- La imagen muestra la clase Node<T>, que representa un nodo genérico en una lista enlazada. Contiene los atributos data (dato almacenado), nextNode y previousNode (referencias al siguiente y anterior nodo), isDeleted (marca lógica de eliminación) y position (posición en la lista). El constructor inicializa estos valores, dejando las referencias en null, la posición en -1 y la marca de eliminación en false, preparando el nodo para su uso en estructuras dinámicas.

```
ab3 > Ej_propuestos > Ej_propuesto01 > List.java > ...
1  package Lab3.Ej_propuestos.Ej_propuesto01;
2
3  public class List<T> {
4      private Node<T> root;           // Primer nodo
5      private Node<T> tail;          // Último nodo
6      private int size;               // Tamaño de la lista
7      private String name;            // Nombre opcional de la lista
8      private int idCounter;          // Contador interno de posición
9
10     // Constructor por defecto (sin forzar nada)
11     public List() {
12         this.root = null;
13         this.tail = null;
14         this.size = 0;
15         this.name = "Lista";         // Nombre por defecto
16         this.idCounter = 0;
17     }
18
19     // Agregar al final
20     public void add(T data) {
21         Node<T> newNode = new Node<>(data);
22         newNode.setPosition(idCounter++); // Asignar posición lógica
23         if (root == null) {
24             root = tail = newNode;
25         } else {
26             tail.setNextNode(newNode);
27             tail = newNode;
28         }
29         size++;
30     }
31
```

```
32     // Obtener un elemento por índice
33     public T get(int index) {
34         if (index < 0 || index >= size) throw new IndexOutOfBoundsException();
35         Node<T> current = root;
36         for (int i = 0; i < index; i++) {
37             current = current.getNextNode();
38         }
39         return current.getData();
40     }
41
42     // Eliminar por índice
43     public void remove(int index) {
44         if (index < 0 || index >= size) throw new IndexOutOfBoundsException();
45         if (index == 0) {
46             root = root.getNextNode();
47             if (root == null) tail = null;
48         } else {
49             Node<T> previous = root;
50             for (int i = 0; i < index - 1; i++) {
51                 previous = previous.getNextNode();
52             }
53             Node<T> toDelete = previous.getNextNode();
54             previous.setNextNode(toDelete.getNextNode());
55             if (toDelete == tail) tail = previous;
56         }
57         size--;
58     }
59
```

```
59
60 // Verificar si contiene un dato
61 public boolean contains(T data) {
62     Node<T> current = root;
63     while (current != null) {
64         if (current.getData().equals(data)) return true;
65         current = current.getNextNode();
66     }
67     return false;
68 }
69
70 // Obtener tamaño
71 public int size() {
72     return size;
73 }
74
75 // Vaciar lista
76 public void clear() {
77     root = null;
78     tail = null;
79     size = 0;
80     idCounter = 0;
81 }
82
83 // Verificar si está vacía
84 public boolean isEmpty() {
85     return size == 0;
86 }
87
```

- La clase List<T> implementa una lista enlazada genérica que permite almacenar, recorrer y gestionar elementos dinámicamente. Incluye atributos como root, tail, size y idCounter, y métodos fundamentales como add, get, remove, contains, clear, isEmpty y printList, permitiendo una gestión eficiente y flexible de datos.

	<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>

```

23 public static void main(String[] args) {
24     List<String> lista = new List<>();
25
26     lista.add(data:"Uno");
27     lista.add(data:"Dos");
28     lista.add(data:"Tres");
29
30     lista.printList(); // Uno -> Dos -> Tres -> null
31
32     System.out.println("Contiene 'Dos'? " + lista.contains(data:"Dos")); // true
33     lista.remove(index:1);
34     lista.printList(); // Uno -> Tres -> null
35
36     System.out.println("Elemento en posición 1: " + lista.get(index:1)); // Tres
37     System.out.println("Tamaño: " + lista.size()); // 2
38     lista.clear();
39     System.out.println("¿Vacía? " + lista.isEmpty()); // true
40 }
41
42

```

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```

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.jav
j_propuestos,Ej_propuesto01.Main'
[Lista] (0) Uno -> (1) Dos -> (2) Tres -> null
Contiene 'Dos'? true
[Lista] (0) Uno -> (2) Tres -> null
Elemento en posición 1: Tres
Tamaño: 2
¿Vacía? true
PS C:\Users\AORUS\OneDrive\Documentos\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

```

- La ejecución demuestra correctamente el funcionamiento de los métodos add, remove, contains, get, size, clear, isEmpty y printList, validando que la lista enlazada genérica opera correctamente. Los resultados concuerdan con las acciones realizadas, mostrando que la implementación es funcional y efectiva.

ii. Calculadora Genérica, Cree un nuevo proyecto en Java: CalculadoraGenerica.

- Implementar las clases Genérica Operador, para declarar sus atributos (valor1 y valor2), constructor (Operador).
- Escribir la clase Main, e implementar el método genérico suma con la siguiente estructura.
- Escribir los métodos genéricos: resta, producto, división, potencia, raíz cuadrada y raíz cúbica.
- Para poder probar las clases y métodos genéricos implementar mediante un menú de opciones las operaciones, mostrando los resultados:
- Menú de Operaciones Clases Genéricas:**

→ 1. Suma

→ 2. Resta

→ 3. Producto

→ 4. División


→ 5. Potencia

→ 6. Raíz Cuadrada


→ 7. Raíz Cúbica

→ 8. Salir del Programa

- **Nota:** El programa debe permitir validar entre valores o tipo de dato (integer o double) para poder utilizar los métodos genéricos, el programa no termina hasta escoger la opción SALIR.

```
Lab3 > Ej_propuestos > Ej_propuesto02 >  Operador.java > ...  
1  package Lab3.Ej_propuestos.Ej_propuesto02;  
2  
3  public class Operador<T extends Number> {  
4      private T valor1;  
5      private T valor2;  
6  
7      public Operador(T valor1, T valor2) {  
8          this.valor1 = valor1;  
9          this.valor2 = valor2;  
10     }  
11  
12     public T getValor1() {  
13         return valor1;  
14     }  
15  
16     public T getValor2() {  
17         return valor2;  
18     }  
19 }  
20
```

- La clase Operador<T> es una clase genérica que almacena dos valores numéricos (valor1 y valor2) del tipo T, donde T debe extender de Number. Esta clase encapsula los operandos que se utilizan en las operaciones matemáticas genéricas, facilitando la reutilización de código con distintos tipos numéricos (Integer, Double, etc.).


```
Lab3 > Ej_propuestos > Ej_propuesto02 >  CalculadoraGenerica.java > ...
1  package Lab3.Ej_propuestos.Ej_propuesto02;
2
3  public class CalculadoraGenerica {
4
5      public static <T extends Number> double suma(T a, T b) {
6          return a.doubleValue() + b.doubleValue();
7      }
8
9      public static <T extends Number> double resta(T a, T b) {
10         return a.doubleValue() - b.doubleValue();
11     }
12
13     public static <T extends Number> double producto(T a, T b) {
14         return a.doubleValue() * b.doubleValue();
15     }
16
17     public static <T extends Number> double division(T a, T b) {
18         if (b.doubleValue() == 0) {
19             throw new ArithmeticException(s:"División por cero");
20         }
21         return a.doubleValue() / b.doubleValue();
22     }
23
24     public static <T extends Number> double potencia(T a, T b) {
25         return Math.pow(a.doubleValue(), b.doubleValue());
26     }
27
28     public static <T extends Number> double raizCuadrada(T a) {
29         return Math.sqrt(a.doubleValue());
30     }
31
32     public static <T extends Number> double raizCubica(T a) {
33         return Math.cbrt(a.doubleValue());
34     }
35 }
36
```

- Esta clase contiene los métodos genéricos que realizan las operaciones matemáticas: suma, resta, producto, división, potencia, raíz cuadrada y raíz cúbica. Todos los métodos trabajan con tipos T extends Number y devuelven un resultado en double, utilizando doubleValue() para convertir los valores genéricos a un tipo numérico compatible. La clase abstrae la lógica matemática del programa.

	<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: 10</p>

```

5 public class Main {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8         int opcion;
9
10        do {
11            System.out.println(x: "\nMenú de Operaciones Clases Genéricas:");
12            System.out.println(x: "1. Suma");
13            System.out.println(x: "2. Resta");
14            System.out.println(x: "3. Producto");
15            System.out.println(x: "4. División");
16            System.out.println(x: "5. Potencia");
17            System.out.println(x: "6. Raíz Cuadrada");
18            System.out.println(x: "7. Raíz Cúbica");
19            System.out.println(x: "8. Salir del Programa");
20            System.out.print(s: "Seleccione una opción: ");
21            opcion = scanner.nextInt();
22
23            if (opcion >= 1 && opcion <= 5) {
24                System.out.print(s: "Ingrese el primer número: ");
25                double n1 = scanner.nextDouble();

```

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** PUERTOS

```

Menú de Operaciones Clases Genéricas:
1. Suma
2. Resta
3. Producto
4. División
5. Potencia
6. Raíz Cuadrada
7. Raíz Cúbica
8. Salir del Programa
Seleccione una opción: 2
Ingrese el primer número: 5
Ingrese el segundo número: 6
Resultado: -1.0

Menú de Operaciones Clases Genéricas:

```

rate UML Java: Ready Lin. 74, col. 1 Espacios: 4 UTF-

- La clase Main implementa el menú interactivo que permite al usuario seleccionar una operación, ingresar valores, y mostrar el resultado correspondiente en consola. Es el punto de ejecución del programa y se encarga de coordinar la entrada del usuario con las llamadas a los métodos de la clase CalculadoraGenerica. También valida la opción seleccionada y mantiene el programa en ejecución hasta que el usuario elige salir.

	<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: 11</p>

II. SOLUCIÓN DEL CUESTIONARIO

a. **¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.**

- Comprender el uso correcto de genéricos con restricciones (<T extends Number>).
- No poder utilizar operadores aritméticos directamente sobre tipos genéricos (como +, -, *), lo que obligó a usar doubleValue().
- Diferenciar cuándo usar Integer y Double, y convertir adecuadamente los valores ingresados.
- Estructurar el programa de forma modular (separando lógica de cálculo, entrada de datos y menú).
- La documentación oficial es extensa, pero requiere ejemplos prácticos adicionales para entender bien su aplicación.

b. **¿Qué diferencia hay entre un List y un ArrayList en Java?**

- List es una interfaz que define métodos para estructuras de datos lineales.
- ArrayList es una clase concreta que implementa la interfaz List usando un arreglo dinámico.
- Al declarar con List, se puede cambiar fácilmente a otra implementación (como LinkedList) sin modificar el resto del código.
- ArrayList es eficiente para acceso rápido por índice, pero menos eficiente para inserciones intermedias.

c. **¿Qué beneficios y oportunidades ofrecen las clases genéricas en Java?**

Las clases genéricas en Java ofrecen una forma poderosa y flexible de crear estructuras de datos y métodos reutilizables que pueden trabajar con distintos tipos de datos sin necesidad de duplicar código. Al usar genéricos, es posible definir clases o métodos que funcionen con cualquier tipo de objeto, mientras se mantiene la seguridad de tipos en tiempo de compilación.

Uno de los beneficios más importantes es la reutilización de código, ya que con una sola implementación se pueden manejar diferentes tipos de datos. Por

	<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>

ejemplo, una clase genérica Caja<T> puede almacenar un Integer, un String, o cualquier otro tipo, sin necesidad de definir versiones específicas para cada uno.

Además, los genéricos mejoran la seguridad del programa, ya que el compilador puede detectar errores de tipo antes de ejecutar el código. Esto evita errores comunes como castings inválidos o pérdida de tipo, lo que también mejora la legibilidad y el mantenimiento del código.

Otro beneficio clave es que se elimina la necesidad de realizar conversiones explícitas de tipo (castings), lo que reduce el riesgo de errores y hace el código más limpio y fácil de entender. En resumen, los genéricos permiten escribir código más claro, seguro y reutilizable.

III. CONCLUSIONES

- La implementación de clases y métodos genéricos en Java permite crear programas más flexibles, reutilizables y seguros en tiempo de compilación.
- El ejercicio reforzó la comprensión de conceptos fundamentales como clases genéricas, encapsulamiento y control de flujo mediante menús interactivos en consola.
- Esta práctica demuestra cómo Java permite crear aplicaciones robustas con estructuras reutilizables aplicables a distintos contextos numéricos.

REFERENCIAS Y BIBLIOGRAFÍA

- Oracle. (n.d.). *Generics (The Java™ Tutorials > Learning the Java Language > Generics)*. Recuperado de: <https://docs.oracle.com/javase/tutorial/java/generics/index.html>
- Oracle. (n.d.). *Why Use Generics?* Recuperado de: <https://docs.oracle.com/javase/tutorial/java/generics/why.html>
- Oracle. (n.d.). *Class Number.* Recuperado de: <https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html>
- GeeksforGeeks. (s.f.). *Generics in Java.* Recuperado de: <https://www.geeksforgeeks.org/generics-in-java/>
- Oracle. (n.d.). *Class ArrayList<E>.* Recuperado de: <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>
- Oracle. (n.d.). *Interface List<E>.* Recuperado de: <https://docs.oracle.com/javase/8/docs/api/java/util/List.html>