

	<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: 2025/05/03</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATUR A:	Programacion Web 2				
TÍTULO DE LA PRÁCTICA:	Laboratorio 05				
NÚMERO DE PRÁCTICA:	5	AÑO LECTIVO:	2025	NRO. SEMESTRE:	1
FECHA DE PRESENTACIÓN	21/05/2024	Repositorio	https://github.com/code50/82924112.git		
INTEGRANTE (s): Silva Pino Jesus Francisco				NOTA:	
DOCENTE(s): Edson Luque Mamani					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>EJERCICIOS PROPUESTOS</p> <p>i. Invertir un matriz de enteros Ejemplo: A=[1 2 3] -> Ain=[3 2 1]</p> <ul style="list-style-type: none"> <pre>public int[] invertirArray(int[] A){ /** */ //Procedimiento para invertir la matriz /** */ return Ain; }</pre> Codigo Utilizado <pre>public class Ejercicio1 { public static void main(String[] args) {</pre>

```
int[] arr1 = {1, 2, 3};  
int[] arrInvertido = invertirArray(arr1);
```

```
System.out.print("Original: ");  
imprimirArray(arr1);  
System.out.print("Invertido: ");  
imprimirArray(arrInvertido);  
}
```

```
public static int[] invertirArray(int[] A) {  
    int[] ArrayIn = new int[A.length];  
    for (int i = 0; i < A.length; i++) {  
        ArrayIn[i] = A[A.length - 1 - i];  
    }  
    return ArrayIn;  
}
```

```
public static void imprimirArray(int[] arr) {  
    System.out.print("[");  
    for (int i = 0; i < arr.length; i++) {  
        System.out.print(arr[i]);  
        if (i < arr.length - 1) {  
            System.out.print(", ");  
        }  
    }  
    System.out.println("]");  
}
```

```
$ cd /workspaces/82924112 ; /usr/bin/env /opt/jdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/ubuntu/.vscode-remote/data/User/workspaceStorage/-2f00f915  
/redhat.java/jdt_ws/82924112_3dd81b53/bin Ejercicio1  
Original: [1, 2, 3]  
Invertido: [3, 2, 1]  
$ ^C
```

ii. Rotación a la Izquierda Ejemplo: Si $d=2$ $A=[1\ 2\ 3\ 4\ 5]$ \rightarrow $Aiz=[3\ 4\ 5\ 1\ 2]$

- ```
public int[] rotarIzquierdaArray(int[] A){
 /** */
 //Procedimiento para rotar la matriz
 /** */
 return Aiz;
}
```

- Código Utilizado

```
public class Ejercicio2 {

 public static void main(String[] args) {

 int[] arrOriginal = { 1, 2, 3, 4, 5 };

 int rotaciones = 2;

 int[] arrRotado = rotarIzquierda(arrOriginal, rotaciones);

 System.out.print("Original: ");
 imprimirArray(arrOriginal);
 System.out.print("Rotado " + rotaciones + " posiciones: ");
 imprimirArray(arrRotado);
 }

 public static int[] rotarIzquierda(int[] A, int d) {
 if (A == null || A.length == 0 || d < 0) {
 System.out.println("Parámetros inválidos");
 return null;
 }

 int[] Aiz = new int[A.length];
 for (int i = 0; i < A.length; i++) {
 Aiz[i] = A[(i + d) % A.length];
 }
 return Aiz;
 }

 public static void imprimirArray(int[] arr) {
 System.out.print("[");
 for (int i = 0; i < arr.length; i++) {
 System.out.print(arr[i]);
 if (i < arr.length - 1) {
 System.out.print(", ");
 }
 }
 System.out.println("]");
 }
}
```

```
$ cd /workspaces/82924112 ; /usr/bin/env /opt/jdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/ubuntu/.vscode-remote/data/User/workspaceStorage/-2f00f915/redhat.java/jdt_ws/82924112_3dd81b53/bin Ejercicio2
Original: [1, 2, 3, 4, 5]
Rotado 2 posiciones: [3, 4, 5, 1, 2]
$
```

### iii. Triángulo recursivo Ejemplo: b=5

```
*
**


```

```
public void trianguloRecursivo(int base){
/** */
//Procedimiento para imprimir triángulo
/** */
}
```

Codigo Utilizado

```
public class Ejercicio3 {
```

```
public static void main(String[] args) {
```



```
int base = 8;
System.out.println("Triángulo con base " + base + ":");
trianguloRecursivo(base);
}
```

```
public static void trianguloRecursivo(int base) {
```

```
if (base < 1) {
return;
}
```

```
trianguloRecursivo(base - 1);
```

```
for (int i = 0; i < base; i++) {
System.out.print("*");
}
System.out.println();
}
```

|                                                                                    |                                                                                                                                                      |                                                                                     |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|   | <p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN<br/>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y<br/>SERVICIOS<br/>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> |                                                                                                                                                      |                                                                                     |
| <p>Aprobación: 2025/05/03</p>                                                      | <p>Código: GUIA-PRLE-001</p>                                                                                                                         | <p>Página: 5</p>                                                                    |

```
}

```

```
$ cd /workspaces/82924112 ; /usr/bin/env /opt/jdk/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/ubuntu/.vscode-remote/data/User/workspaceStorage/-2f00f915/redhat.java/jdt_ws/82924112_3dd81b53/bin Ejercicio3
Triángulo con base 8:
*
**

$
```

## II. SOLUCIÓN DEL CUESTIONARIO

¿Qué diferencia hay entre un List y un ArrayList en Java? ¿Qué beneficios y oportunidades ofrecen las clases genéricas en Java?

En Java, List es una **interfaz**, mientras que ArrayList es una **clase concreta** que implementa la interfaz List.

### Ejemplo de uso:



```
List<String> lista = new ArrayList<>(); // Recomendado (flexibilidad)
ArrayList<String> arrayList = new ArrayList<>(); // Menos flexible
```

### Beneficios de las clases genéricas en Java

Las clases genéricas (**Generics**) permiten definir tipos de datos como parámetros, lo que mejora la seguridad y reutilización del código. Sus principales ventajas son:

#### 1. Seguridad de tipos (Type Safety)

- **Evita errores en tiempo de ejecución** al garantizar que solo se usen los tipos

|                                                                                                                       |                                                                                                                                                                                  |                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|                                      | <p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN<br/>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y<br/>SERVICIOS<br/>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> |                                                                                                                                                                                  |                                                                                     |
| <p><b>Aprobación:</b> 2025/05/03</p>                                                                                  | <p><b>Código:</b> GUIA-PRLE-001</p>                                                                                                                                              | <p><b>Página:</b> 6</p>                                                             |

especificados.

- **Ejemplo sin Generics** (problema):

```
List lista = new ArrayList();
```

```
lista.add("Hola");
```

```
lista.add(123); // Compila, pero puede causar ClassCastException después.
```

- **Ejemplo con Generics** (solución):

```
List<String> lista = new ArrayList<>();
```

```
lista.add("Hola");
```

```
lista.add(123); // Error de compilación (¡detectado temprano!).
```

## 2. Reutilización de código

- **Una misma clase/método puede trabajar con múltiples tipos** sin duplicar implementaciones.

```
public class Caja<T> {
 private T contenido;
 public void setContenido(T contenido) { this.contenido = contenido; }
 public T getContenido() { return contenido; }
}
```

```
Caja<String> cajaStr = new Caja<>();
```

```
Caja<Integer> cajaInt = new Caja<>();
```

## 3. Eliminación de casts innecesarios

- **Sin Generics:**

```
List lista = new ArrayList();
```

```
String texto = (String) lista.get(0); // Cast explícito necesario.
```

- **Con Generics:**

```
List<String> lista = new ArrayList<>();
```

```
String texto = lista.get(0); // Sin cast.
```

## 4. Compatibilidad con algoritmos genéricos

- **Colecciones como Collections.sort()** pueden trabajar con cualquier tipo que implemente Comparable<T>:

```
List<Integer> numeros = Arrays.asList(3, 1, 4);
```

|                                                                                    |                                                                                                                                                      |                                                                                     |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|   | <p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN<br/>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y<br/>SERVICIOS<br/>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p> |  |
| <p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p> |                                                                                                                                                      |                                                                                     |
| <p>Aprobación: 2025/05/03</p>                                                      | <p>Código: GUIA-PRLE-001</p>                                                                                                                         | <p>Página: 7</p>                                                                    |

Collections.sort( numeros ); // Funciona porque Integer implementa Comparable<Integer>.

### 5. Mejor legibilidad del código

- El código es más expresivo al indicar claramente qué tipos se esperan.

## III. CONCLUSIONES

- **List vs ArrayList:** La interfaz List ofrece flexibilidad para cambiar implementaciones, mientras que ArrayList es una implementación eficiente para acceso aleatorio.
- **Generics:** Proporcionan seguridad de tipos, reutilización y claridad, evitando errores comunes en tiempo de compilación.

## RETROALIMENTACIÓN GENERAL

## REFERENCIAS Y BIBLIOGRAFÍA