

D{VZ



COMMUNITY

SESIÓN 3

# Aprendiendo Pilas y Colas

D{VZ

Pilas

# ¿Cómo funciona una Pila?

D{VZ

### Definición

Estructura de datos en donde el primer valor insertado es el último en salir.

(LIFO - Last In First Out)



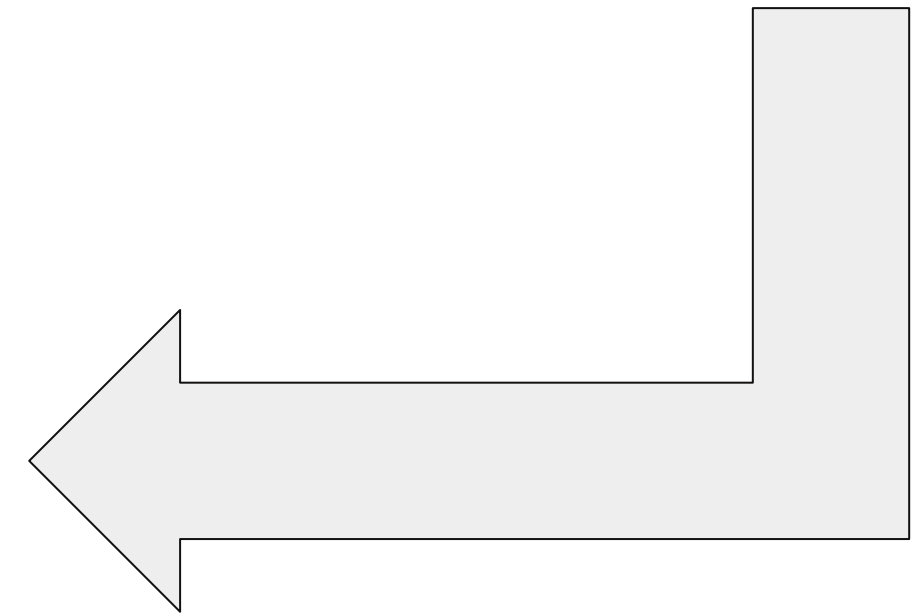


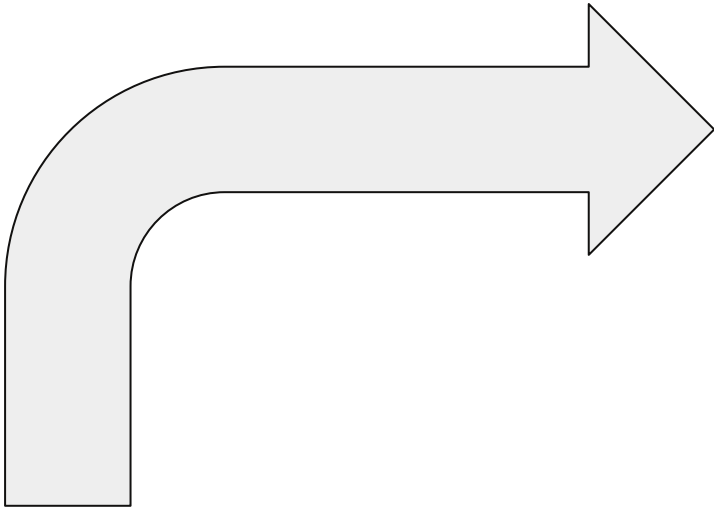




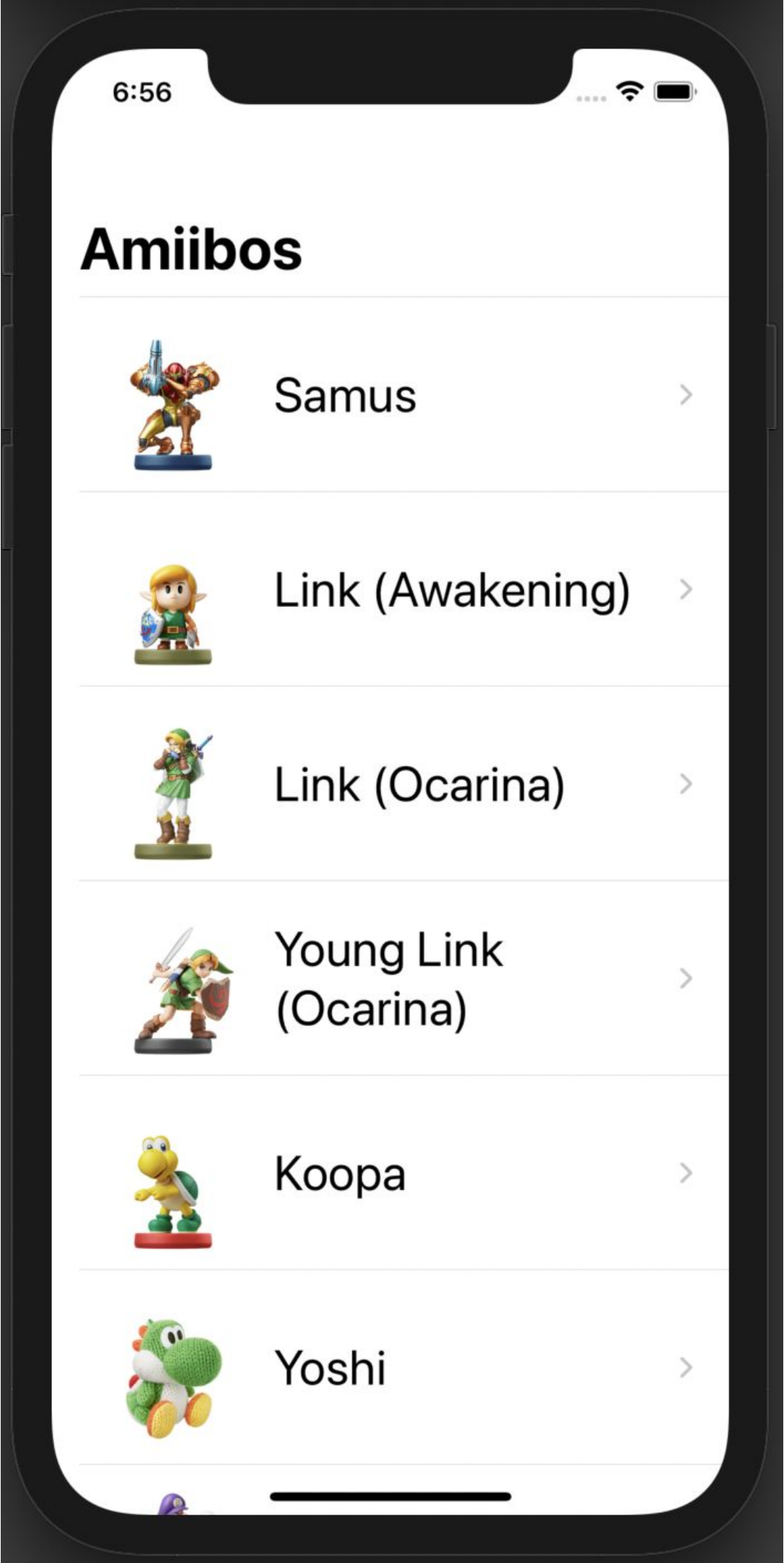


Gracias a esta “pila”  
te graduaste :)





iOS App





ViewDemo PID 6991

CPU 47%

Memory 74.2 MB

Disk Zero KB/s

Network Zero KB/s

UIWindowScene - (Foregrou...

UIWindow

UITransitionView

UIDropShadowView

Hosting View Contr...

Hosting view

AmiiboList

NavigationVi...

FeatureCo...

Column...

Colu...

\_C...

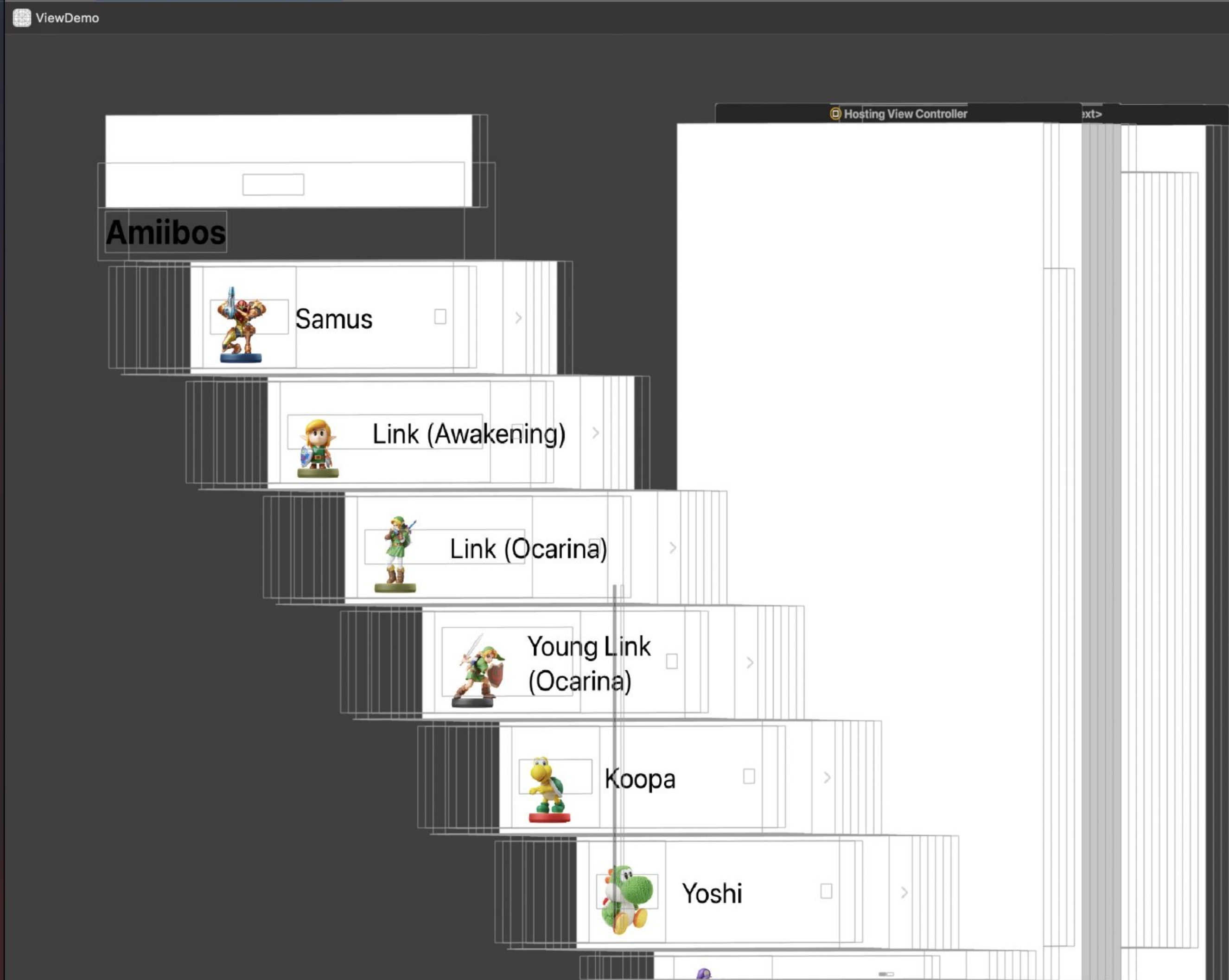
...

View Host

NotifyingMul...

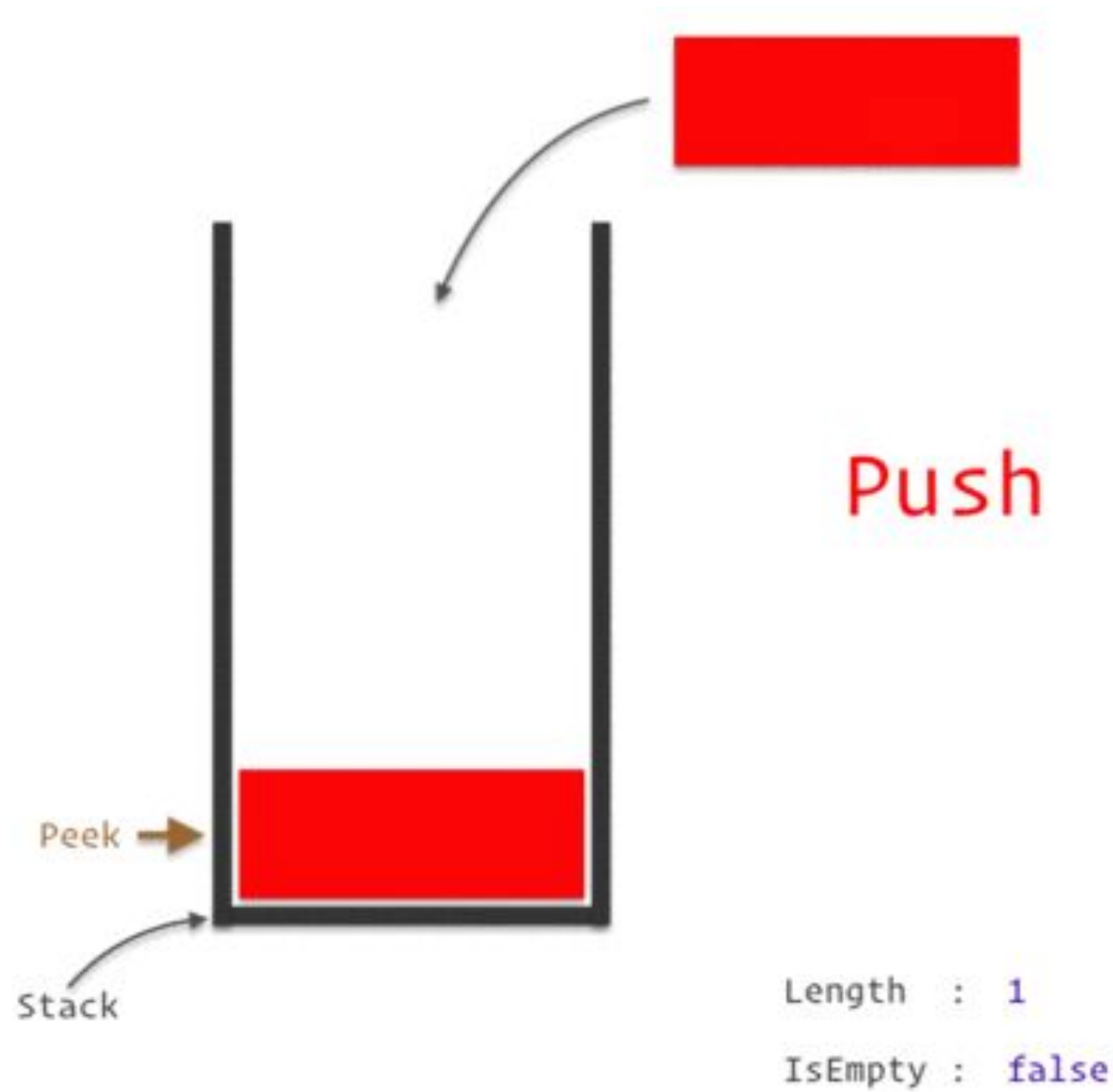
\_UISplitVi...

\_UIPan...



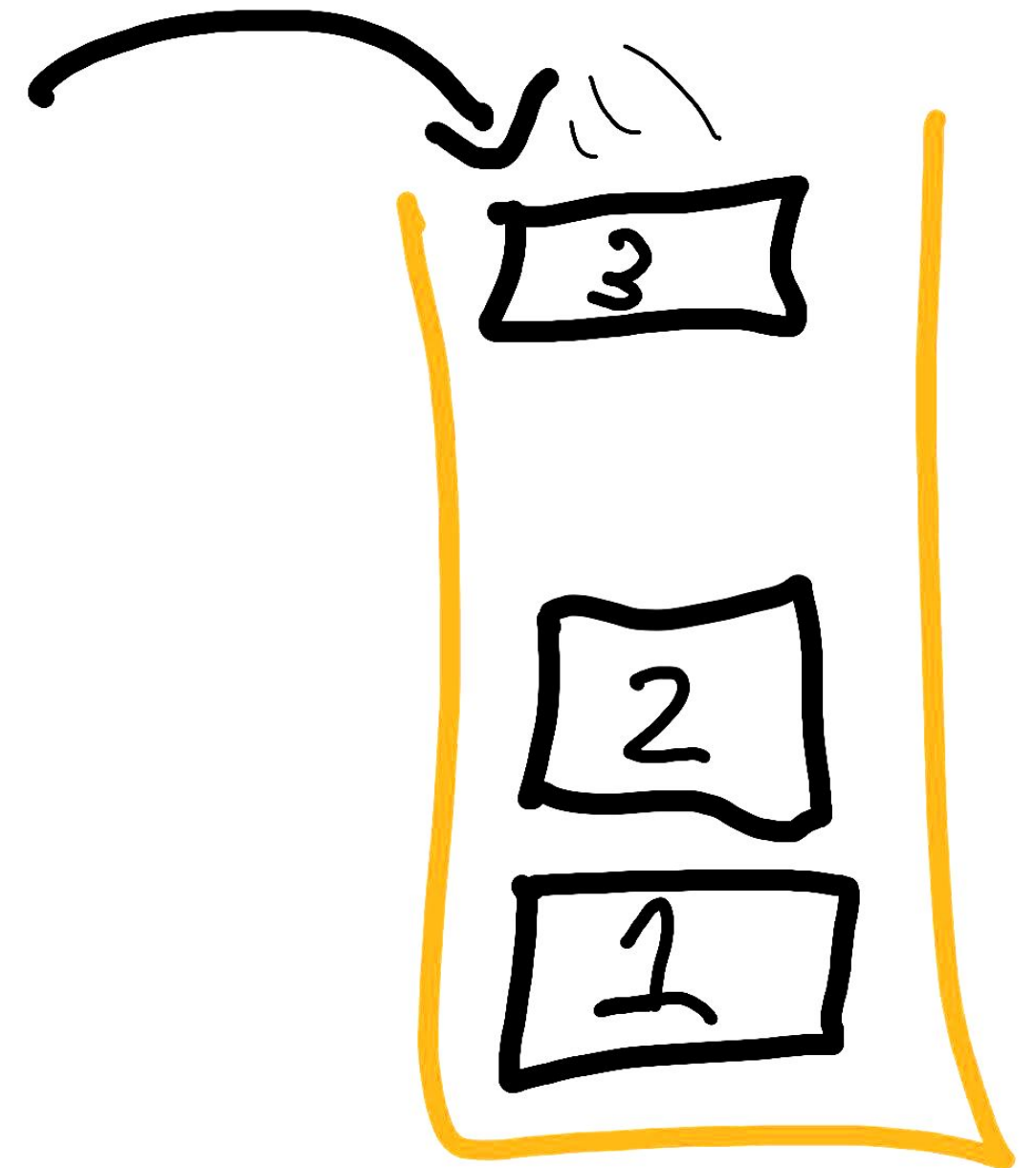
## Operaciones

pop ( )	Remueve el <u>último</u> valor insertado
push(item)	Agrega un nuevo valor “ <u>arriba</u> ” del stack.
peek ( )	Regresa el <u>último</u> valor insertado (sin borrarlo)
isEmpty ( )	Regresa true si la pila está vacía



**VS Arreglos**

- Pilas no ofrecen tiempo constante para obtener el n-esimo valor.
- Pero si es constante en borrar y agregar valores, lo cual no requiere “shifting”.






Pilas

¿Cómo se  
implementan?

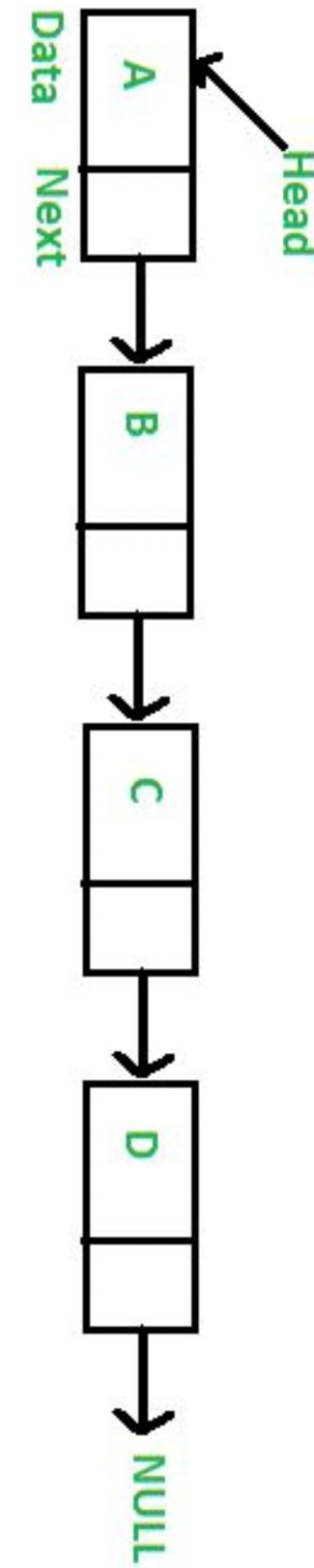
D{VZ

En algunos lenguajes, un array puede ser usado como pila. Pero de manera general, usaremos la estructura “node” que vimos en listas enlazadas:



```
class Node<T> {  
    T data;  
    Node next;  
  
    Node(T data) { this.data = data }  
}
```

¡De hecho, una pila puede verse como una lista ligada que funciona de manera específica!



Pilas

# ¿Cómo funciona una Cola?

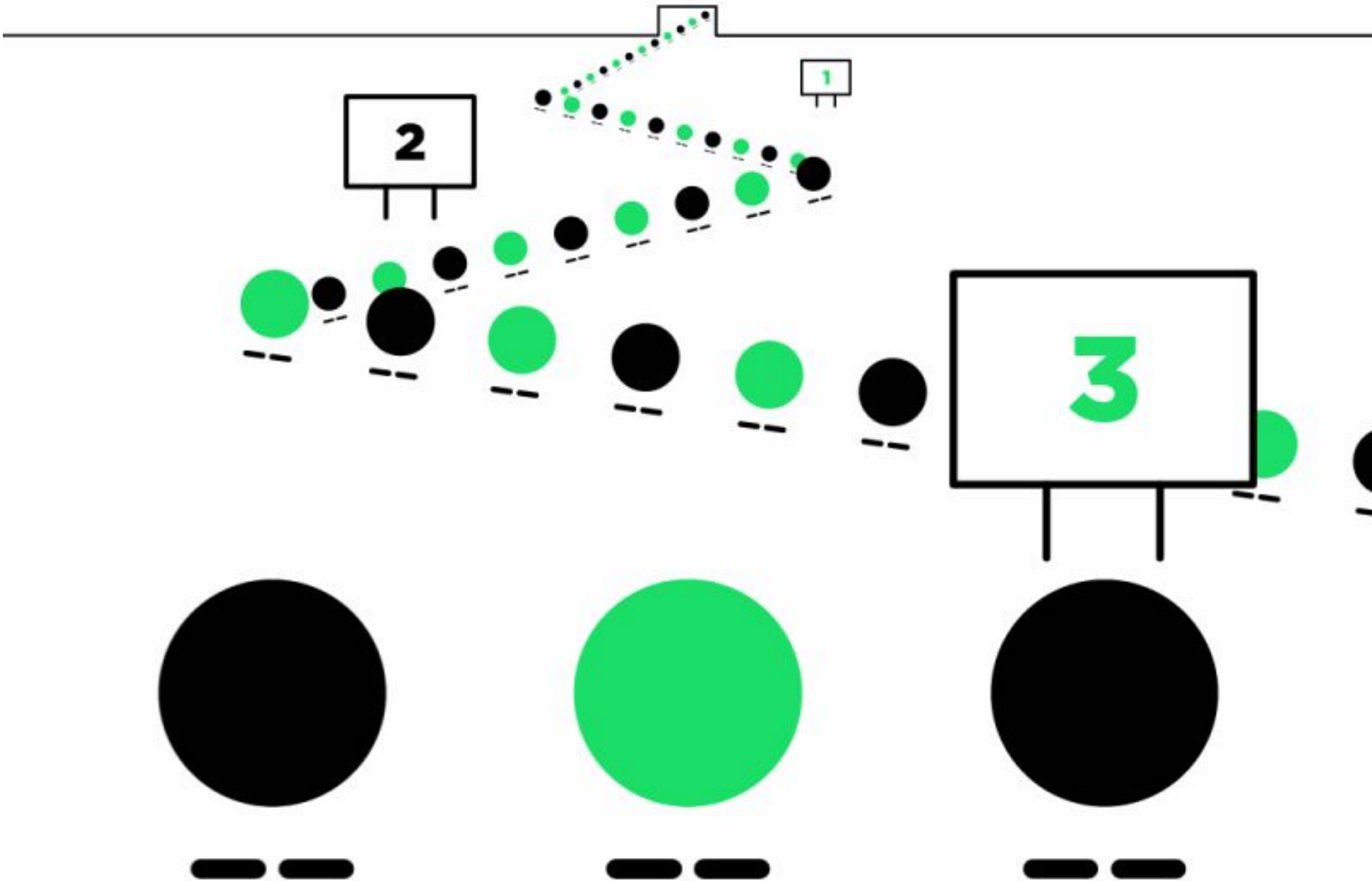
D{VZ



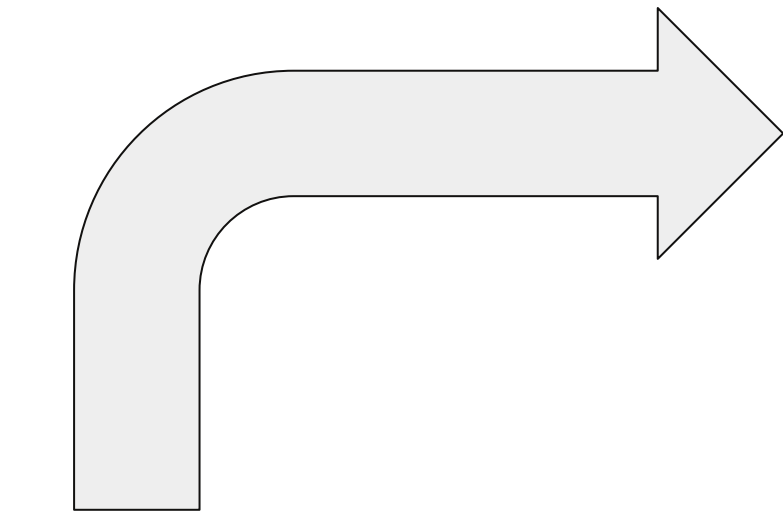
**Definición**

Estructura de datos en donde el primer valor insertado es el primero en salir.

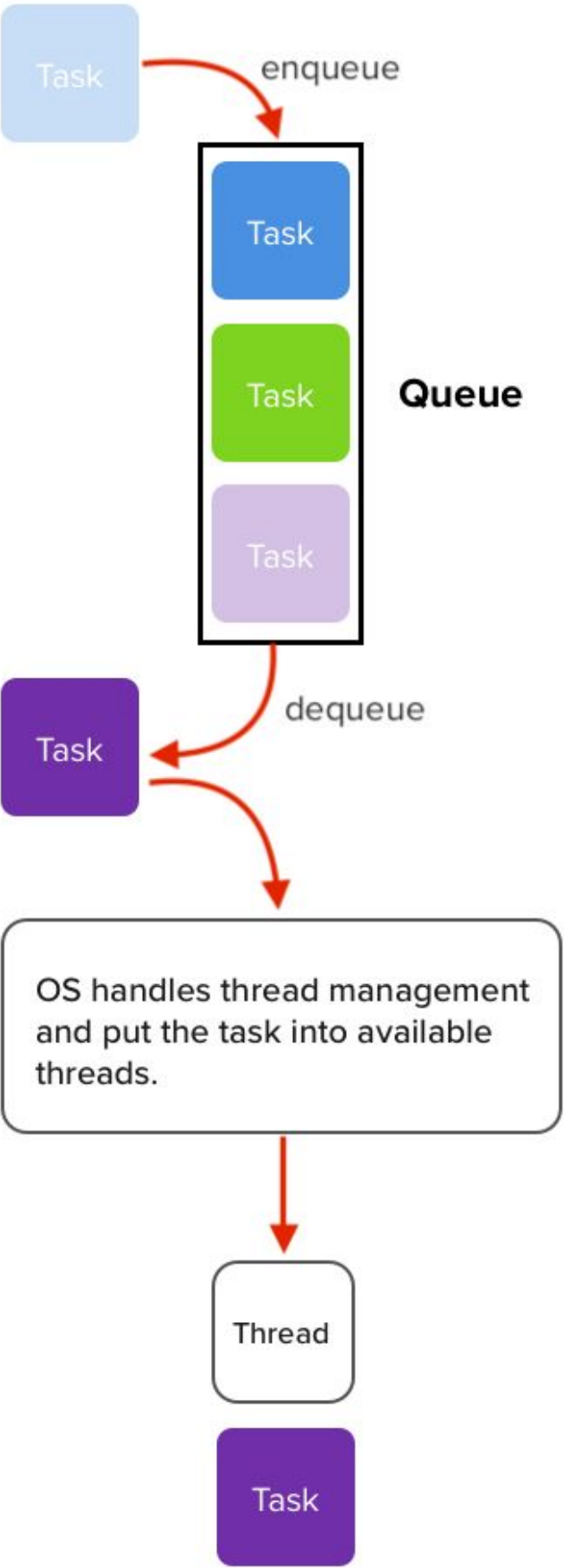
(FIFO - First In First Out)





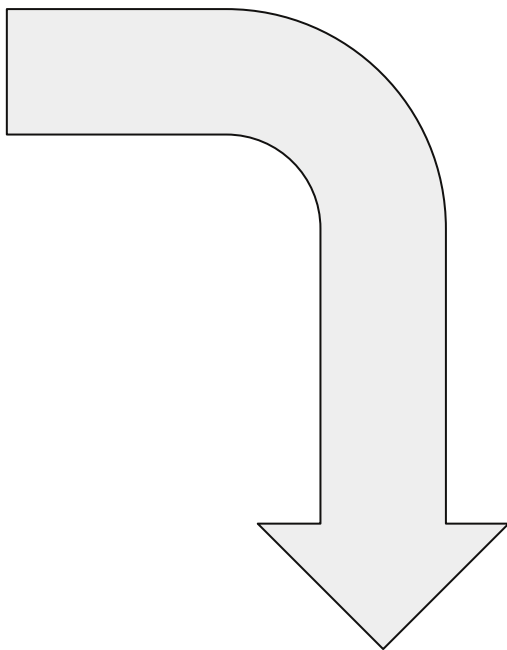


Queue of processes

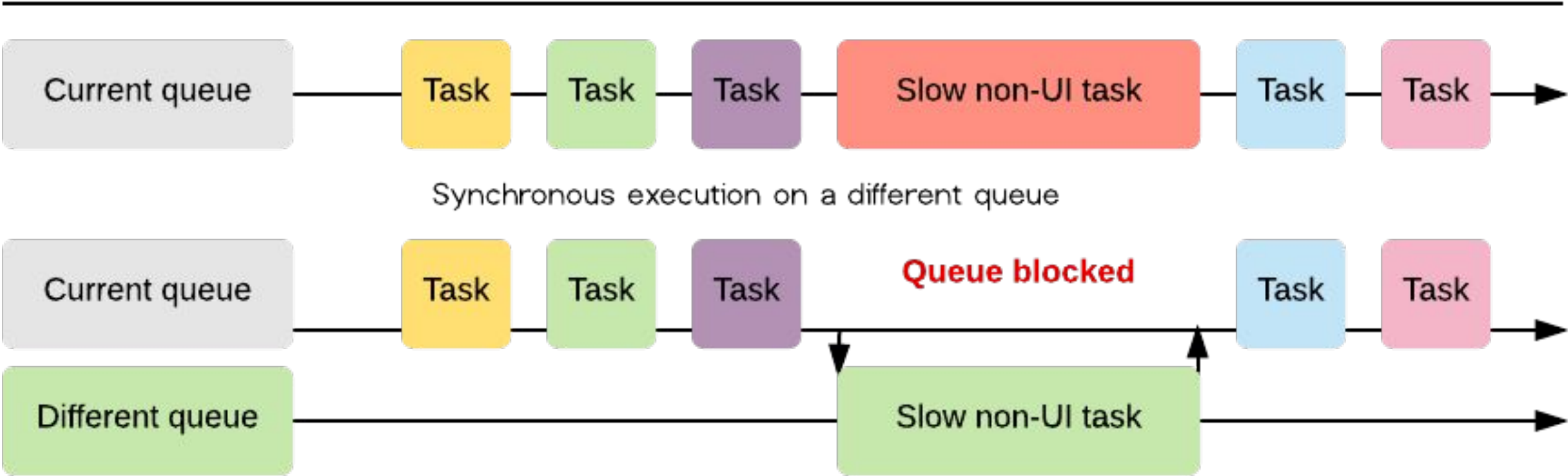




# Grand Central Dispatch in iOS

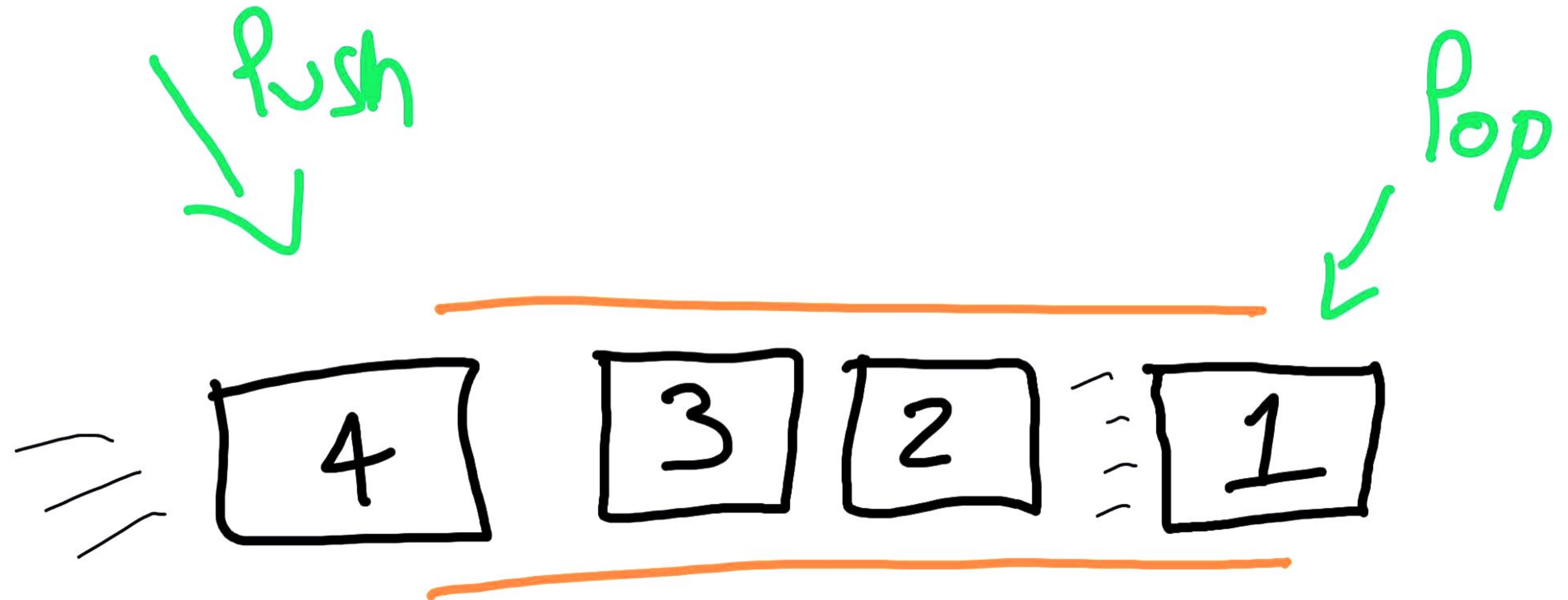


Synchronisity



Operaciones

pop ( )	Remueve el <u>primer</u> valor insertado
push(item)	Agrega un nuevo valor “ <u>al final</u> ” de la cola.
peek ( )	Regresa el <u>primer</u> valor insertado (sin borrarlo)
isEmpty ( )	Regresa true si la cola está vacía

**VS Arreglos**

- Colas no ofrecen tiempo constante para obtener el n-esimo valor.
- Pero si es constante en borrar y agregar valores, lo cual no requiere “shifting”.

**Tip**

Colas también pueden ser usadas para crear un cache o implementar búsquedas a lo ancho (Breadth-First-Search) .

¡Recuerden esto para futuras sesiones!




Colas

¿Cómo se  
implementan?

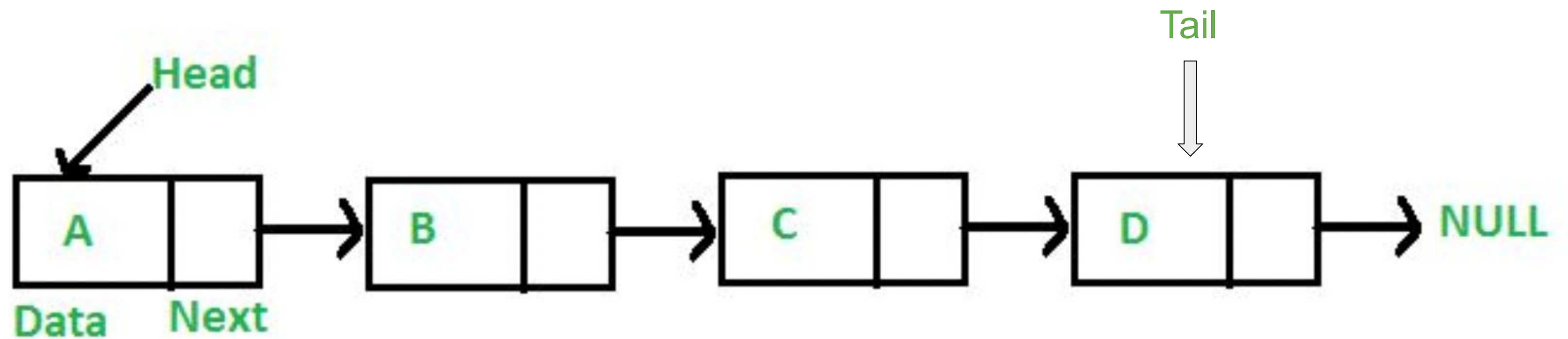
D{VZ

¡La misma estructura que una Pila!



```
class Node<T> {  
    T data;  
    Node next;  
  
    Node(T data) { this.data = data }  
}
```

Misma estructura, pero el orden de las operaciones cambia a FIFO.



# Pilas y Colas

## Actividades

- Lee sobre **pilas y colas**.
  - Página 96 de Cracking The Coding Interview.
  - Pilas en [wikipedia](#)
  - Colas en [wikipedia](#)
- **Recomendado** practicar e implementar en el lenguaje de tu preferencia.
  - Crear ambas estructuras con sus respectivas operaciones para un ejemplo con enteros.
- Resuelve los problemas requeridos de la semana
- **Bonus:** Resuelve los problemas opcionales de la semana

EJERCICIOS

# Problemas Requeridos.

D{VZ



**Problema #1**

Implementa un stack de enteros con sus operaciones push, pop, peek y una funcion getMin que obtendra el valor mas pequeño de la pila. Todas las operaciones (incluyendo getMin) deben ser **constantes** ( $O(1)$ ).

*Ejemplos*

```
//Output
let minStack = MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

## Problema #2

Implementar una cola usando dos pilas.

Asumiendo que solo pudieras usar las operaciones de pila, ¿Cómo conseguirías el comportamiento de una cola?

Fuente: <https://leetcode.com/problems/implement-queue-using-stacks/>

**Problema #3**

Crea una función que regrese el mínimo número de paréntesis que hay que agregar para que los paréntesis sean válidos.

**In:** “())”

**Out:** 1

Porque agregando uno al inicio se satisface la condición: “(())”

**In:** “()))((”

**Out:** 4

Porque agregando 2 al inicio y 2 al final se satisface la condición: “((()))((”

EJERCICIOS

# Problemas Opcionales.

D{VZ

**Problema #4**

Dado una lista de procesos, cada proceso sólo puede tener un proceso “Padre”, pero de 0 a n procesos hijos. Implementa una función `killProcess` para cerrar un proceso y a sus procesos hijo. Dicha función recibe los siguientes parámetros:

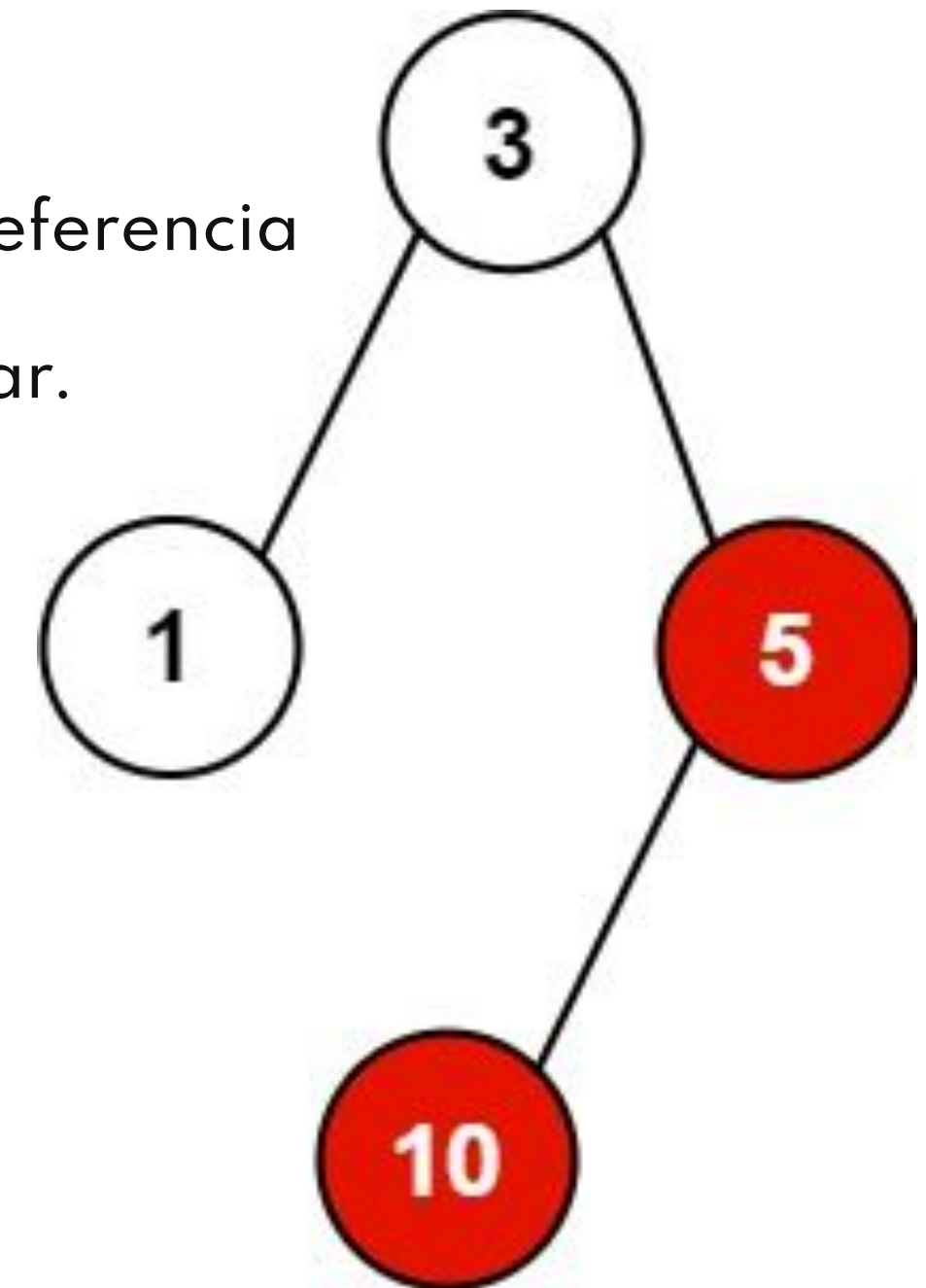
- Un arreglo `pid`, que son los ids de cada proceso, por ejemplo: `[1, 3, 5, 10]`
- Un arreglo `ppid`, que es el padre de cada proceso (utiliza 0 para hacer referencia al padre nodo raíz), por ejemplo: `[3, 0, 3, 5]`
- Un valor `kill` de tipo entero, que será el id del proceso que quieres eliminar.

Regresa el proceso eliminado junto con sus hijos.

**In:** `pid: [1, 3, 5, 10], ppid: [3, 0, 3, 5], kill: 5`

**Out:** `[5, 10]`

Fuente: <https://leetcode.com/problems/kill-process/>





### Problema #5

Dada un string, implementa una función “calculate” que tal cual simula las operaciones de una calculadora y regrese el resultado.

Nota: La división debe ser trunca a entero.

Ejemplos:

**In:** “7 + 3 \* 5”

**Out:** 22

**In:** “3+5/2”

**Out:** 5

# Gracias.



slack.devz.mx



@devzcommunity