



Programación II

Pilas

Ing. Weiss Gonzalo



Temario

- *TDA*
- *Pila*
- *Especificación*
- *Ejemplos*
- *Implementación estática*

TDA

- Es una **abstracción**, ignoramos algunos detalles y nos concentramos en los que nos interesan.
- A la definición del TDA la llamamos **especificación** y a la forma de llevar a cabo lo definido lo denominamos **implementación**.

Recordar que:

Existen siempre 2 visiones diferentes en el TDA: usuario e implementador.

Son separadas, y una oculta a la otra.



Pilas

Pilas

*La pila es una estructura que permite almacenar valores, eliminarlos y recuperarlos, con la particularidad de que **el elemento que se recupera o elimina es el último que ingresó.***

*Es por esto que una pila es lo que se suele llamar una estructura **LIFO** (del inglés Last In, First Out).*

Pilas - Especificación

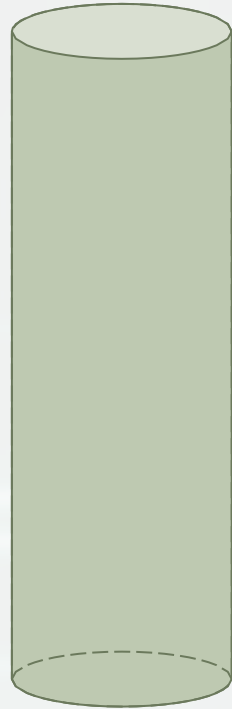
*Las operaciones que necesitaremos son: agregar y eliminar datos de la pila (que llamaremos **apilar** y **desapilar**), consultar el valor del primer elemento (que llamaremos **tope**) y consultar si la pila está o no vacía (a esta consulta la llamaremos **pilaVacía**). Además agregaremos la inicialización de una pila (que llamaremos **inicializarPila**).*

Pilas - Especificación - Operaciones

- *inicializarPila*: permite inicializar la estructura de la pila.
- *apilar*: permite agregar un elemento a la pila (se supone que la pila está inicializada).
- *desapilar*: permite eliminar el último elemento agregado a la pila (se supone que la pila está inicializada y no está vacía).
- *tope*: permite conocer cuál es el último elemento ingresado a la pila (se supone que la pila está inicializada y no está vacía).
- *pilaVacía*: indica si la pila contiene elementos o no (se supone que la pila está inicializada).

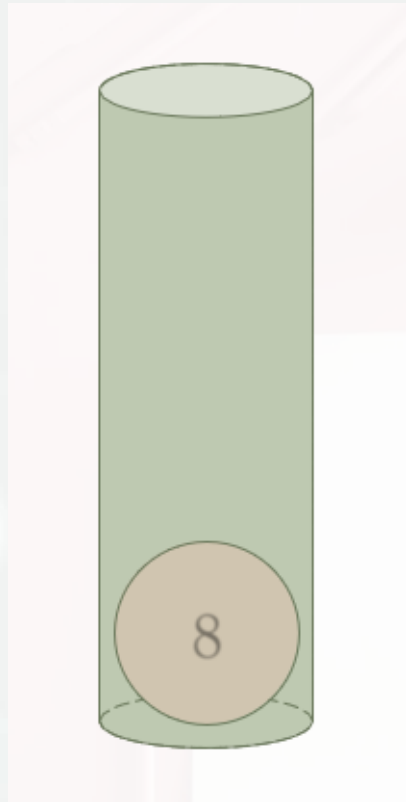
*Recordar que:
Las **precondiciones**, son condiciones que deben cumplirse antes de la ejecución de la operación.*

Pilas - Especificación



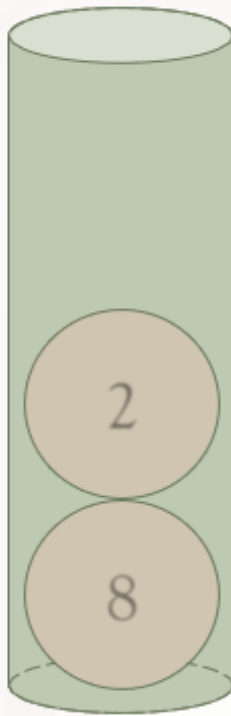
`pilaVacia () = true`

Pilas - Especificación



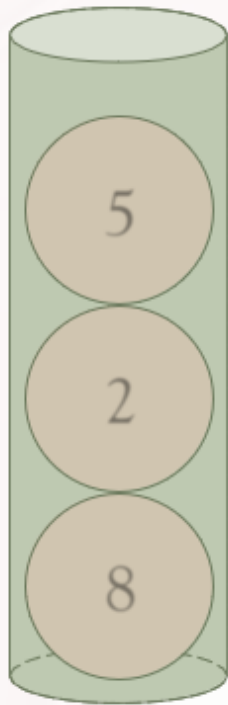
apilar (8)
pilaVacía () = false
tope () = 8

Pilas - Especificación



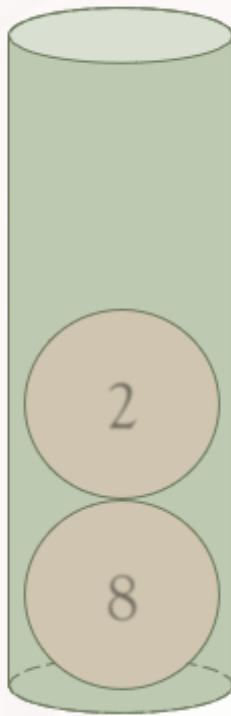
apilar (2)
pilaVacía () = false
tope () = 2

Pilas - Especificación



apilar (5)
pilaVacía () = false
tope () = 5

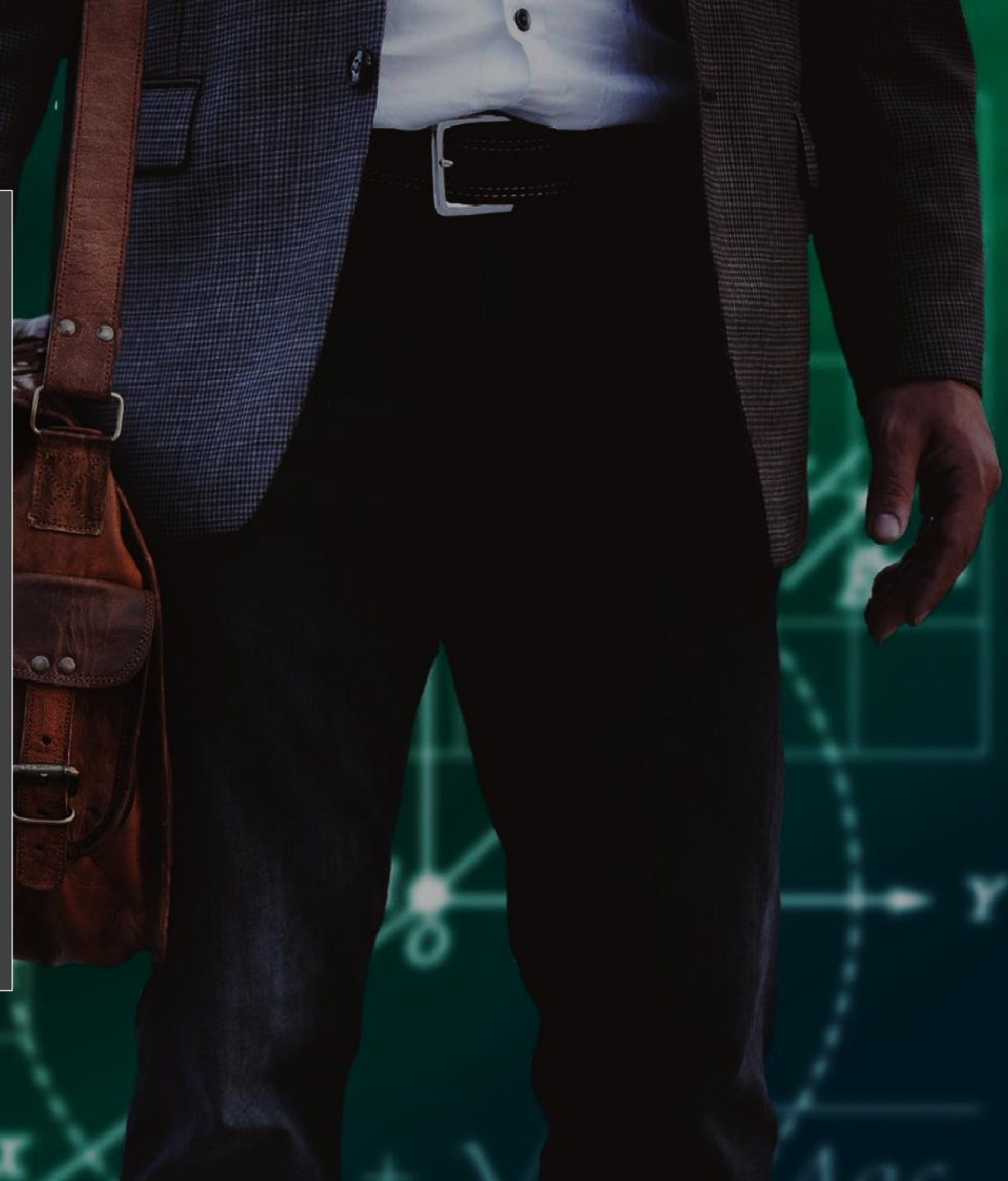
Pilas - Especificación



desapilar ()
pilaVacía () = false
tope () = 2

Especificación - Interfaz

```
public interface PilaTDA {  
    void inicializarPila( );  
    void apilar(int x); //pila inicializada  
    void desapilar( ); //pila inicializada y no vacía  
    int tope( ); //pila inicializada y no vacía  
    boolean pilaVacía( ); //pila inicializada  
}
```



A person wearing a grey checkered blazer, a white shirt, and a brown leather bag is standing in front of a green chalkboard. The chalkboard is filled with various mathematical diagrams, including a coordinate system with a vector \mathbf{a} and an angle θ , a vector \mathbf{b} , and a vector \mathbf{r} . There are also some algebraic expressions like $|a \times b|$ and γ .

Uso

Pila - Uso - Ejemplos

- *Vamos a pasar los elementos de una pila origen a una pila destino, en donde se perderán los elementos de la pila origen y en la pila destino quedarán esos elementos en orden inverso a la origen.*
- *Queremos escribir un método que nos devuelva la suma de los valores que contiene la pila. En éste caso el método va a retornar un valor del tipo int.*

The background of the slide features a person from the waist down, wearing a grey checkered blazer, a white shirt, and dark trousers with a silver belt. A brown leather messenger bag is slung over their shoulder. The entire scene is set against a dark green background with faint, glowing mathematical diagrams and equations, including vectors, angles, and geometric shapes.

Implementación

Pila - Implementación estática

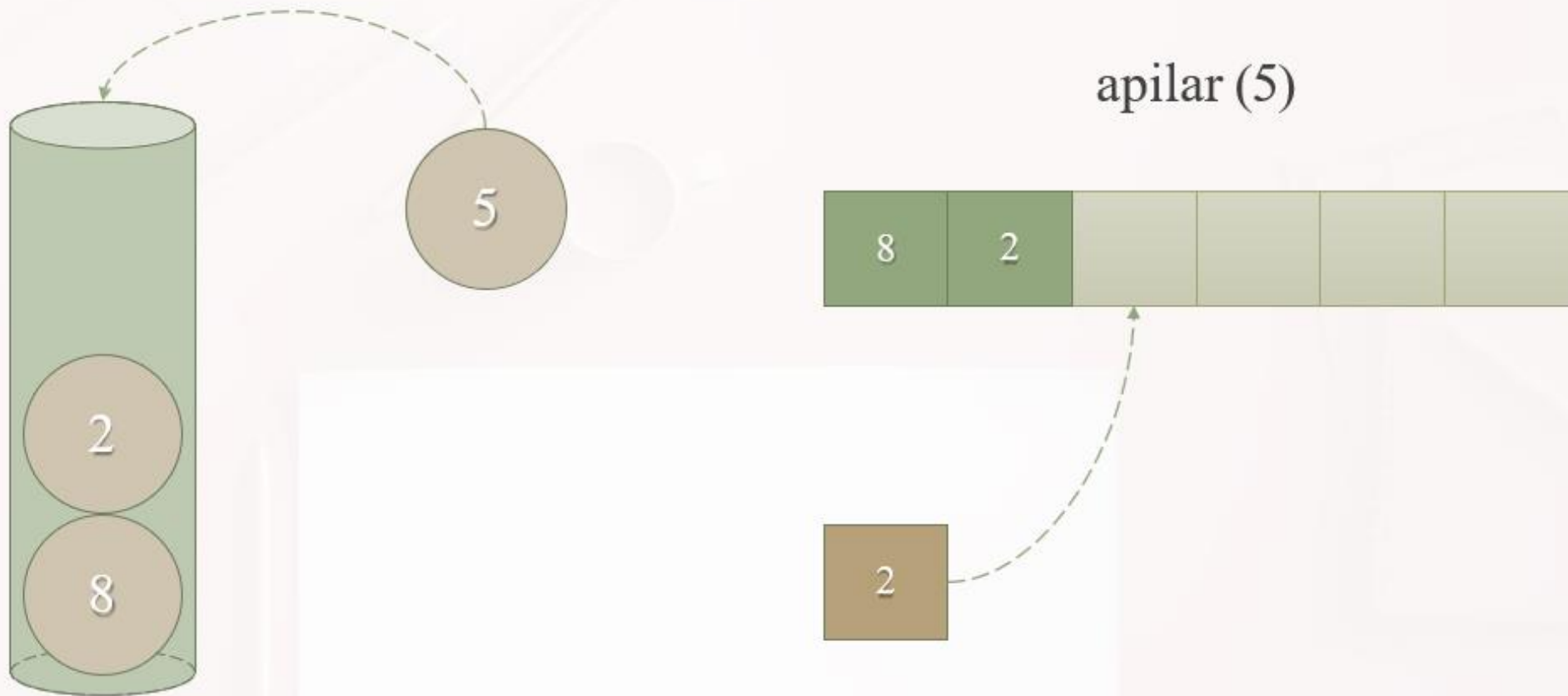
Estrategia 1

- *Se guardan los datos en un **arreglo** y se tiene una **variable** que indica la cantidad de elementos que se tienen guardados en la pila.*
- *Cuando agregamos un nuevo elemento a la pila, el mismo se **guarda** en la posición indicada por la variable que me indica la cantidad de elementos, y esta variable se incrementa en uno.*
- *Cuando se tiene que **desapilar** un elemento de la pila, se decrementa en una unidad la variable que me indica la cantidad de elementos (borrado lógico).*
- *El valor de la variable apunta a la primera posición libre del arreglo.*

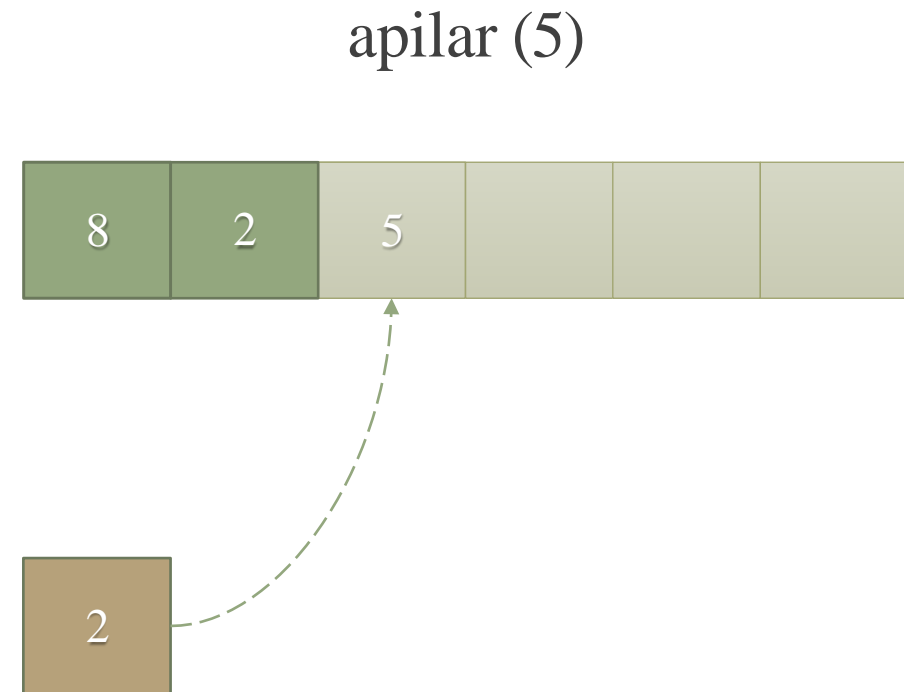
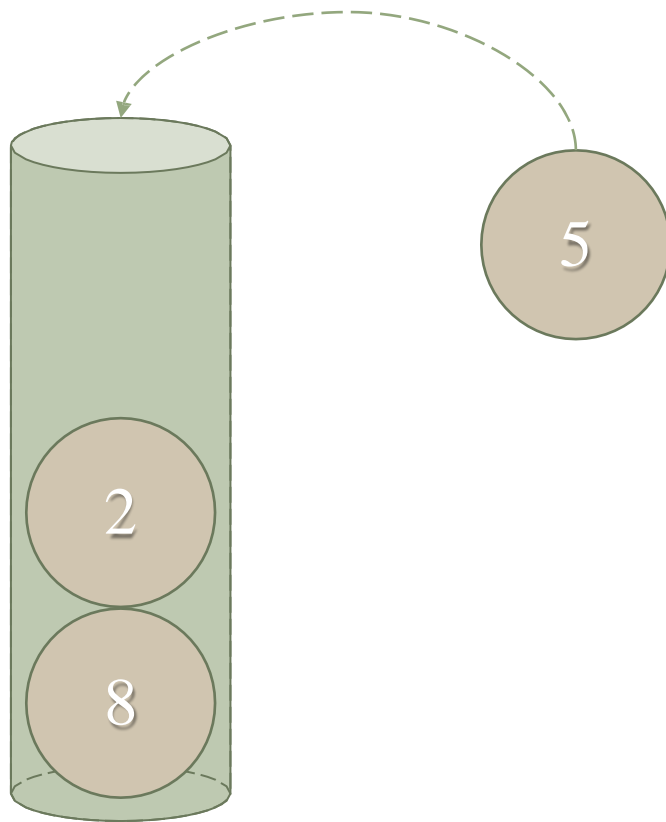
Pila - Implementación estática - Estrategia 1



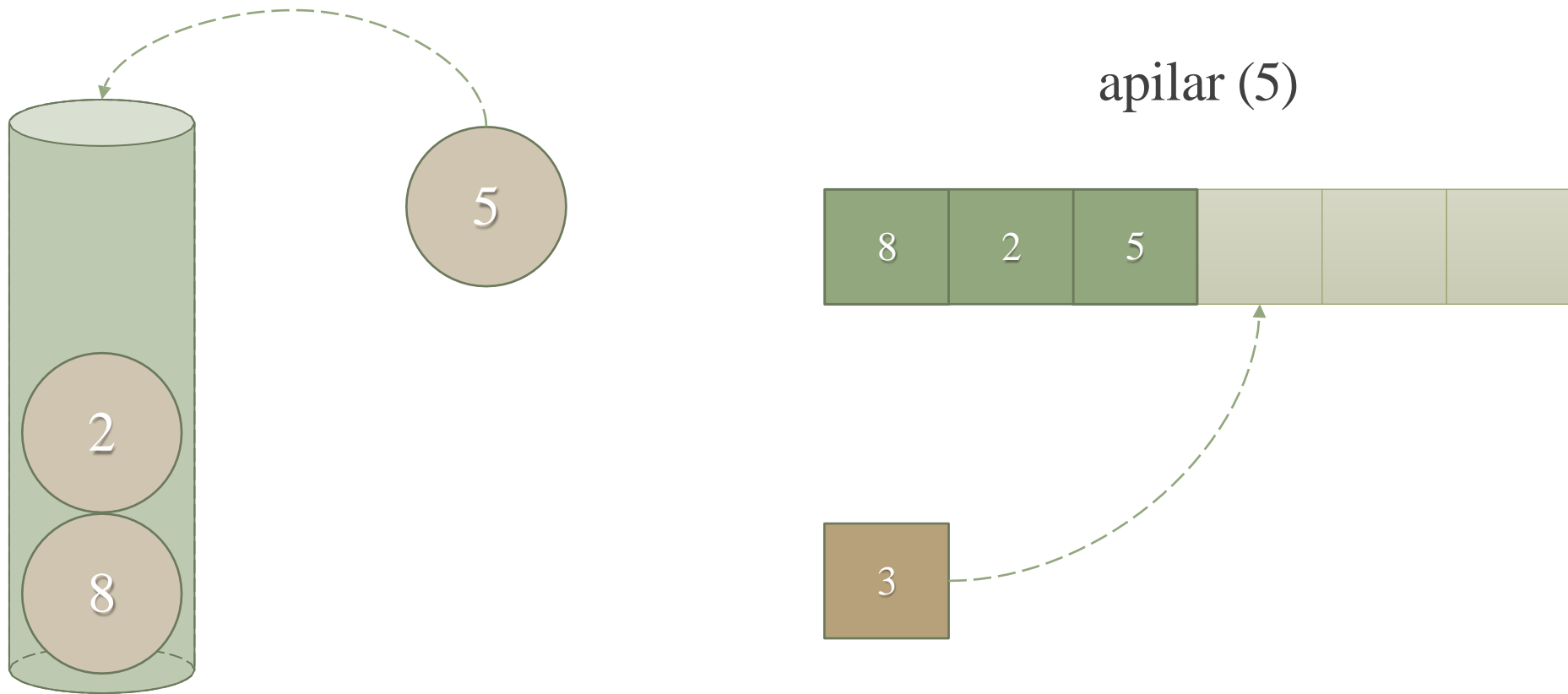
Pila - Implementación estática - Estrategia 1



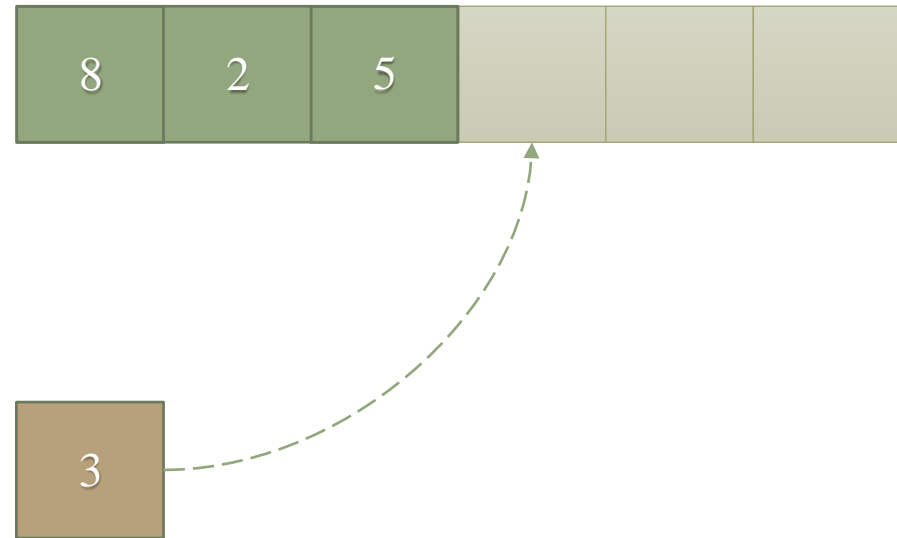
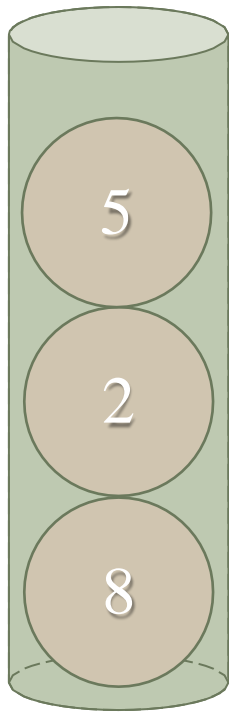
Pila - Implementación estática - Estrategia 1



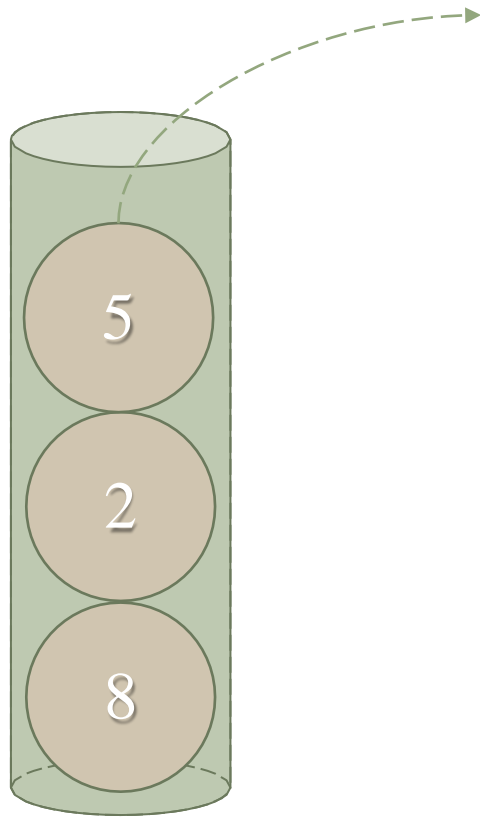
Pila - Implementación estática - Estrategia 1



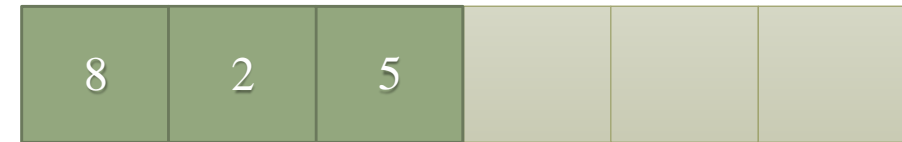
Pila - Implementación estática - Estrategia 1



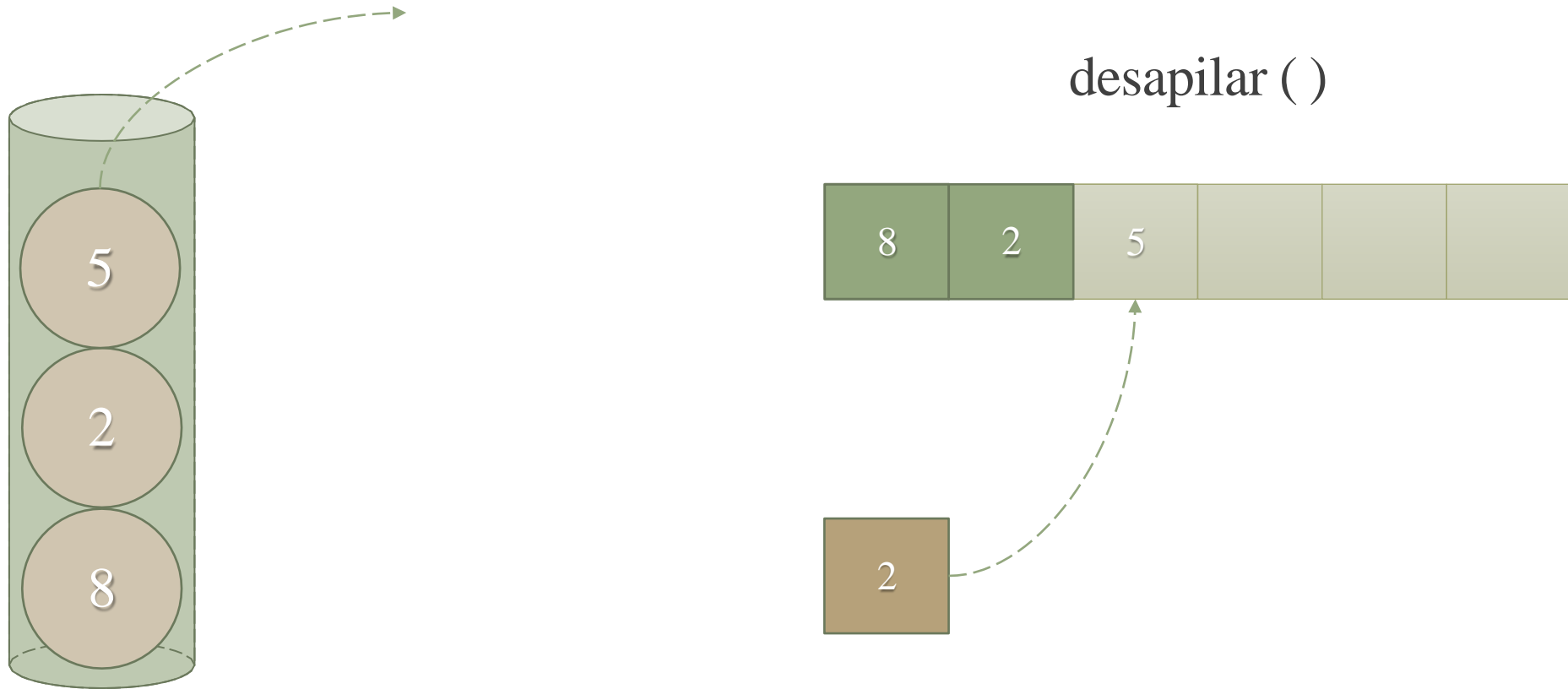
Pila - Implementación estática - Estrategia 1



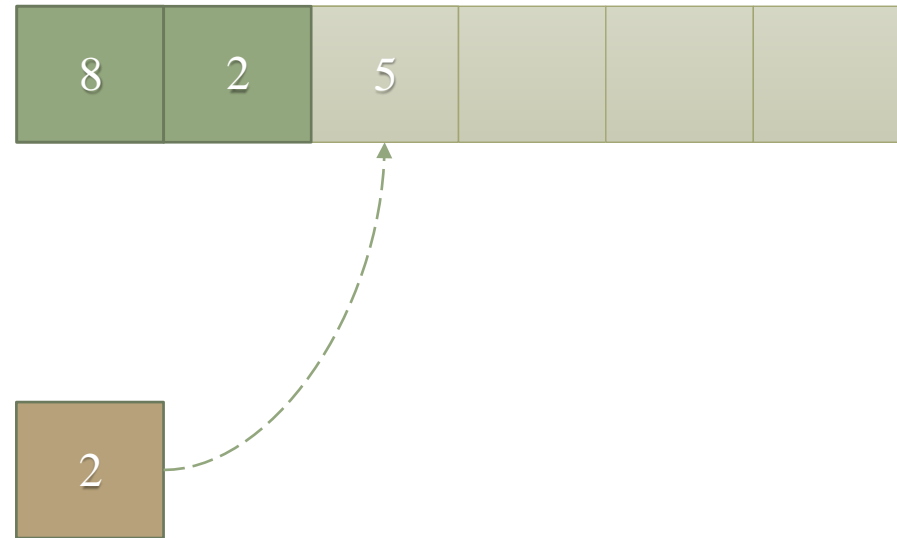
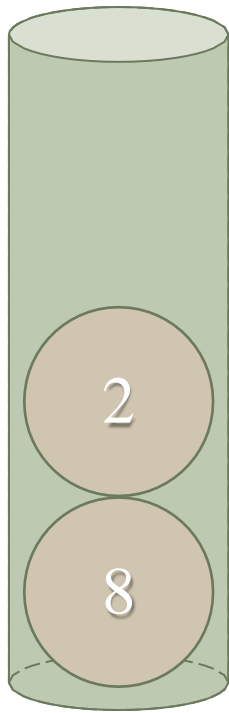
desapilar ()



Pila - Implementación estática - Estrategia 1



Pila - Implementación estática - Estrategia 1



Pila - Implementación estática

Aclaraciones

- *La **eliminación** de un elemento del vector arr se representa dejándolo afuera de la parte del arreglo delimitada por la variable indice; a los efectos prácticos, cualquier elemento arr[i] situado en una posición $i \geq \text{indice}$ **no existe más en la pila**.*
- *Tanto el vector arr, como el entero índice **no son accesibles desde afuera de la implementación** (son privados).*



Bibliografía

👑 *Programación II – Apuntes de
Cátedra – V1.3 – Cuadrado
Trutner – UADE*

👑 *Programación II – Apuntes de
Cátedra – Wehbe – UADE*