[Create] Stack

```
class MyStack {
    // Do not allow to change the initialized number of
the array
    private int[] data = new int[8];
    private int count = 0;

    public void push(int value) {
        // To be implemented
        // Need to extend the array size when not enough
by calling resize() method
       if (count == data.length) resize();
       data[count] = value;
       count++;
    }

    public int pop() throws Exception {
        // To be implemented
       if (count == 0) throw new Exception("Stack is
empty");
       int index = count-1;
       int result = data[index];
       data[index] = 0;
       count--;
       return result;
    }

    public int peek() throws Exception {
        // To be implemented
       if (count == 0) throw new Exception("Stack is
empty");

       return data[count];
    }

    private void resize() {
        // To be implemented
       int length = data.length;
       int newLength = data.length + 8;
       int[] tempData = data;
```

```
        data = new int[newLength];

        for (int i = 0; i < length; i++) {
            data[i] = tempData[i];
        }
    }



    @Override
    public String toString() {
        String result = "";
        for (int i : data) {
            result += " " + i;
        }
        return result;
    }

    public static void main(String[] args) throws
Exception {
        MyStack stack = new MyStack();
        stack.push(4);
        stack.push(5);

        System.out.println("Stack: " + stack.toString());


        stack.pop();
        stack.push(6);

        System.out.println("Stack: " + stack.toString());

    }
}
```

```
package com.tiendn.javacode.mpp_082019;

class Node {
    String data;
    Node nextNode;

    public Node(String data) {
        this.data = data;
```

```
        nextNode = null;
    }
}

public class Stack {
    private Node head = null;
    private int size = 0;

    public int getSize() {
        return size;
    }


    public void push(String item) {
        Node newNode = new Node(item);
        if (isEmpty()) {
            head = newNode;
        } else {
            newNode.nextNode = head;
            head = newNode;
        }
        size++;
    }

    public String pop() throws Exception {
        if (isEmpty()) throw new Exception("Stack is
empty");
        Node currNode = head;
        head = currNode.nextNode;
        currNode.nextNode = null;
        return currNode.data;
    }

    public String peek() throws Exception {
        if (isEmpty()) throw new Exception("Stack is
empty");
        return head.data;
    }

    public boolean isEmpty() {
        return size == 0;
    }
```

```
    @Override
    public String toString() {
        String result = "";
        Node currNode = head;
        while (currNode != null) {
            result += currNode.data;
            currNode = currNode.nextNode;
        }
        return result;
    }

    public static void main(String[] args) throws
Exception {
        Stack stack = new Stack();
        stack.push("A");
        stack.push("B");

        System.out.println("Stack: " + stack.toString());


        stack.pop();
        stack.push("C");

        System.out.println("Stack: " + stack.toString());

    }
}
```