

# Estructura de datos y algoritmos

Rodrigo Alvarez

[rodrigo.alvarez2@mail.udp.cl](mailto:rodrigo.alvarez2@mail.udp.cl)

# Stack

 VISUALGO.NET / en / list

LL **STACK** QUEUE DLL DEQUE

Exploration Mode ▾

LOGIN

<

Create(A)  
Peek  
Push  
Pop

>

■ 1x

AboutTeamTerms of usePrivacy 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 screenshot shows the Visualgo website interface for the Queue data structure. The top navigation bar includes the Visualgo logo, a language dropdown set to 'en', and links for 'list', 'LL', 'STACK', 'QUEUE' (highlighted), 'DLL', and 'DEQUE'. On the right, it says 'Exploration Mode' with a dropdown arrow and a 'LOGIN' button. The main content area is a large black rectangle. On the left side of this area, there is a vertical menu with a back arrow and four options: 'Create(A)', 'Peek', 'Enqueue', and 'Dequeue'. At the bottom left of the main area, there is a '1x' zoom indicator. At the bottom right, there are 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 :  $O(1)$

