

Estructura de datos y algoritmos

Rodrigo Alvarez

rodrigo.alvarez2@mail.udp.cl

Stack

 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:

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

Queue

The image is a screenshot of the Visualgo website. At the top, there is a navigation bar with the Visualgo logo, a language selector set to 'en', and links for 'list', 'LL', 'STACK', 'QUEUE' (which is highlighted), 'DLL', and 'DEQUE'. On the right side of the top bar, it says 'Exploration Mode' with a dropdown arrow and a 'LOGIN' button. The main area is a large, dark rectangle representing the simulation. On the left side of this area, there is a white menu with the following options: 'Create(A)', 'Peek', 'Enqueue', and 'Dequeue'. At the bottom of the page, there is a footer with a '1x' zoom indicator and links for 'About', 'Team', 'Terms of use', and '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:

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

- Stack using two queues
- Balanced brackets
- Reverse a doubly linked list
- Add two numbers
- Merge two sorted lists