

Estructura de datos y algoritmos

Rodrigo Alvarez

rodrigo.alvarez2@mail.udp.cl

Laboratorio 1

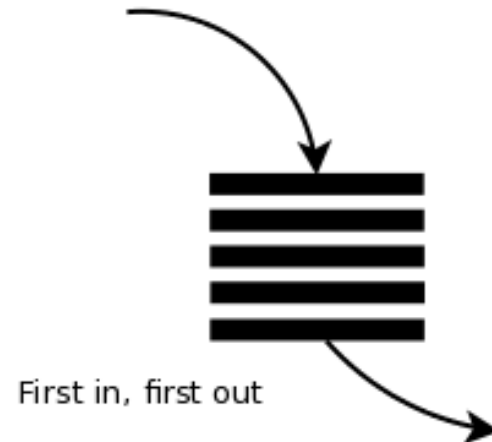
Pilas y colas

- **pila:** El único elemento accesible es el ultimo añadido.
- **cola:** El único elemento accesible es el que se añadió más temprano.
- Menos posibilidades de operaciones que una lista pero más eficientes.

Stack:



Queue:



Recordatorio: TDA

- **Tipo de dato abstracto (TDA):** Una especificación de un conjunto de datos y las operaciones que pueden ser realizadas sobre esos datos.
 - Describe qué hace, no cómo lo hace.
- No sabemos exactamente cómo se implementa, solo qué hace y qué operaciones podemos hacer.
 - Solo necesitamos entender la idea de la colección de datos y las operaciones que puede realizar.

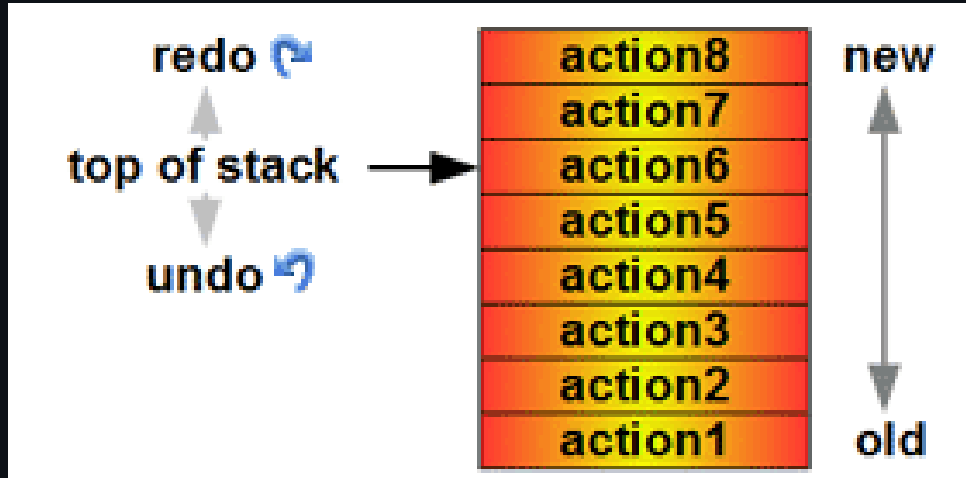
las pilas y colas usualmente se implementan con listas enlazadas

Pila (stack)

- **LIFO**: Last In First Out
 - Los elementos son guardados en el orden de inserción
 - No solemos pensar en la posición de los elementos, solo en el último añadido.
 - El "cliente" solo puede acceder al último elemento añadido.
-
- Operaciones básicas:
 - push: añadir un elemento
 - pop: quitar el último elemento añadido
 - peek (o top): ver el último elemento añadido

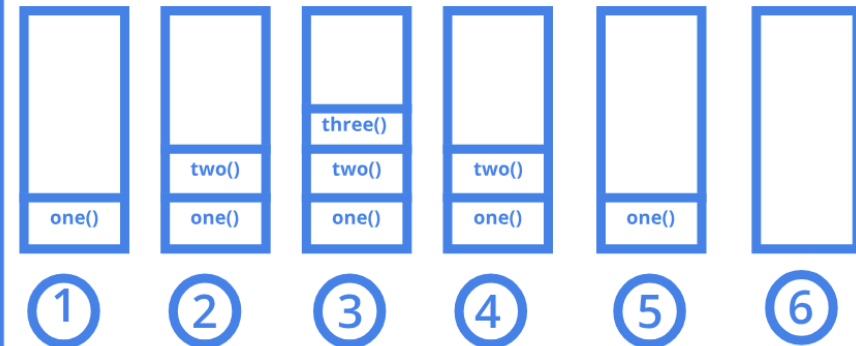


Pila (stack)



```
function one(){  
  two();  
}  
  
function two() {  
  three();  
}  
  
function three() {  
  console.trace("Call  
Stack");  
}
```

Call Stack



Class Stack

java.util.Stack

Constructor Summary

Constructors

| Constructor and Description |
|---|
| Stack() Creates an empty Stack. |

Method Summary

| All Methods | Instance Methods | Concrete Methods |
|-------------------|--|------------------|
| Modifier and Type | Method and Description | |
| boolean | empty() Tests if this stack is empty. | |
| E | peek() Looks at the object at the top of this stack without removing it from the stack. | |
| E | pop() Removes the object at the top of this stack and returns that object as the value of this function. | |
| E | push(E item) Pushes an item onto the top of this stack. | |
| int | search(Object o) Returns the 1-based position where an object is on this stack. | |

Class stack

Main.java

+

428jybu4h

NEW

JAVA ▾

RUN ▶

⋮

1

STDIN

Input for the program (Optional)

Output:

[]

- No se suele iterar sobre una pila.
- Se usa un bucle `while` para vaciar la pila.

```
while (!stack.isEmpty()) {  
    System.out.println(stack.pop());  
}
```

https://visualgo.net/en/stack

7 VISUALGO.NET/en /list LL **STACK** QUEUE DLL DEQUE

Exploration Mode ▾ LOGIN

<

Create(A)
Peek
Push
Pop

<

1x

⏮ ⏪ ⏸ ⏩ ⏭

About Team Terms of use Privacy Policy

```
class Stack {
    class Node {
        int data;
        Node next;
        public Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    Node head;
    public Stack() {
        this.head = null;
    }
    public void push(int e) {
        Node newNode = new Node(e);
        if (head != null) {
            newNode.next = head;
        }
        head = newNode;
    }
    public int pop() {
        int e = head.data;
        head = head.next;
        return e;
    }
    public int peek() {
        return head.data;
    }
}
```

Complejidad de las operaciones de una pila:

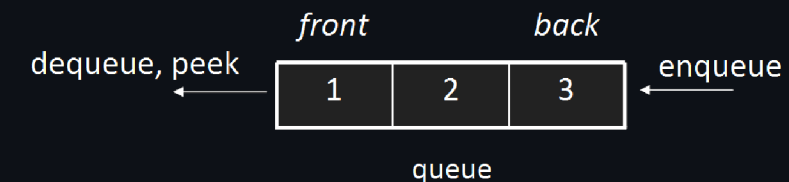
- `push` : $O(1)$
- `pop` : $O(1)$
- `peek` : $O(1)$

Cola (queue)

- **FIFO**: First In First Out
- Los elementos son guardados en el orden de inserción y no suelen tener índices.
- El "cliente" puede añadir elementos al final y examinar/quitar elementos del principio.



- Operaciones básicas:
 - enqueue: añadir un elemento al final
 - dequeue: quitar el elemento del principio
 - peek: ver el elemento del principio



Cola (queue)



www.datainfinities.com

Interface Queue

```
java.util.Queue
```

Method Summary

| All Methods | Instance Methods | Abstract Methods |
|-------------------|------------------------|--|
| Modifier and Type | Method and Description | |
| boolean | add(E e) | Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an <code>IllegalStateException</code> if no space is currently available. |
| E | element() | Retrieves, but does not remove, the head of this queue. |
| boolean | offer(E e) | Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions. |
| E | peek() | Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty. |
| E | poll() | Retrieves and removes the head of this queue, or returns null if this queue is empty. |
| E | remove() | Retrieves and removes the head of this queue. |

https://visualgo.net/en/queue

7 VISUALGO.NET / en /list LL STACK **QUEUE** DLL DEQUE

Exploration Mode ▾ LOGIN

<

Create(A)
Peek
Enqueue
Dequeue

<

1x

⏮ ⏪ ⏸ ⏩ ⏭

About Team Terms of use Privacy Policy


```
class Queue {
    class Node {
        int data;
        Node next;
        public Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    Node head, tail;
    public Queue() {
        this.head = this.tail = null;
    }
    public void enqueue(int e) {
        Node newNode = new Node(e);
        if (head == null) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            tail = newNode;
        }
    }
    public int dequeue() {
        int e = head.data;
        head = head.next;
        return e;
    }
    public int peek() {
        return head.data;
    }
}
```

Complejidad de las operaciones de una cola:

- enqueue : $O(1)$
- dequeue : $O(1)$
- peek : $O(1)$

Ejercicios

- List
 - Reverse a doubly linked list (easy)
 - Cycle Detection (medium)
 - Get Node Value (easy)
- Stack
 - Equal stacks (easy)
 - Balanced brackets (medium)
- Queue
 - Queue using two stacks (medium)
-