



# Estructura de Datos

Maria C. Torres

# Maria C. Torres

Ing. Electrónica (UNAL)

M.E. Ing. Eléctrica (UPRM)

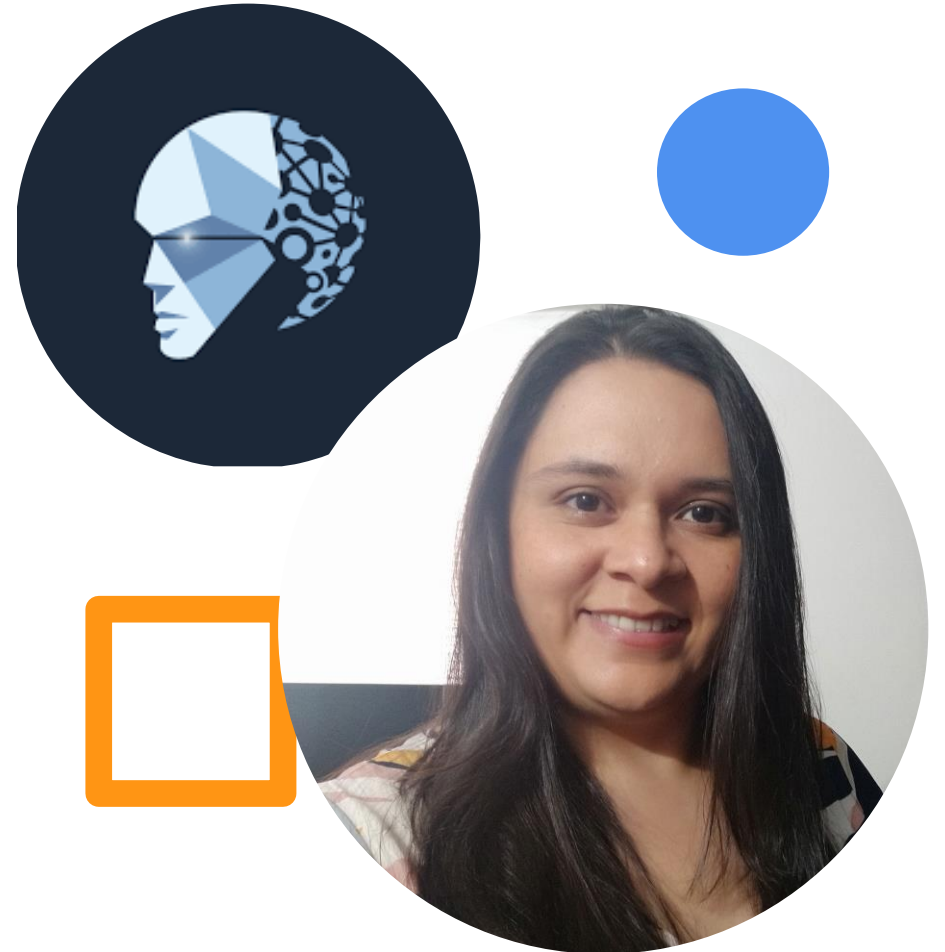
Ph.D. Ciencias e Ingeniería de la Computación y la Información (UPRM)


Profesora asociada

Dpto. Ciencias de la Computación y la Decisión


[mctorresm@unal.edu.co](mailto:mctorresm@unal.edu.co)

HORARIO DE ATENCIÓN: Martes 10:00 am a 12:00 m – Oficina 313 M8A





# Contenido del Curso

- 
- ☐ Introducción: revisión fundamentos y POO
  - ☐ Análisis de complejidad
  - ☐ Arreglos
  - ☐ Listas enlazadas
  - ☐ **Pilas y colas**
  - ☐ Heap
  - ☐ Árboles binarios
  - ☐ Tablas hash
  - ☐ Grafos



# Pilas y colas

- ❑ Pilas - Stacks
  - ❑ Implementación usando arreglos y lista simple
- ❑ Colas - Queue
  - ❑ Implementación usando arreglos y lista simple
- ❑ Análisis de complejidad operaciones
- ❑ Aplicaciones y algoritmos

# PILA - STACK

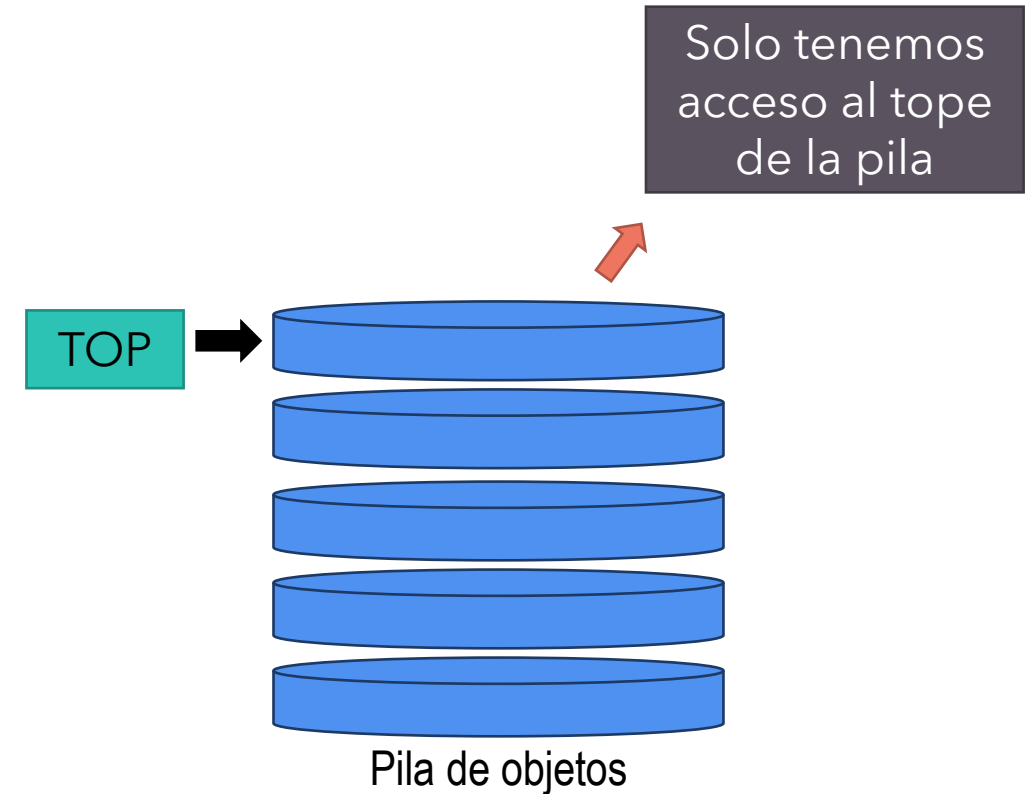
- ❑ **Acceso secuencial - limitado:** solo podemos acceder a un elemento de la estructura
- ❑ Una pila o STACK es una colección de objetos que son insertados o eliminados de acuerdo con el principio "**ultimo en entrar - primero en salir**"
- ❑ En ingles este principio se denomina **LIFO** (last-in first out), es decir, el ultimo objeto insertado en la pila es el único elemento al cual se tiene acceso



# PILA - STACK

Una pila o STACK tiene dos atributos

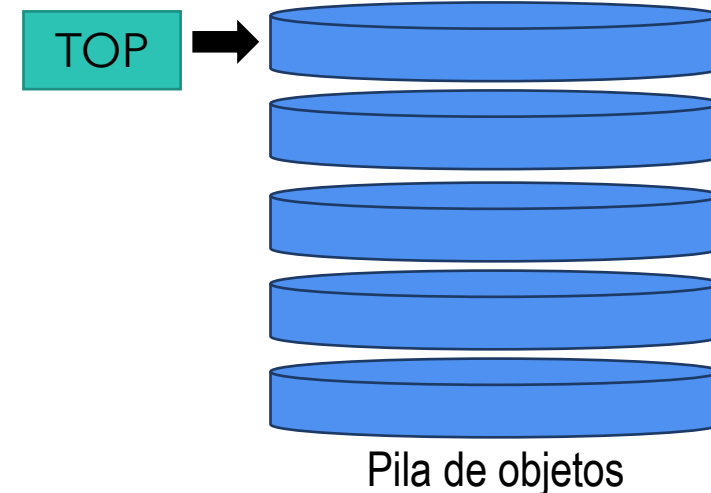
- ❑ TOP: objeto en el tope de la pila, es el único al que se tiene acceso y se puede eliminar
- ❑ SIZE: número de objetos almacenados en la pila



# PILA - STACK

Una pila o STACK solo se permite dos operaciones

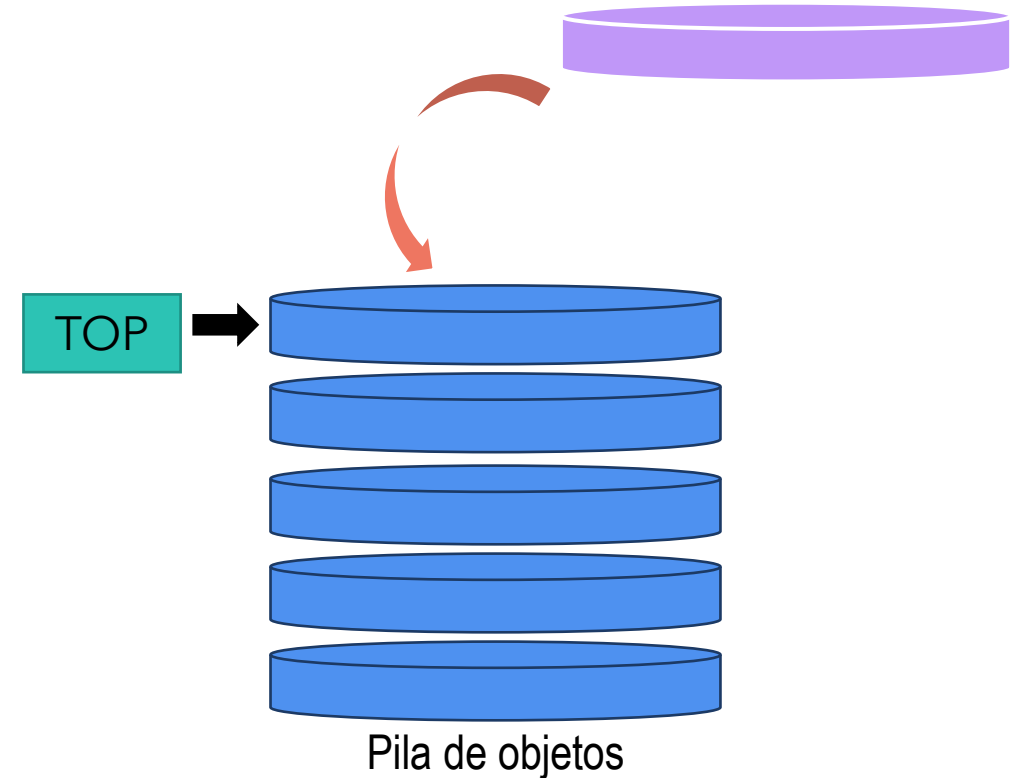
- ❑ PUSH: operación de empujar o colocar un nuevo objeto en la pila, este objeto siempre se agrega al tope de la pila
- ❑ POP: operación de sacar o remover el objeto en el tope de la pila



# PILA - STACK

Una pila o STACK solo se permite dos operaciones

- ❑ **PUSH:** operación de empujar o colocar un nuevo objeto en la pila, este objeto siempre se agrega al tope de la pila
- ❑ **POP:** operación de sacar o remover el objeto en el tope de la pila

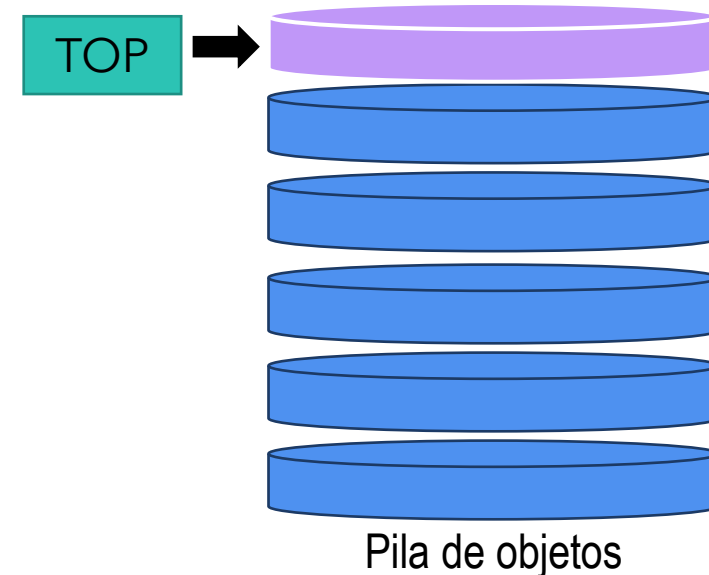




# PILA - STACK

Una pila o STACK solo se permite dos operaciones

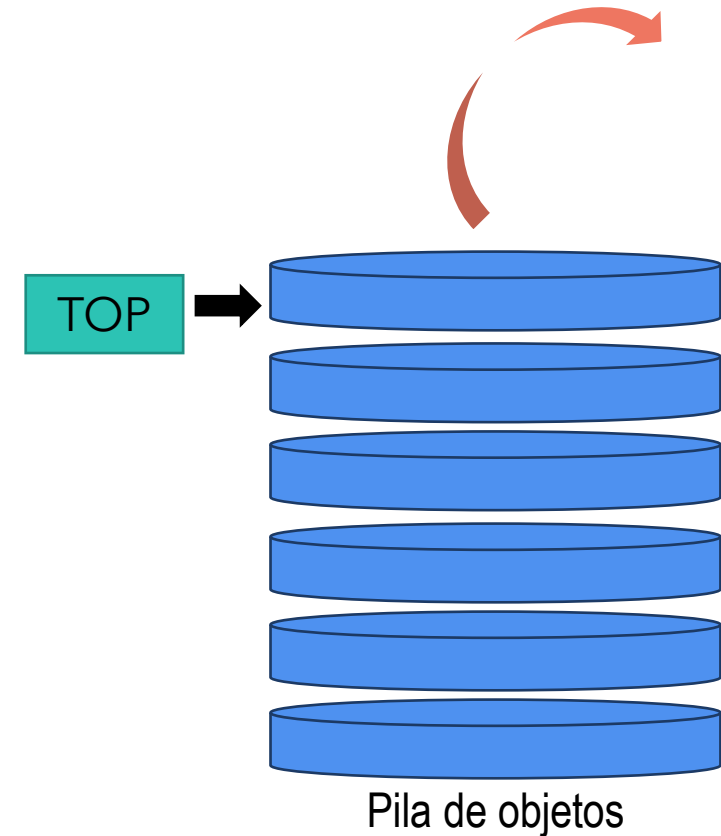
- ❑ **PUSH:** operación de empujar o colocar un nuevo objeto en la pila, este objeto siempre se agrega al tope de la pila
- ❑ **POP:** operación de sacar o remover el objeto en el tope de la pila



# PILA - STACK

Una pila o STACK solo se permite dos operaciones

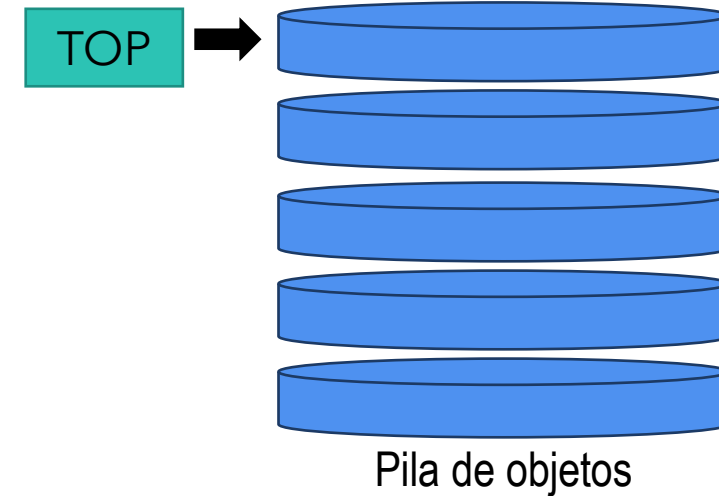
- ❑ **PUSH:** operación de empujar o colocar un nuevo objeto en la pila, este objeto siempre se agrega al tope de la pila
- ❑ **POP:** operación de sacar o remover el objeto en el tope de la pila



# PILA - STACK






Una pila o STACK solo se permite dos operaciones

- ❑ PUSH: operación de empujar o colocar un nuevo objeto en la pila, este objeto siempre se agrega al tope de la pila
- ❑ **POP:** operación de sacar o remover el objeto en el tope de la pila

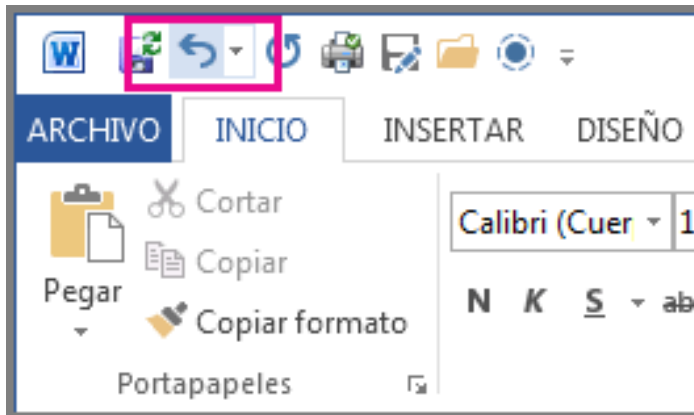


# Aplicaciones

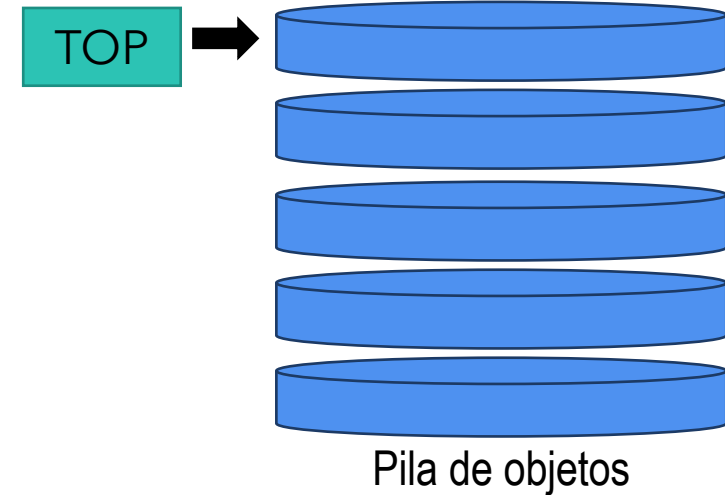
- Historial de pagina web

|                          |         |   |  |                           |
|--------------------------|---------|---|--|---------------------------|
| <input type="checkbox"/> | 3:05 PM |  | PowerPoint                                 | www.office.com            |
| <input type="checkbox"/> | 3:05 PM |  | Working...                                 | www.office.com            |
| <input type="checkbox"/> | 3:05 PM |  | Working...                                 | login.microsoftonline.com |
| <input type="checkbox"/> | 3:04 PM |  | WhatsApp                                   | web.whatsapp.com          |
| <input type="checkbox"/> | 3:03 PM |  | Calendario: Maria Constanza Torres Madr... | outlook.office.com        |

- Operación deshacer en editores de texto



# PILA - STACK



- Balance de paréntesis en expresiones matemáticas

$$\begin{aligned} & (9+(2-1))+6 \\ &= (9+1)+6 \\ &= 10+6 \\ &= 16 \end{aligned}$$



# Pilas y colas

- ❑ Pilas – Stacks
  - ❑ **Implementación usando arreglos y lista simple**
- ❑ Colas – Queue
  - ❑ Implementación usando arreglos y lista simple
- ❑ Análisis de complejidad operaciones
- ❑ Aplicaciones y algoritmos

# IMPLEMENTACIÓN DE PILAS

- ❑ Estudiaremos una primera implementación STACK usando arreglos
- ❑ Esta implementación se recomienda usar cuando requerimos un STACK con un número limitado y pequeño de datos

## Clase ArrayStack

Incluye los siguientes atributos:

- Un arreglo con un tamaño por defecto que almacena los datos de la pila
- TOP: correspondiente a un índice del arreglo, este índice indica donde se encuentra el tope de la pila
  - El atributo TOP inicia en -1, y puede tomar valores hasta la longitud del vector-1

### ArrayStack

- data[ ]: Object
- top: int

# IMPLEMENTACIÓN DE PILAS

## Clase ArrayStack

Los métodos que se incluyen son:

- Constructor: recibe la capacidad del arreglo
- size: retorna el número de objetos en el stack
- isEmpty(): retorna TRUE si el stack está vacío

### ArrayStack

- data[ ]: Object  
- top: int

+ ArrayStack(int capacity)  
+ size(): int  
+ isEmpty(): Boolean

# IMPLEMENTACIÓN DE PILAS

## Clase ArrayStack

Los métodos que se incluyen son:

- push: inserta un nuevo elemento en la pila empleando el principio LIFO
- pop: extrae (retorna y elimina) el elemento en el tope de la pila
- top: retorna (sin eliminar) el elemento en el tope de la pila

### ArrayStack

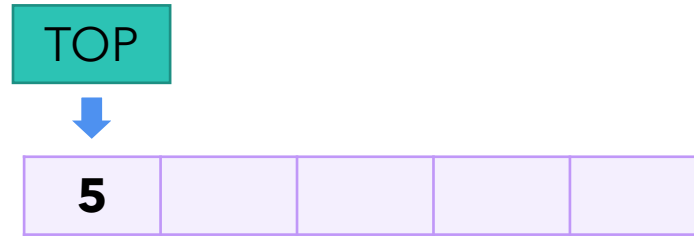
- data[ ]: Object  
- top: int

+ ArrayStack(int capacity)  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object



# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



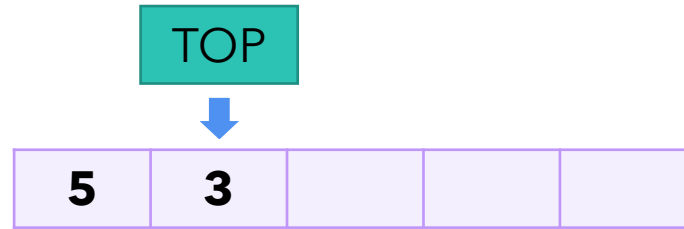
>> push(5)



Pila de objetos

# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



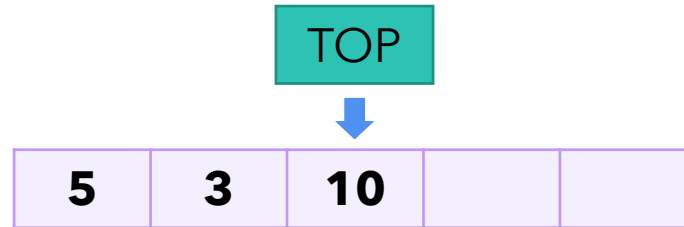
```
>> push(5)  
>> push(3)
```



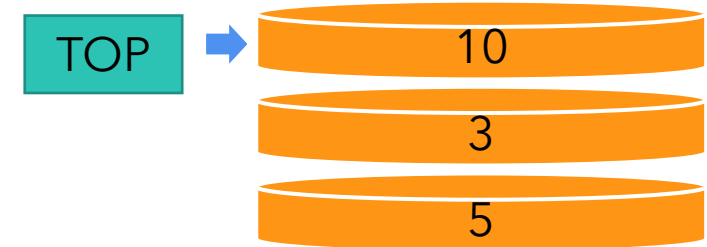
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



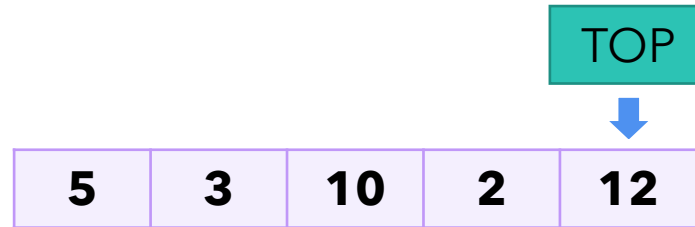
```
>> push(5)  
>> push(3)  
>> push(10)
```



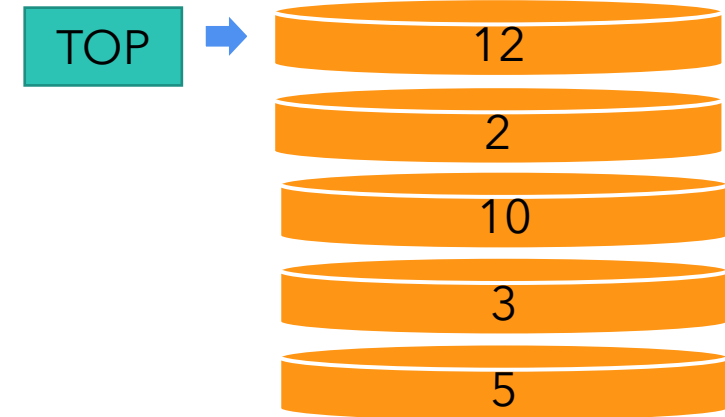
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



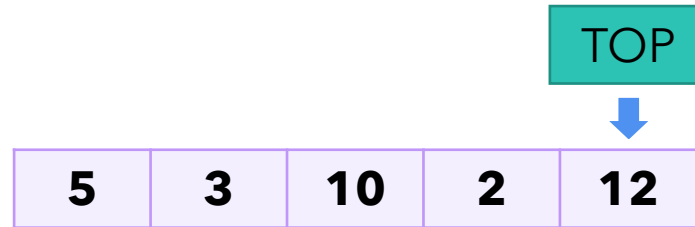
```
>> push(5)
>> push(3)
>> push(10)
>> push(2)
>> push(12)
```



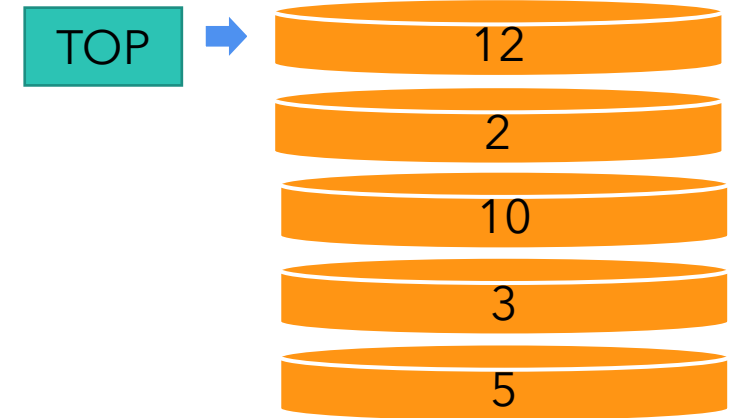
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



```
>> push(5)
>> push(3)
>> push(10)
>> push(2)
>> push(12)
>> push(1) !!Stack overflow
```

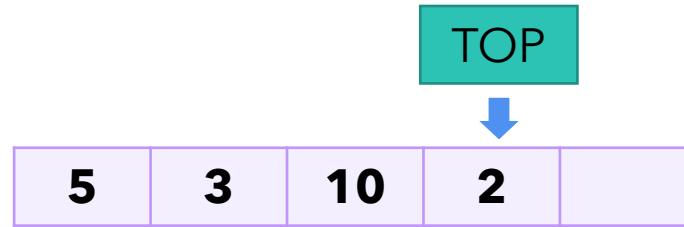


Pila de objetos

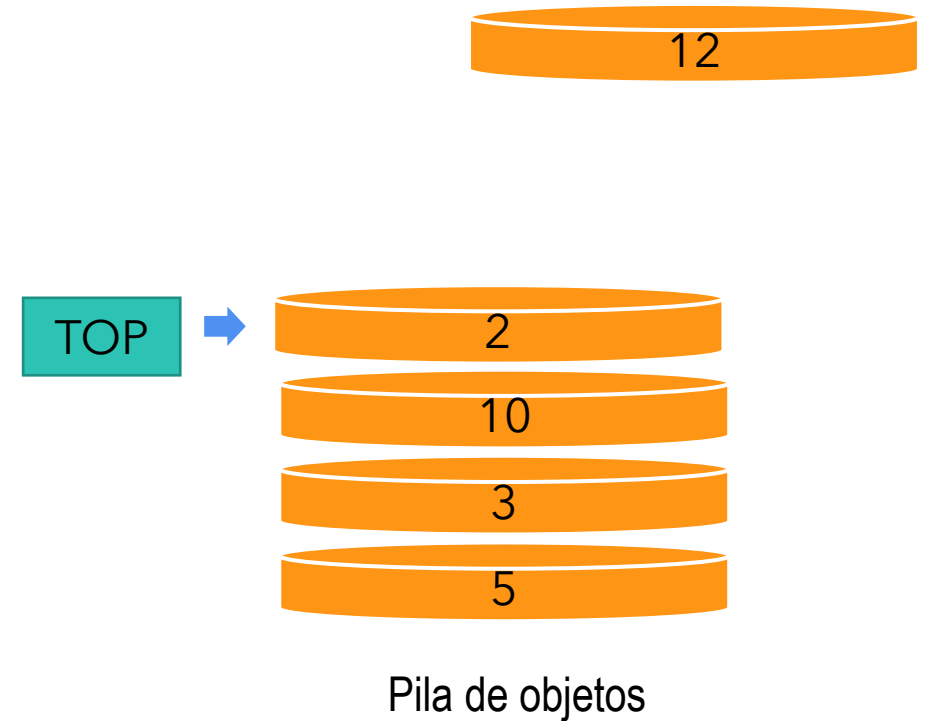
Recuerda: la capacidad (capacity) determina el tamaño máximo de la pila

# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |

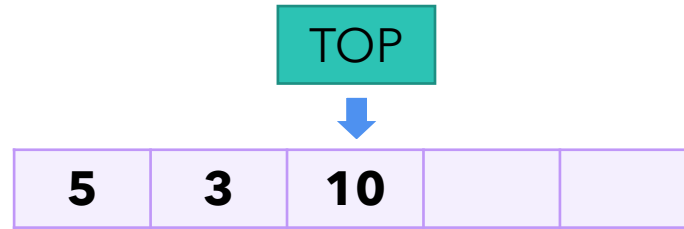


>> pop() >> 12

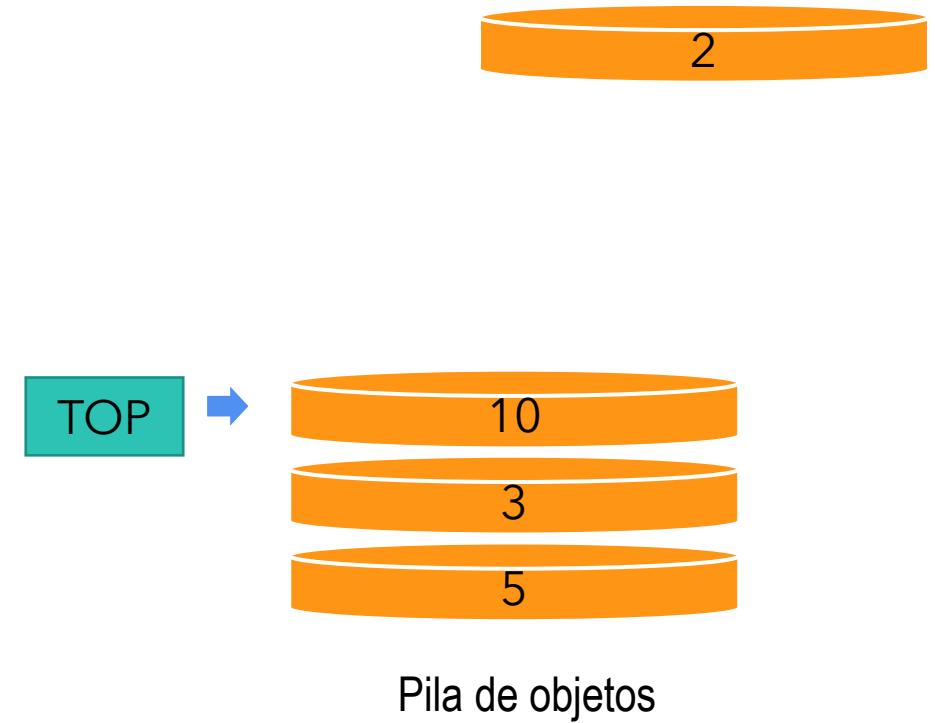


# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |

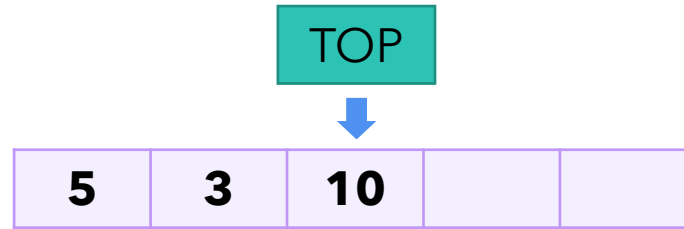


```
>> pop() >> 12  
>> pop() >> 2
```

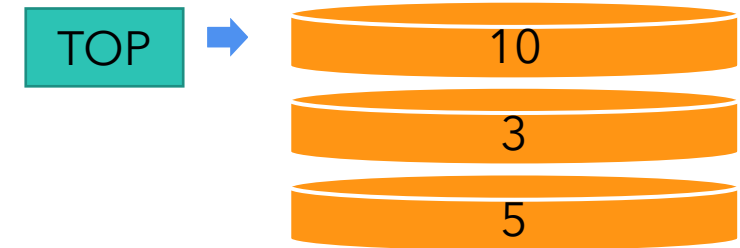


# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



```
>> pop() >> 12  
>> pop() >> 2  
>> top() >> 10
```

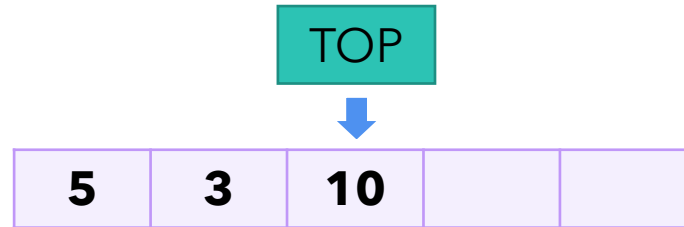


Pila de objetos



# IMPLEMENTACIÓN DE PILAS

| ArrayStack  |
|---|
| - data[ ]: Object<br>- top: int   |
| + ArrayStack(int capacity)<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



```
>> pop() >> 12  
>> pop() >> 2  
>> top() >> 10
```



Pila de objetos

El número de datos en la pila es igual  
a la posición del top + 1

# IMPLEMENTACIÓN DE PILAS

```
ArrayStack(int capacity)
    data = new Object[capacity]
    top = -1
```

```
size()
    return top+1
```

```
isEmpty()
    return size()==0
```

## ArrayStack

- data[ ]: Object
- top: int

- + ArrayStack(int capacity)
- + size(): int
- + isEmpty(): Boolean
- + push(Object e)
- + pop(): Object
- + top(): Object

# IMPLEMENTACIÓN DE PILAS

```
push(Object e)
    if top < data.length-1
        top++
        data[top]=e
    else
        print("Stack overflow")
```

## ArrayStack

- data[ ]: Object
- top: int
- + ArrayStack(int capacity)
- + size(): int
- + isEmpty(): Boolean
- + push(Object e)
- + pop(): Object
- + top(): Object

# IMPLEMENTACIÓN DE PILAS

```
pop()
    if !isEmpty()
        temp = data[top]
        data[top] = null
        top--
        return temp
    else
        return null
```

## ArrayStack

- data[ ]: Object
- top: int
- + ArrayStack(int capacity)
- + size(): int
- + isEmpty(): Boolean
- + push(Object e)
- + pop(): Object
- + top(): Object

# IMPLEMENTACIÓN DE PILAS

```
top()
  if !isEmpty()
    return data[top]
  else
    return null
```

## ArrayStack

- data[ ]: Object  
- top: int

+ ArrayStack(int capacity)  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object

# IMPLEMENTACIÓN DE PILAS

- ❑ En la segunda implementación del STACK usaremos listas simples
- ❑ Esta implementación se recomienda usar cuando requerimos un STACK que use memoria dinámica

## Clase Stack

Se mantienen los siguientes atributos

- Una lista simple
- No se requiere el atributo TOP, dado que la cabecera de la lista corresponderá a este elemento

| Stack        |
|--------------|
| - data: List |
|              |

# IMPLEMENTACIÓN DE PILAS

Recuerda las clases de nodo simple y lista simple!!!

| Node   |
|--|
| -data:Object<br>-next:Node   |
| - Node ()<br>- Node (Object e)<br>- getData(): Object<br>- getNext(): Node<br>- setData(Object e)<br>- setNext(Node n) |

| List   |
|--|
| -head: Node<br>-tail: Node<br>-size: int   |
| +List()<br>+size(): int<br>+isEmpty(): Boolean<br>+setSize(int s)<br>+First():Node<br>+Last():Node<br>+addFirst(Object e)<br>+addLast(Object e)<br>+removeFirst():Object<br>+removeLast():Object |

| Stack        |
|--------------|
| - data: List |
|              |

# IMPLEMENTACIÓN DE PILAS

## Clase Stack

Los métodos incluyen:

- El constructor que inicializa la lista vacía
- Los métodos size, isEmpty, push, pop y top que tienen el mismo comportamiento de la implementación con arreglos

### Stack

- data: List

+ Stack()  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object

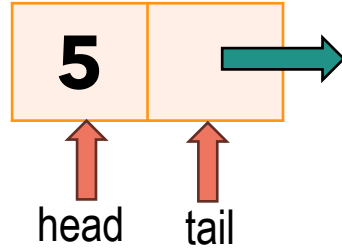


# IMPLEMENTACIÓN DE PILAS

## Stack

- data: List

+ Stack()  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object



>> push(5)



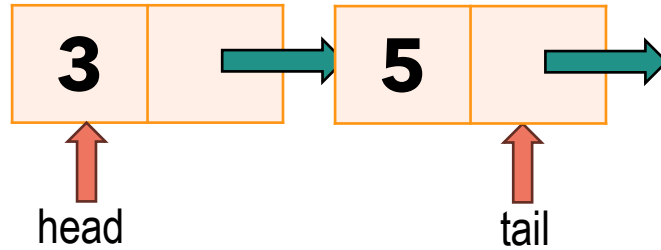
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

## Stack

- data: List

+ Stack()  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object



```
>> push(5)  
>> push(3)
```



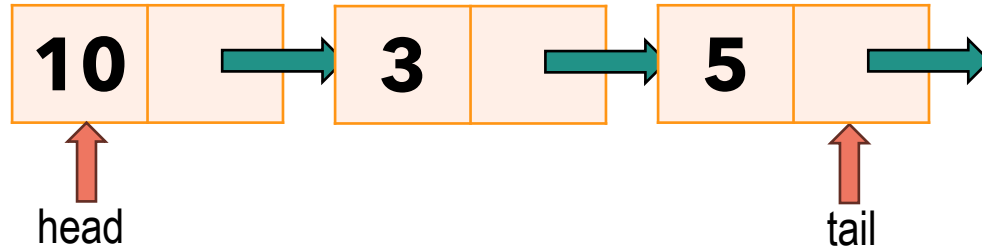
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

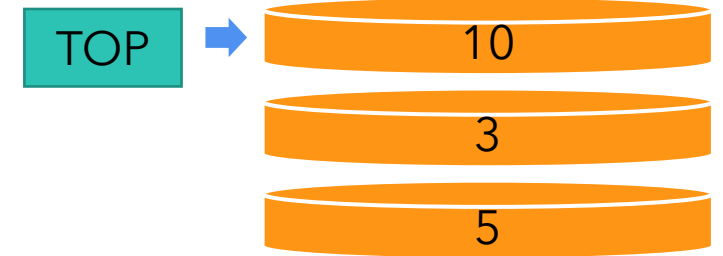
## Stack

- data: List

+ Stack()  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object



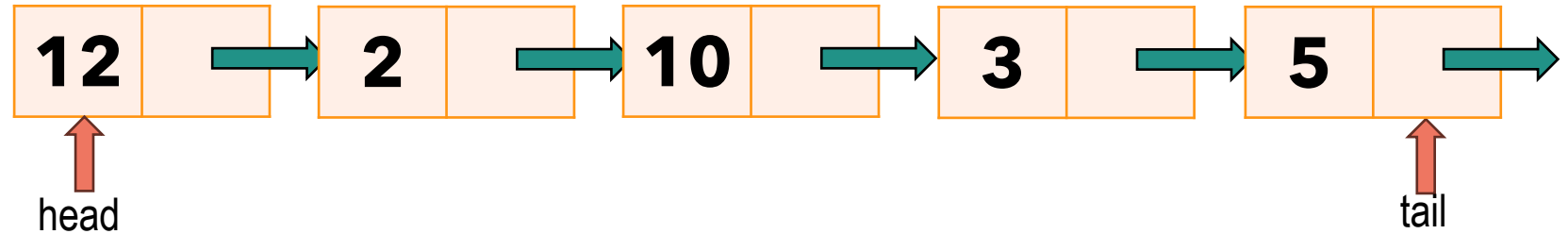
```
>> push(5)  
>> push(3)  
>> push(10)
```



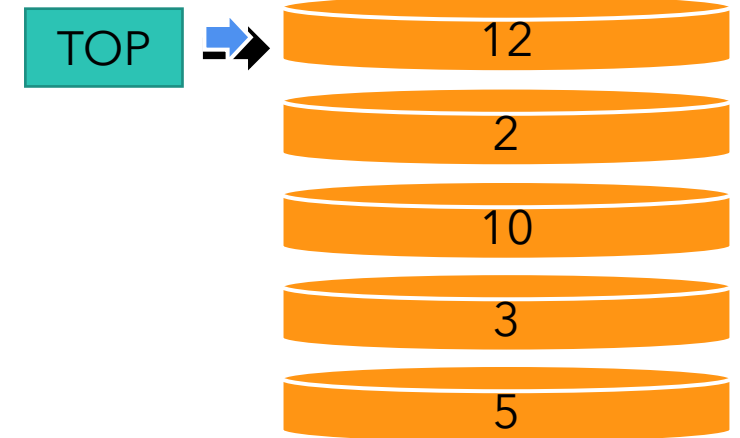
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

| Stack  |
|--|
| - data: List   |
| + Stack()<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



```
>> push(5)  
>> push(3)  
>> push(10)  
>> push(2)  
>> push(12)
```

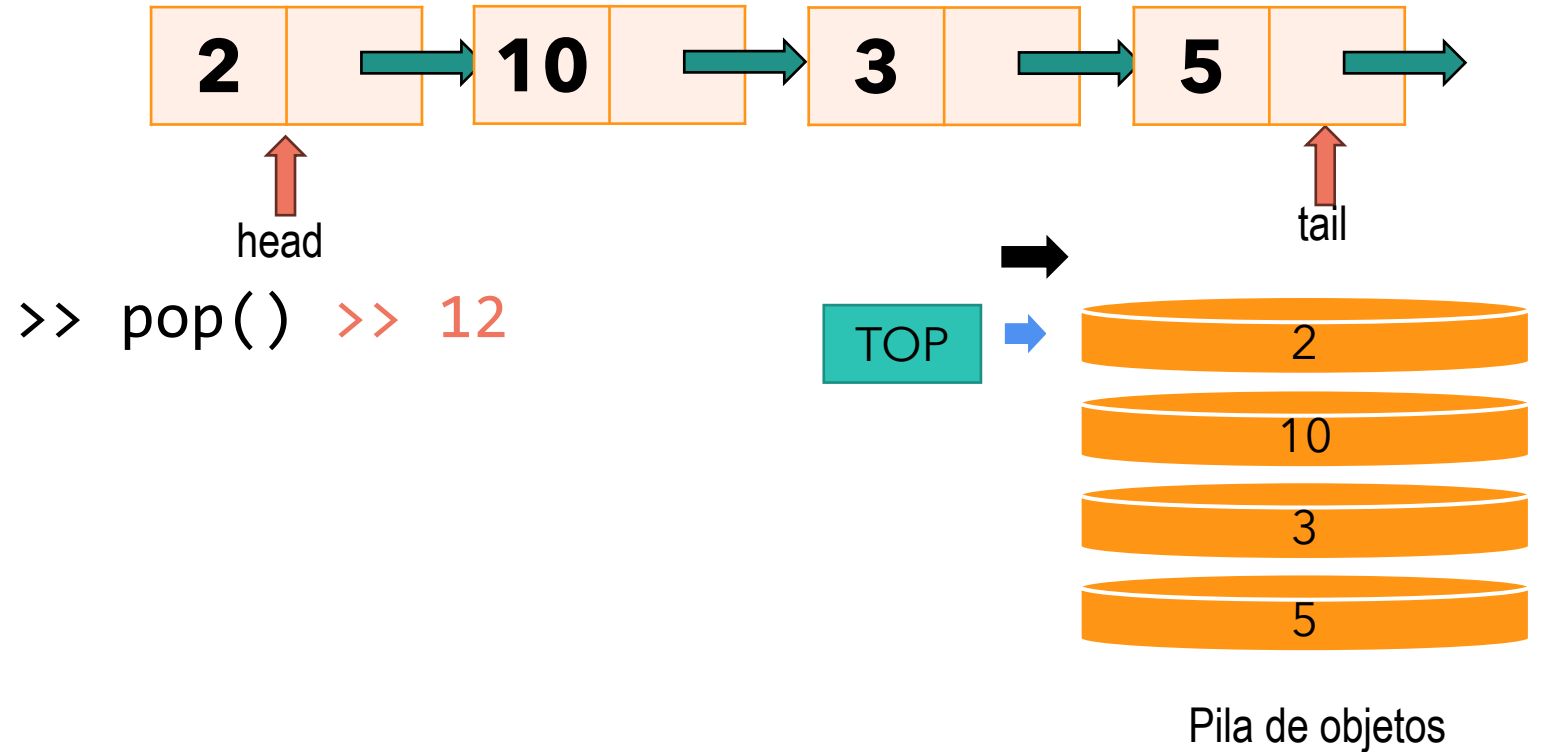


Pila de objetos

La implementación con la lista simple  
no tiene una capacidad limitada!!!

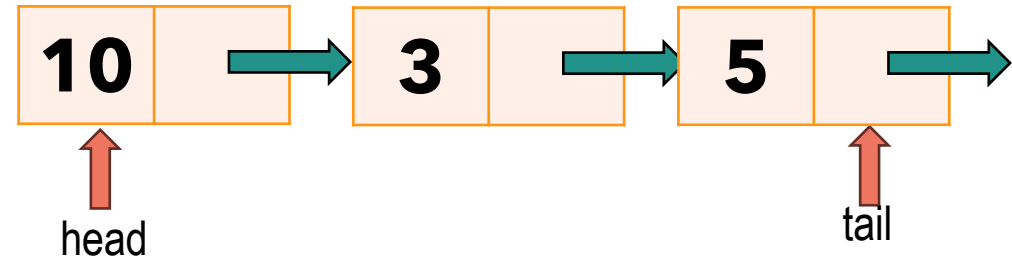
# IMPLEMENTACIÓN DE PILAS

| Stack  |
|--|
| - data: List   |
| + Stack()<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |

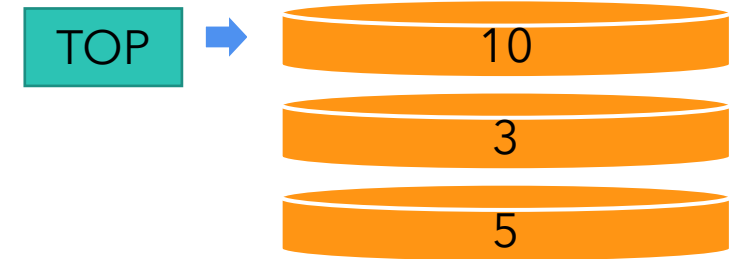


# IMPLEMENTACIÓN DE PILAS

| Stack  |
|--|
| - data: List   |
| + Stack()<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



```
>> pop() >> 12  
>> pop() >> 2
```



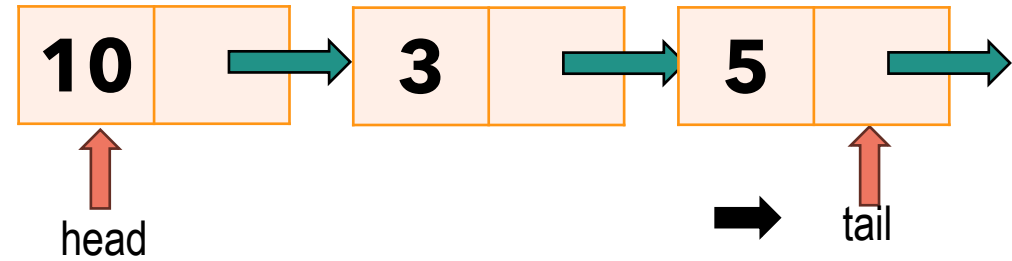
Pila de objetos

# IMPLEMENTACIÓN DE PILAS

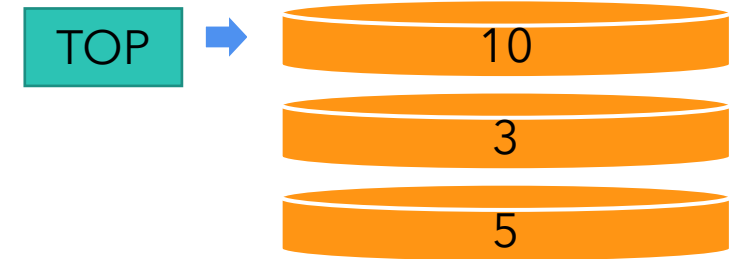
## Stack

- data: List

+ Stack()  
+ size(): int  
+ isEmpty(): Boolean  
+ push(Object e)  
+ pop(): Object  
+ top(): Object



```
>> pop() >> 12  
>> pop() >> 2  
>> top() >> 10
```



Pila de objetos

# IMPLEMENTACIÓN DE PILAS

```
Stack()  
    data = new List()  
  
size()  
    return data.size()  
  
isEmpty()  
    return size()==0
```

| Stack  |
|--|
| - data: List   |
| + Stack()<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |



# IMPLEMENTACIÓN DE PILAS

```
push(Object e)
    data.addFirst(e)

pop()
    return data.removeFirst()

top()
    if !isEmpty()
        return data.first().getData()
    else
        return null
```

| Stack  |
|--|
| - data: List   |
| + Stack()<br>+ size(): int<br>+ isEmpty(): Boolean<br>+ push(Object e)<br>+ pop(): Object<br>+ top(): Object |

| List   |
|--|
| -head: Node<br>-tail: Node<br>-size: int   |
| +List()<br>+size(): int<br>+isEmpty(): Boolean<br>+setSize(int s)<br>+First():Node<br>+Last():Node<br>+addFirst(Object e)<br>+addLast(Object e)<br>+removeFirst():Object<br>+removeLast():Object |

