TAD Pila

Práctica 6
Pilas con Arreglo

Objetivo

El objetivo de esta práctica es que el alumno refuerce sus conocimientos acerca del TAD Pila a través de la implementación de dos pilas haciendo uso de un arreglo simple.

Descripción general

Se programará la clase `PilasConArreglo` que implementará a la interfaz `PilaDoble`. Dicha clase implementará dos pilas usando solamente un arreglo simple, es decir, las pilas deberán almacenar sus elementos en un solo arreglo. Cada pila será identificada con un valor entero, por ejemplo, la primer pila con el valor 1 y la segunda con el valor 2. Por ello cada método definido en la interfaz `PilaDoble` recibe un identificador de valor entero que corresponde a la pila sobre la que se trabaja.
El arreglo tendrá una capacidad limitada, sin embargo se requiere que, siempre que haya espacio en el arreglo, sea posible agregar elementos a las pilas. Si las pilas colisionan, se indicará con un mensaje en pantalla que el arreglo se encuentra lleno.
Se busca que las operaciones de la estructura de datos sean de tiempo constante y que el uso del espacio del arreglo sea eficiente. **Ayuda**: las pilas en el arreglo no tienen que crecer en la misma dirección.

Material

El material de esta práctica consta de los siguientes archivos:

- **PilaDoble.class** interfaz para el tipo abstracto de datos Pila.
- **PilasConArreglo.java** clase que implementará la interfaz `PilaDoble`.
- **PruebaPilas.class** programa para probar la clase `PilasConArreglo`.
- Documentación:
    - PilaDoble.html documentación de la interfaz `PilaDoble`.

Desarrollo

1. Descargar los archivos `PilaDoble.class, PilasConArreglo.java y PruebaPilas.class` en el directorio donde se va a desarrollar la práctica.

2. Leer la documentación y revisar el material de la práctica. Recordar que la firma de los métodos proporcionados y de los requeridos en el desarrollo de la práctica no podrán ser modificados.

3. Implementar los siguientes métodos en la clase `PilasConArreglo`:

    a. **Constructor por omisión** que crea dos pilas en un arreglo con capacidad para 10 elementos.
    b. **Constructor** de la clase que crea dos pilas y recibe como parámetro un valor entero para definir el tamaño del arreglo que contiene a las pilas. La capacidad mínima aceptada del arreglo debe ser de 2 elementos.
    c. Los **métodos** especificados en la interfaz `PilaDoble`. El iterador no debe soportar la operación *remove*.

4. Ejecutar el programa `PruebaPilas` para probar la implementación realizada. Si el programa funciona adecuadamente se verán los siguientes mensajes:

```
Creando estructura de datos PilasConArreglo para 10 elementos...

Insertando elementos "A","B","C","D","E","F" a la Pila 1 (Push)...

Insertando elementos "G","H","I","J" a la Pila 2 (Push)...

Iterando elementos de la Pila 1
F
E
D
C
B
A

Iterando elementos de la Pila 2
J
I
H
G

Top Pila1: F
Top Pila2: J

Insertar elemento "K" a la Pila 2 (Push)...
El arreglo se encuentra lleno (pila 2).
```

```
Pop Pila1: F
Pop Pila1: E
Pop Pila1: D
Pop Pila2: J

Top Pila1: C
Top Pila2: I

Iterando elementos de la Pila 1
C
B
A
Iterando elementos de la Pila 2
I
H
G

Vaciando solamente la Pila 1...

¿Esta vacia la Pila 1? true
¿Esta vacia la Pila 2? false
```

5. Escribir una tabla con la complejidad de los métodos programados en la clase `PilasConArreglo`.

Operación	**Tiempo de ejecución**
estaVacia	
vaciar	
top	
pop	
push	
iterador	

Ejercicio opcional

Se propone desarrollar una variación de esta estructura de datos en la que se implementen las pilas dobles haciendo uso de un arreglo dinámico. Al agregar un elemento, el arreglo doblará su tamaño cuando este se encuentre lleno, mientras que al eliminar un elemento, su capacidad se reducirá a la mitad si se encuentra ocupado en una cuarta parte.