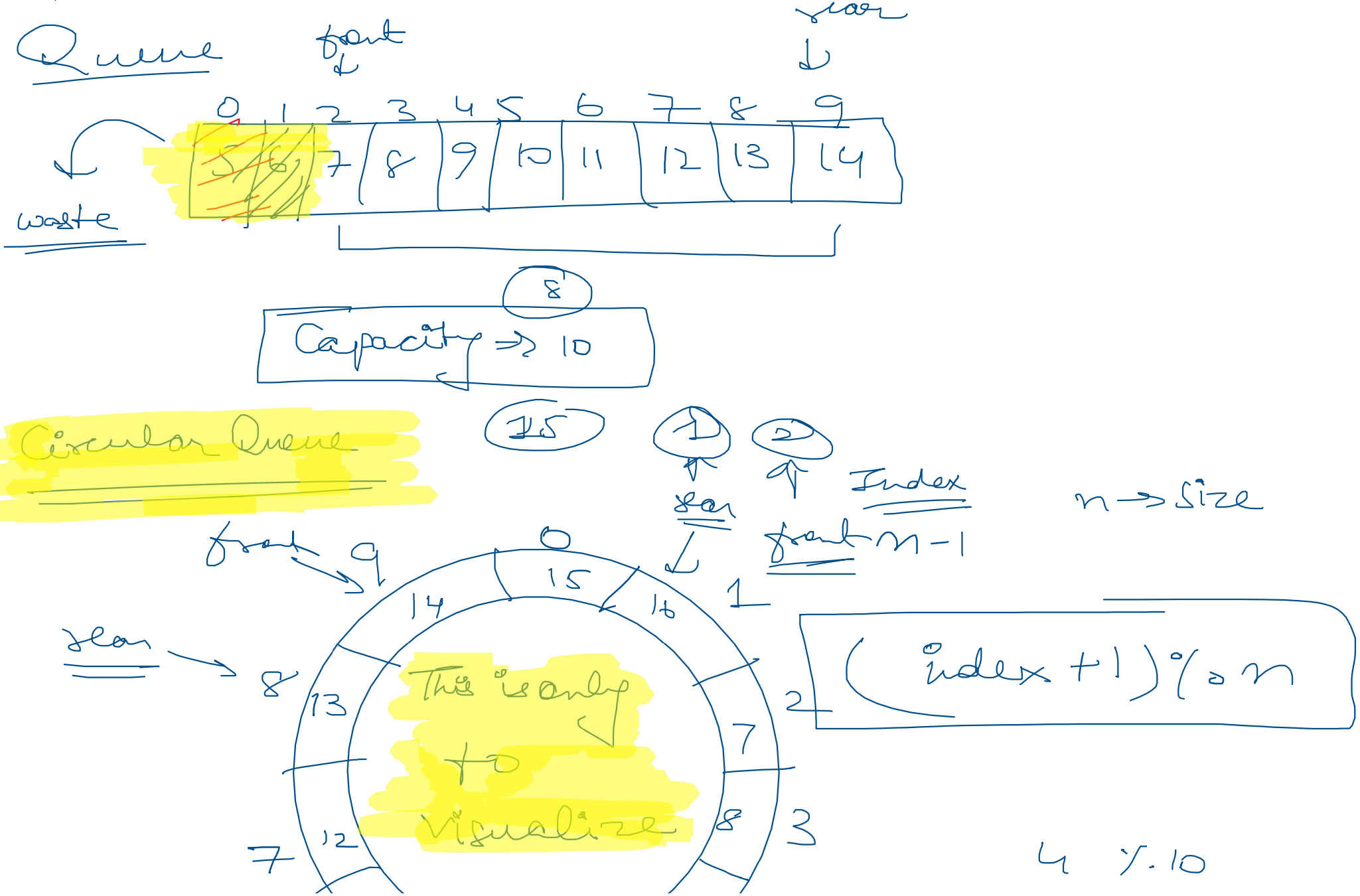
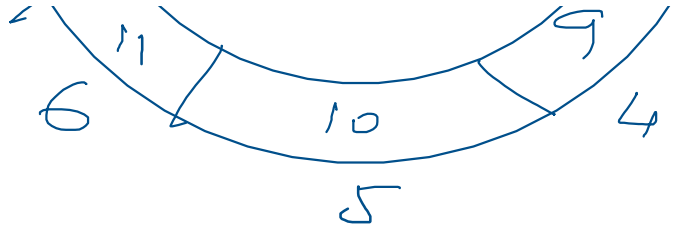
Stacks & Queues, July 27

27 July 2020 19:11





$(9+1) \% 10 = 0$

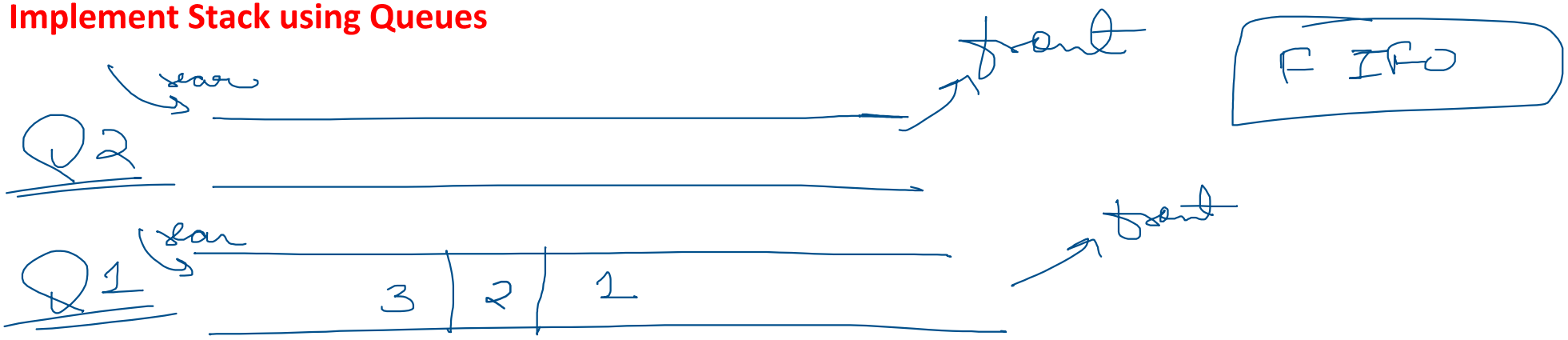
front is next of rear

↳ Queue full

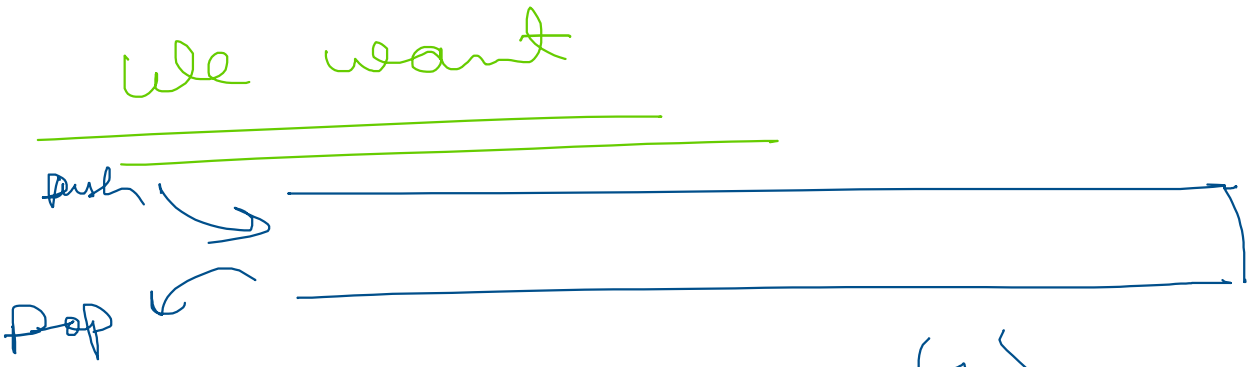
front = (rear + 1) % n

Task: Linked List Implementation of Queue

Implement Stack using Queues



FIFO



O(1)

Method - 1

Making Pop operation costly:

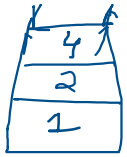
Push Operation:

1. **For ease lets suppose we will insert in Q1**

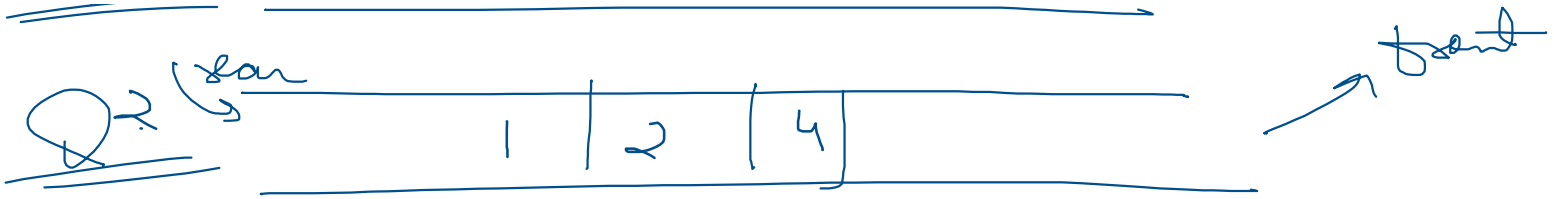
Pop Operation:

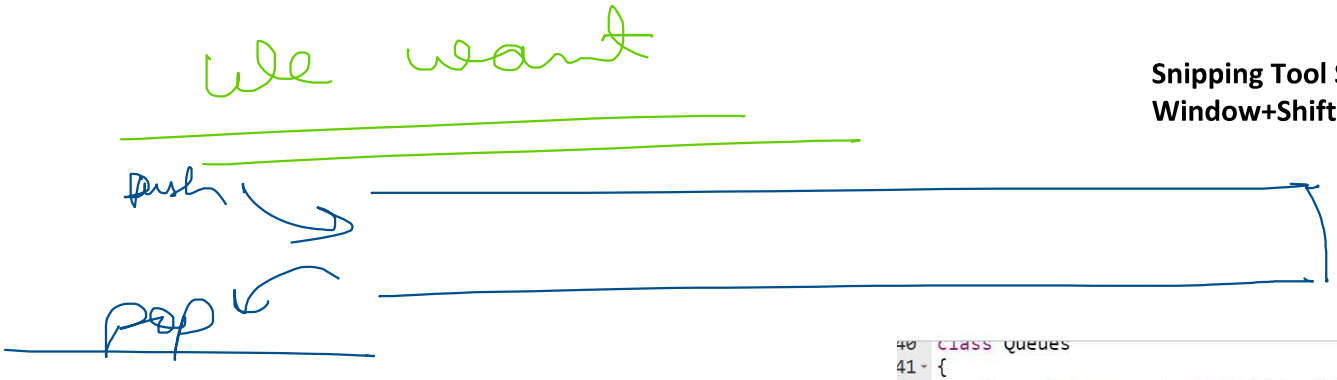
1. **Dequeue all elements except last from Q1 and Enqueue to Q2**
2. **Remove and return the last element**
3. **Swap Q1 and Q2**

```
42      Queue<Integer> q1 = new LinkedList<Integer>();
43      Queue<Integer> q2 = new LinkedList<Integer>();
44
45      /*The method pop which return the element poped out of the stack*/
46      int pop()
47      {
48          if(q1.isEmpty()) return -1;
49          // Your code here
50          while(q1.size()!=1)
51          {
52              q2.add(q1.poll());
53          }
54          int x = q1.poll();
55          //Swapping Q1 and Q2
56          Queue<Integer> temp = q1;
57          q1=q2;
58          q2=temp;
59          return x;
60      }
61
62      /* The method push to push element into the stack */
63      void push(int a)
64      {
65          // Your code here
66          q1.add(a);
67      }
```



Method - 2





Snipping Tool Shortcut:
Window+Shift+S

Making Push operation costly:
Pop Operation:

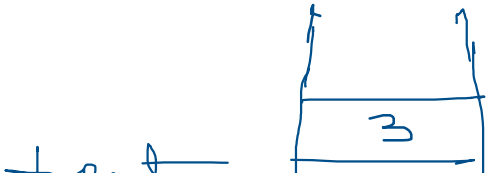
1. **For ease lets suppose we will dequeue from Q1**

Push Operation:

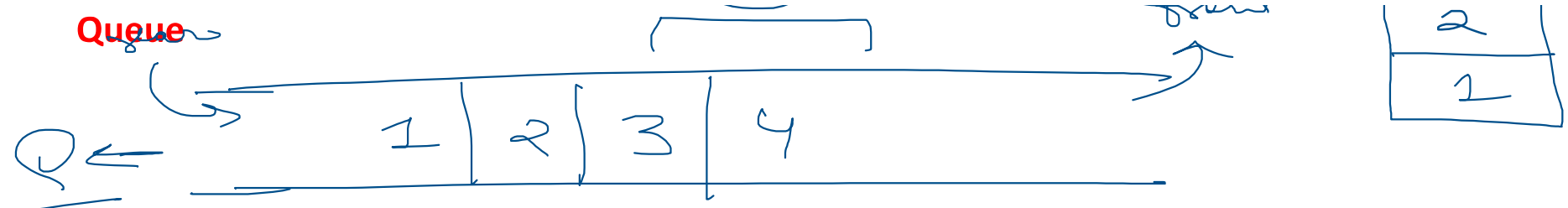
1. **Add new Element to Q2**
2. **Dequeue all the elements from Q1 and enqueue them to Q2**
3. **Swap Q1 and Q2**

```
class Queues
{
    Queue<Integer> q1 = new LinkedList<Integer>();
    Queue<Integer> q2 = new LinkedList<Integer>();

    /*The method pop which return the element poped out of the stack*/
    int pop()
    {
        if(q1.isEmpty()) return -1;

        return q1.poll();
    }

    /* The method push to push element into the stack */
    void push(int a)
    {
        // Your code here
        q2.add(a);
        while(!q1.isEmpty())
        {
            q2.add(q1.poll());
        }
        Queue<Integer> temp = q1;
        q1=q2;
        q2=temp;
    }
```

Method - 3 Using a single



Queue



Pop → Dequeue

Push

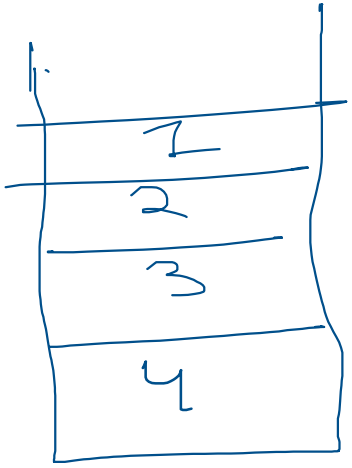
Queue

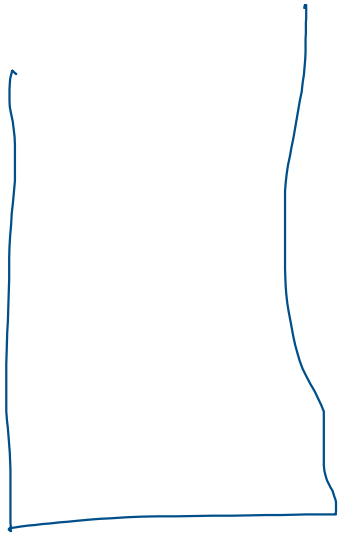
Array

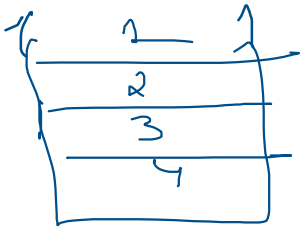
LL

Stack

Implement Queue using Stack







we want S1

S1 S2

4 | 3 | 2 | 1

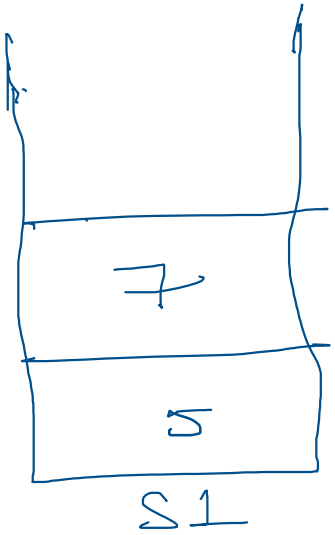
Method 2

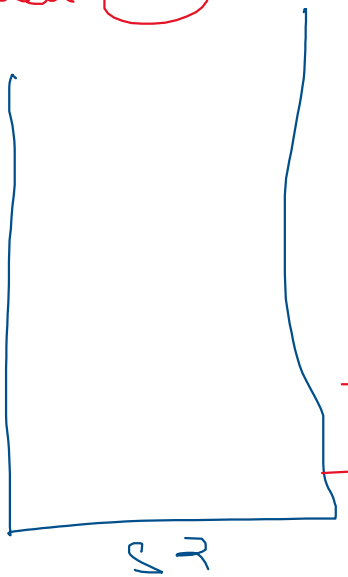
→ Making enqueue Costly
→ Put everything into S2
→ Put new element in S1
→ Put everything back into S1
→ Dequeue
→ Pop from S1

```
40 class StackQueue
41 {
42     Stack<Integer> s1 = new Stack<Integer>();
43     Stack<Integer> s2 = new Stack<Integer>();
44     /* The method insert to push element
45        into the queue */
46     void Push(int x)
47     {
48         while(!s1.isEmpty())
49         {
50             s2.push(s1.pop());
51         }
52         s1.push(x);
53         while(!s2.isEmpty())
54         {
55             s1.push(s2.pop());
56         }
57     }
58     /* The method remove which return the
59       element popped out of the queue*/
60     int Pop()
61     {
62         if(s1.isEmpty()) return -1;
63
64         return s1.pop();
65     }
66 }
```

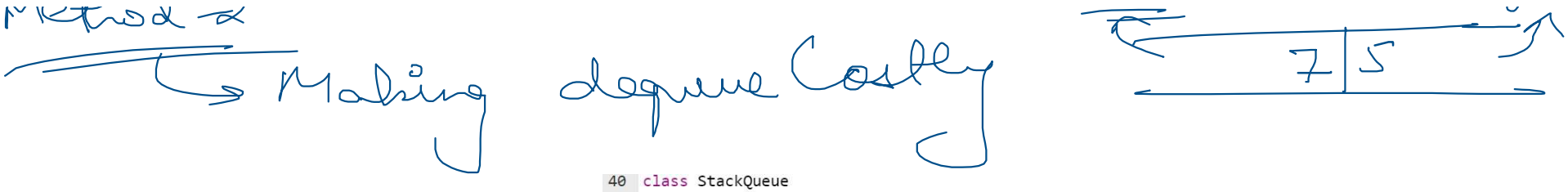
Method 2 Pop Costly

→ Better than Method (1)





Transfer only when S2 is empty




```
40 class StackQueue
41 {
42     Stack<Integer> s1 = new Stack<Integer>();
43     Stack<Integer> s2 = new Stack<Integer>();
44
45     /* The method insert to push element
46        into the queue */
47     void Push(int x)
48     {
49         s1.push(x);
50     }
51     /* The method remove which return the
52       element popped out of the queue*/
53     int Pop()
54     {
55        if(s1.isEmpty() && s2.isEmpty()) return -1;//If both empty then error
56        if(s2.isEmpty()) //Transfering only if s2 is empty
57        {
58            while(!s1.isEmpty())
59            {
60                s2.push(s1.pop());
61            }
62        }
63        return s2.pop();
64     }
65 }
```

Method 3

Use 1 user defined stack and use another memory stack. (Recursion)