STACKS & QUEUES

Ramiro Guillen
Pierre Campoverde
Adrian Mendoza

STACK

```
using System.Collections;

0 referencias
public class SamplesStack
{
    0 referencias
    public static void Main()
    {
        // Creates and initializes a new Stack.

        Stack myStack = new Stack();
        myStack.Push("Hello");
        myStack.Push("World");
        myStack.Push("!");

        // Displays the properties and values of the Stack.
        Console.WriteLine("myStack");
        Console.WriteLine("\tCount: {0}", myStack.Count);
        Console.Write("\tValues:");
        PrintValues(myStack);
    }

    1 referencia
    public static void PrintValues(IEnumerable myCollection)
    {
        foreach (Object obj in myCollection)
            Console.Write("    {0}", obj);
        Console.WriteLine();
    }
}

/*
This code produces the following output.

myStack
    Count:    3
    Values:    !    World    Hello
*/
```

- Implementa una política de Last-In, First-Out o LIFO, en la cual se remueve el elemento que se insertó más recientemente.
- Para instanciar la clase se utiliza: Stack stack = new Stack();
- Para insertar un elemento al stack se utiliza el método Push.
- Para eliminar y devolver el elemento superior del stack se utiliza el método Pop.
- También es posible obtener el elemento superior del stack sin eliminarlo utilizando Peek.

QUEUE

- Implementan una política de First-In, First-Out, en la cual se remueve el primer elemento que se insertó en la colección.
- Para instanciar la clase se utiliza: Queue queue = new Queue();
- Para insertar un elemento al queue se utiliza el método Enqueue.
- Para eliminar y devolver el elemento superior del queue se utiliza el método Dequeue.
- También es posible obtener el elemento superior del queue sin eliminarlo utilizando Peek.

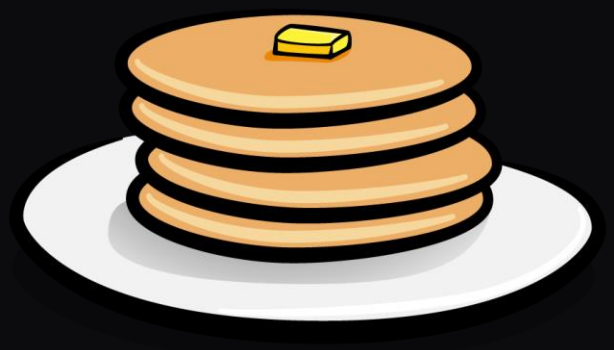
```
using System.Collections;

0 referencias
public class SamplesQueue
{
    0 referencias
    public static void Main()
    {
        // Creates and initializes a new Queue.
        Queue myQ = new Queue();
        myQ.Enqueue("Hello");
        myQ.Enqueue("World");
        myQ.Enqueue("!");

        // Displays the properties and values of the Queue.
        Console.WriteLine("myQ");
        Console.WriteLine("\tCount: {0}", myQ.Count);
        Console.Write("\tValues:");
        PrintValues(myQ);
    }

    1 referencia
    public static void PrintValues(IEnumerable myCollection)
    {
        foreach (Object obj in myCollection)
            Console.Write("{0}", obj);
        Console.WriteLine();
    }
}

/*
This code produces the following output.

myQ
        Count:    3
        Values:    Hello    World    !
*/
```

USE CASES - STACKS

- Evaluation and Conversion of Expressions
- Navigation History in Web Browsers
- Management of Calls and Returns in Programs

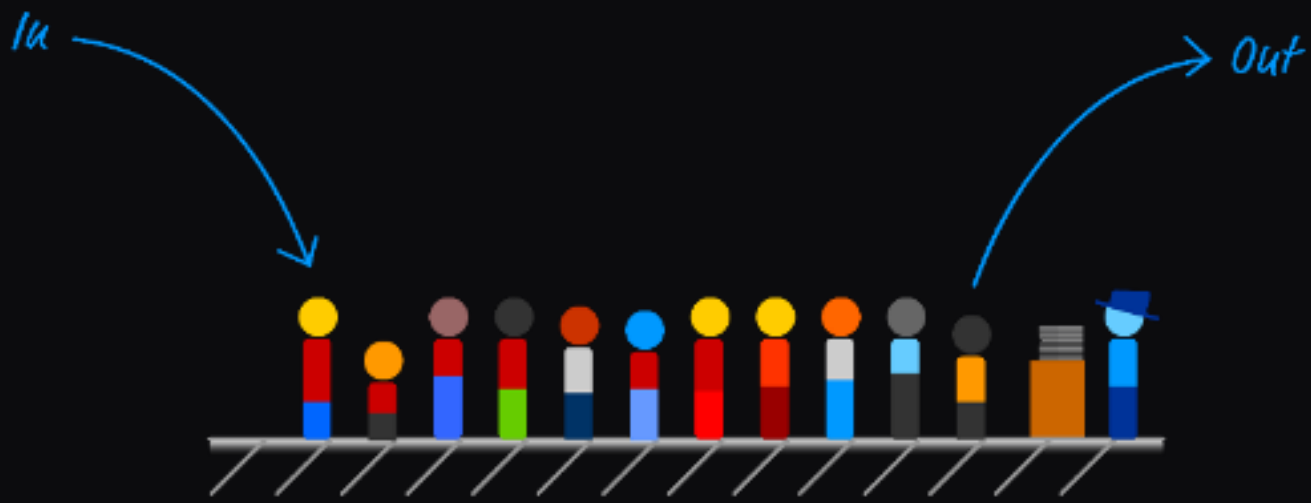
USE CASES - STACKS

- Development of Search and Traversal Algorithms
- Undo and Redo Functionalities in Applications
- Syntax Analysis in Compilers



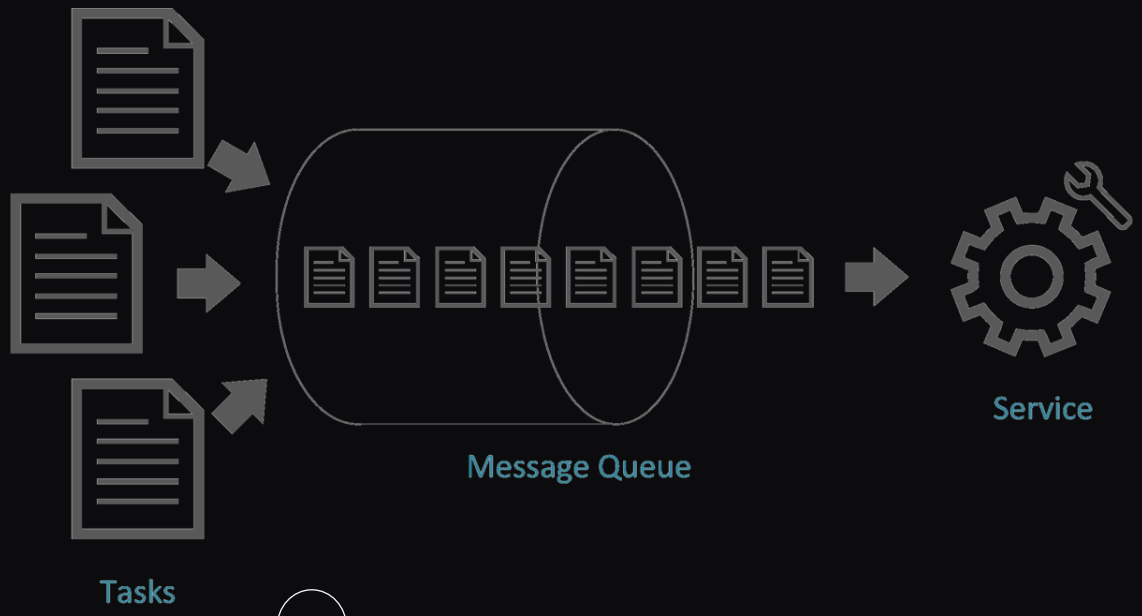
USE CASES - QUEUES

- Simulation of Waiting Lines
- Implementation of Buffers
- Planning Routes or Itineraries:



USE CASES - QUEUES

- Task and Process Management in Operating Systems
- Handling Requests in Web Servers
- Print Queues



<DEMO/>