
	<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-PRLD-001	Página: 1



GUÍA DE LABORATORIO

(formato docente)

INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE DATOS Y ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	PILAS Y COLAS				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2025 – A	NRO. SEMESTRE:	TERCERO III
TIPO DE PRÁCTICA:	INDIVIDUAL	X			
	GRUPAL	—	MÁXIMO DE ESTUDIANTES	00	
FECHA INICIO:	02/06/2025	FECHA FIN:	06/06/2025	DURACIÓN:	90 minutos.
RECURSOS A UTILIZAR: <ul style="list-style-type: none"> • Github. • Lenguaje de Programación Java. • Ide Java Eclipse/Visual Studio Code. 					
DOCENTE(s): <ul style="list-style-type: none"> • Mg. Ing. Rene Alonso Nieto Valencia. 					

OBJETIVOS/TEMAS Y COMPETENCIAS	
OBJETIVOS: <ul style="list-style-type: none"> • Aprenda Pilas y Colas. • Aplicar conceptos elementales de programación a resolver utilizando POO en problemas de algoritmos. • Desarrollar pruebas. 	
TEMAS: <ul style="list-style-type: none"> • Introducción. • Pilas y operaciones. • Colas y operaciones. 	
COMPETENCIAS	C.a C.b C.c C.d

CONTENIDO DE LA GUÍA

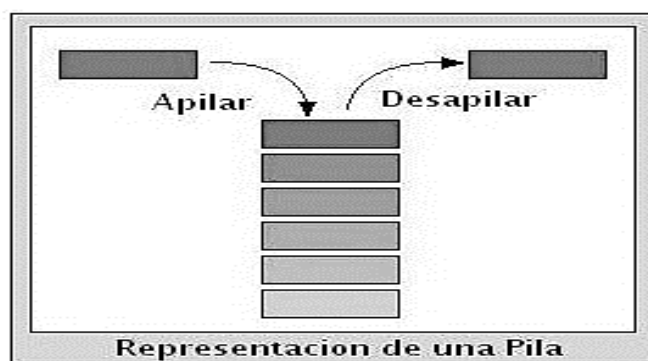
	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01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

I. MARCO CONCEPTUAL

- <https://www.w3schools.com/java/>
- <https://www.eclipse.org/downloads/packages/release/2022-03/r/eclipse-ide-enterprisejava-and-web-developers>
- <https://javaplot.panayotis.com/>
- <https://sourceforge.net/projects/gnuplot/files/gnuplot/5.4.3/>

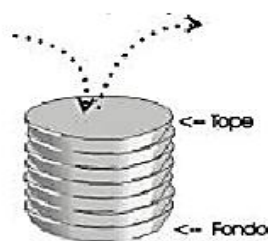
1. PILA.

Una **PILA** es una estructura que nos permite **apilar** elementos y recopilarlos en el orden inverso al cual los apilamos mediante operaciones de **desapilar**.



En todo momento sólo se tiene acceso a la parte superior de la pila, es decir, al último objeto apilado. Por esta razón, a una pila se le conoce como una estructura de datos **LIFO (Last In First Out)**.

Estructura LIFO (*Last In First Out*): "último en entrar primero en salir"



Operaciones de las Cola:

Las operaciones deben obedecer a la manera como este tipo de lista accede a los datos (LIFO).

Entre las principales operaciones están las siguientes:

- **push(x)**: inserta en la pila el elemento 'x'. (apilar)
- **pop()**: elimina de la pila el elemento que está en la cima o el tope de la pila. (desapilar)
- **top()**: recupera el elemento del tope o cima de la pila. (tope)
- **destroyStack()**: elimina todos los elementos, dejando la pila vacía.
- **isEmpty()**: verifica si la pila está vacía.
- **isFull()**: verifica si la pila está llena. Se usa cuando la pila esta implementa sobre una estructura de datos estática

Implementación Genérica de la Pila.

Interfaz (Java)

```
public interface Pila<E>
{
    boolean estaVacia();
    E cima();
    void apilar(E elem);
    void desapilar();
}
```

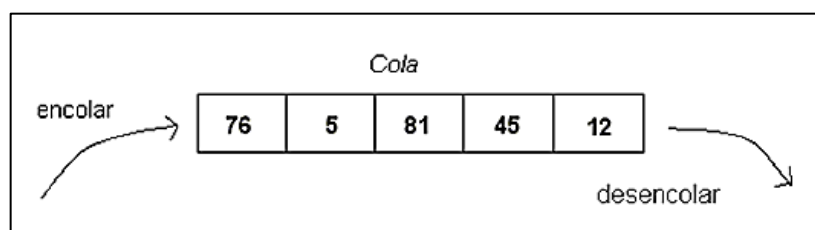
```
public class PilaArray<E> implements Pila<E>
{
    private E[] vec;
    private int tam;
    public PilaArray() {
        vec = (E[]) new Object[100];
        tam = 0;
    }
    boolean estaVacia() { return tam == 0; }
    E cima() { return vec[tam-1]; }
    void apilar(E elem) {
        if(tam == vec.length) {
            Object[] tmp = new Object[2*tam];
            System.arraycopy(vec, 0, tmp, 0, tam-1);
            vec = (E[]) tmp;
        }
        vec[tam++] = elem;
    }
    void desapilar() { return vec[--tam]; }
}
```



2. COLA.

En una **Cola** los elementos se añaden desde la parte de atrás o la parte final de la cola, sin embargo, la información se extrae desde el frente, es decir, los elementos que se añadieron primero serán los primeros en salir, esto se conoce como estructura **FIFO (First In First Out)**.

La adición de elementos se realiza a través de una operación llamada **encolar (enqueue)**, mientras que la eliminación se denomina **desencolar (dequeue)**.

La operación de **encolar** inserta elementos por un extremo de la cola, mientras que la de **desencolar** los elimina por el otro.



	<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-PRLD-001</p>	<p>Página: 4</p>

Operaciones de la Cola.

Las operaciones que se aplican a las colas obedecen a la forma como esta lista accede y/o manipula la información almacenada en ella, las cuales son:



- **enqueue(x):** agrega el elemento 'x' al final de la cola. La cola debe existir y no estar llena. (encolar).
- **dequeue():** elimina el elemento que se encuentra al frente (front) de la cola. La cola debe existir y no estar vacía. (desencolar).
- **destroyQueue():** elimina los elementos de la cola dejándola vacía.
- **isEmpty():** verifica si la cola está o no vacía.
- **isFull():** verifica si la cola está llena o no. Se usa cuando la cola está implementada sobre una estructura estática.
- **front():** retorna el elemento que se encuentra al frente de la cola (primer elemento)
- **back():** retorna el último elemento de la cola.

Implementación Genérica de la Pila.

```
class ColaArreglo
{
    private Object[] arreglo;
    private int primero, ultimo, numElem;
    private int MAX_ELEM=100; // maximo numero de elementos en la cola

    public ColaArreglo()
    {
        arreglo=new Object[MAX_ELEM];
        primero=0;
        ultimo=MAX_ELEM-1;
        numElem=0;
    }

    public void encolar(Object x)
    {
        if (numElem<MAX_ELEM) // si esta llena se produce OVERFLOW
        {
            ultimo=(ultimo+1)%MAX_ELEM;
            arreglo[ultimo]=x;
            numElem++;
        }
    }
}
```

	<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-PRLD-001</p>	<p>Página: 5</p>

```

public Object sacar()
{
    if (!estaVacia()) // si esta vacia se produce UNDERFLOW
    {
        Object x=arreglo[primero];
        primero=(primero+1)%MAX_ELEM;
        numElem--;
        return x;
    }
}

public boolean estaVacia()
{
    return numElem==0;
}

```

II. EJERCICIO/PROBLEMA RESUELTO POR EL DOCENTE

En un editor Java, realizar la integración de los siguientes ejercicios, revisar y mostrar los resultados obtenidos y realizar una explicación del funcionamiento de forma concreta y clara.

1. **Ejercicio 1:** Implementar una Pila utilizando una clase **StackList** y una clase nodo e **ingresar** los elementos **1, 2, 3, 4, 5, 6, 7 y 8**. De acuerdo a la implementación del marco teórico utilizando clases y métodos genéricos.

Referencia: <https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html>



2. **Ejercicio 2:** Implementar una Cola utilizando una clase **QueueList** y una clase nodo e **ingresar** los elementos **1, 2, 3, 4, 5, 6, 7 y 8**. De acuerdo a la implementación del marco teórico utilizando clases y métodos genéricos.

Referencia: <https://docs.oracle.com/javase/7/docs/api/java/util/AbstractQueue.html>

III. EJERCICIOS/PROBLEMAS PROPUESTOS

De acuerdo a los ejercicios propuestos desarrollar los algoritmos y mostrar las siguientes indicaciones:

- Enunciado del ejercicio.
- Código en java desarrollado.
- Resultados obtenidos.
- Explicación breve y concreta del código implementado.

	<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-PRLD-001</p>	<p>Página: 6</p>

Referencia: <https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html>

Usando la clase genérica: `public class Stack<E>` usar métodos genéricos `public E Metodo()`

1. Implementar una **Pila** que tenga los elementos del 1 al 10, usando la clase nodo en java.
2. Implementar una **Pila** que tenga los elementos del 1 al 10, usando la clase nodo en java y los **métodos** vistos en el **marco teórico** (push, pop, top, destroyStack, isEmpty, isFull, printStack) y probar una clase **Principal** con un menú de opciones para probar los métodos.

Referencia: <https://docs.oracle.com/javase/7/docs/api/java/util/AbstractQueue.html>

Usando la clase genérica: `public class Queue<E>` usar métodos genéricos `public E Metodo()`

3. Implementar una **Cola** que tenga los elementos del 1 al 10, usando la clase nodo en java.
4. Implementar una **Cola** que tenga los elementos del 1 al 10, usando la clase nodo en java y los **métodos** vistos en el **marco teórico** (enqueue, dequeue, destroyQueue, isEmpty, isFull, front, back, printQueue) y probar una clase **Principal** con un menú de opciones para probar los métodos.

IV. CUESTIONARIO


1. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.
2. ¿Es posible reutilizar la clase nodo para otras estructuras de datos, además de listas enlazadas, pilas y colas?
3. ¿Qué tipo de dato es NULL en java?
4. ¿Cuáles son los beneficios de utilizar tipos genéricos en las pilas y colas?

V. REFERENCIAS Y BIBLIOGRAFÍA RECOMENDADAS:

- 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
- The Java™ Tutorials - <https://docs.oracle.com/javase/tutorial/>
- Sedgewick, R., Algorithms in Java, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching, Part 5:
- Graph Algorithms, Addison-Wesley.
- Malik D., Data Structures Using C++, 2003, Thomson Learning.
- Knuth D., The Art of Computer Programming, Vol. 1 y 3, Addison - Wesley.

TÉCNICAS E INSTRUMENTOS DE EVALUACIÓN

<p>TÉCNICAS: <i>Actividades Resueltas</i> <i>Ejercicios Propuestos</i></p>	<p>INSTRUMENTOS: <i>Rubricas</i></p>
---	--

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

CRITERIOS DE EVALUACIÓN

Los criterios de evaluación se encuentran en el silabo DUFA ANEXO en la sección EVOLUCIÓN CONTINUA