算法 04 栈和队列

作者：AshShawn
原文链接：https://ld246.com/article/1646216106806
来源网站：链滴

1.循环队列

问题: 用数组实现一个循环队列,实现poll和offer方法

```
public class CycleArray {
    private int size;
    private int front;
    private int rear;
    private int[] arr;

    public CycleArray(int size) {
        this.size = 0;
        this.front = 0;
        this.rear = -1;
        arr = new int[size];
    }

    public int poll() {
        if (size == 0) {
            throw new RuntimeException("队列为空");
        }
        int i = arr[front];
        decreaseFront();
        size--;
        return i;
    }

    public boolean offer(Integer node) {
        if (size == arr.length) {
            return false;
        }
        increaseRear();
        arr[rear] = node;
        size++;
        return true;
    }

    private void increaseRear() {
        if (rear == arr.length - 1) {
            rear = 0;
        } else {
            rear++;
        }
    }

    private void decreaseFront() {
        if (front == 0) {
            front = arr.length - 1;
        } else {
            front--;
        }
    }
}
```

2.用栈结构实现队列

问题：使用栈结构实现一个先进先出的队列结构

分析:

1. 使用两个栈,一个为push栈,存放用户push进来的数据
2. 另一个为pop栈,用来存放pop出去的数据
3. push栈导数据到pop栈时必须一次性全部完成
4. pop栈中有数据,不能从push栈中导数据

```
public static class TwoStacksQueue {
    public Stack<Integer> stackPush;
    public Stack<Integer> stackPop;

    public TwoStacksQueue() {
        stackPush = new Stack<Integer>();
        stackPop = new Stack<Integer>();
    }

    // push栈向pop栈倒入数据
    private void pushToPop() {
        if (stackPop.empty()) {
            while (!stackPush.empty()) {
                stackPop.push(stackPush.pop());
            }
        }
    }

    public void add(int pushInt) {
        stackPush.push(pushInt);
        pushToPop();
    }

    public int poll() {
        if (stackPop.empty() && stackPush.empty()) {
            throw new RuntimeException("Queue is empty!");
        }
        pushToPop();
        return stackPop.pop();
    }

    public int peek() {
        if (stackPop.empty() && stackPush.empty()) {
            throw new RuntimeException("Queue is empty!");
        }
        pushToPop();
        return stackPop.peek();
    }
}
```

3.用队列实现栈

问题: 用队列实现栈结构

分析:

1. 队列是先进先出,栈是先进后出
2. 可以使用两个队列来回倒
3. 来回倒时,最后一个node就是栈的最上一个元素

```
public static class TwoQueueStack<T> {
    public Queue<T> queue;
    public Queue<T> help;

    public TwoQueueStack() {
        queue = new LinkedList<>();
        help = new LinkedList<>();
    }

    public void push(T value) {
        queue.offer(value);
    }

    public T poll() {
        while (queue.size() > 1) {
            help.offer(queue.poll());
        }
        T ans = queue.poll();
        Queue<T> tmp = queue;
        queue = help;
        help = tmp;
        return ans;
    }

    public T peek() {
        while (queue.size() > 1) {
            help.offer(queue.poll());
        }
        T ans = queue.poll();
        help.offer(ans);
        Queue<T> tmp = queue;
        queue = help;
        help = tmp;
        return ans;
    }

    public boolean isEmpty() {
        return queue.isEmpty();
    }

}
```