Queue

First in, first out → FIFO

Syntax

Interface

Class

Queue < Integer > que = new ArrayDeque <>();

functions

1) add → que.add(5); ; que.offer(7);

2) remove → que.remove();

size → que.size();

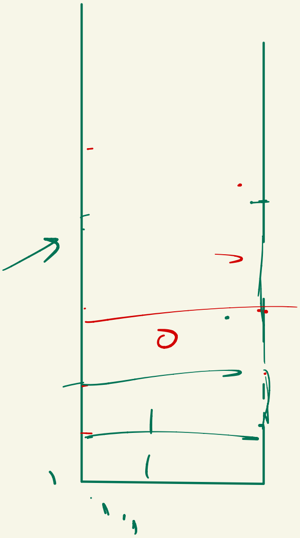
3) ; first → que.peek();

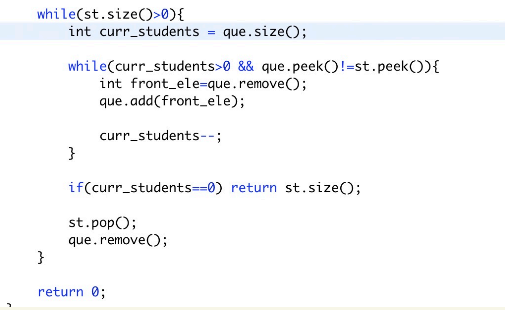
0 → circular

1 → square

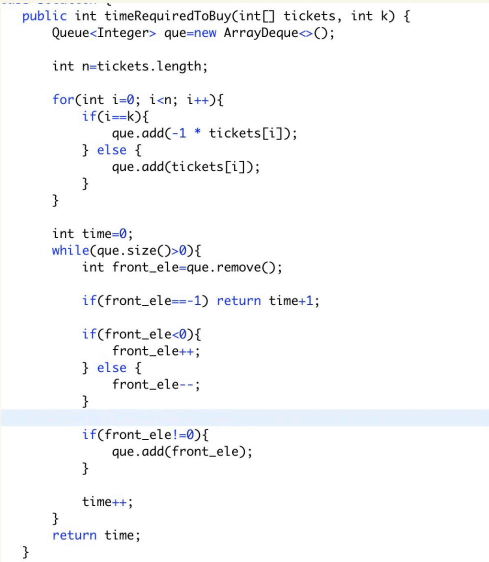
cs = ~~3~~ ~~2~~ ~~1~~ 0

0, 1, 1, 1





```
    while(st.size()>0){
        int curr_students = que.size();

        while(curr_students>0 && que.peek()!=st.peek()){
            int front_ele=que.remove();
            que.add(front_ele);

            curr_students--;
        }

        if(curr_students==0) return st.size();

        st.pop();
        que.remove();
    }

    return 0;
```



```
class Solution {
    public int timeRequiredToBuy(int[] tickets, int k) {
        Queue<Integer> que=new ArrayDeque<>();

        int n=tickets.length;

        for(int i=0; i<n; i++){
            if(i==k){
                que.add(-1 * tickets[i]);
            } else {
                que.add(tickets[i]);
            }
        }

        int time=0;
        while(que.size()>0){
            int front_ele=que.remove();

            if(front_ele==-1) return time+1;

            if(front_ele<0){
                front_ele++;
            } else {
                front_ele--;
            }

            if(front_ele!=0){
                que.add(front_ele);
            }

            time++;
        }
        return time;
    }
```

k = 2

2, 3, 2

time = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

, 1

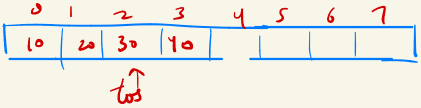
fe = -1 → 1

~~6~~ 5

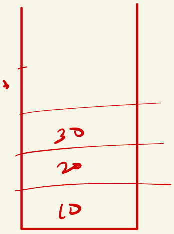
---

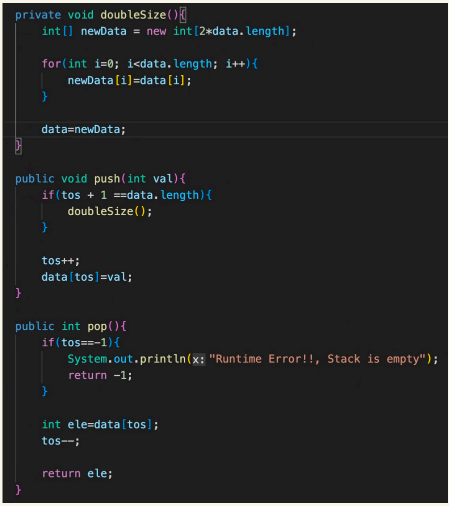
data ⇒

tos ↓

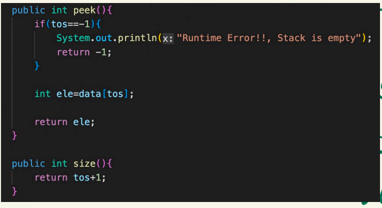


ele = 105





```
private void doubleSize(){
    int[] newData = new int[2*data.length];

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

    data=newData;
}

public void push(int val){
    if(tos + 1 ==data.length){
        doubleSize();
    }

    tos++;
    data[tos]=val;
}

public int pop(){
    if(tos==-1){
        System.out.println(x: "Runtime Error!!, Stack is empty");
        return -1;
    }

    int ele=data[tos];
    tos--;

    return ele;
}
```



```
public int peek(){
    if(tos==-1){
        System.out.println(x: "Runtime Error!!, Stack is empty");
        return -1;
    }

    int ele=data[tos];

    return ele;
}

public int size(){
    return tos+1;
}
```

✓ st.push(20)

✓ st.push(30)

✓ st.push(40)

st.peek(); → return data[tos]

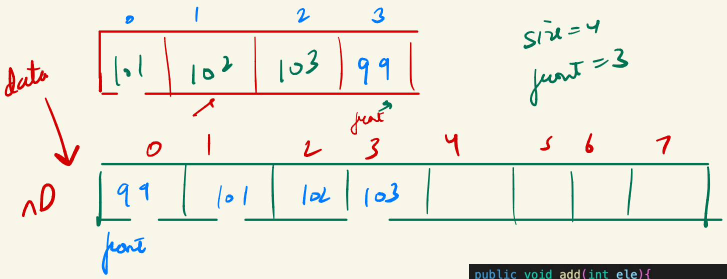
st.pop() ✓

st.push(75) ✓

st.push(95) ✓

st.push(105)

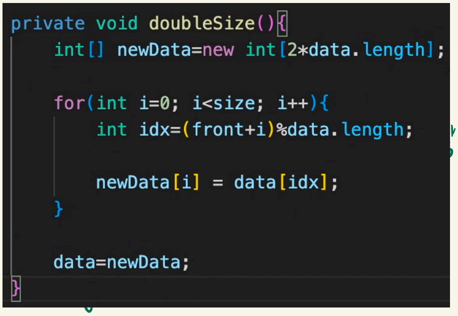
st.pop()



size=4
front=3

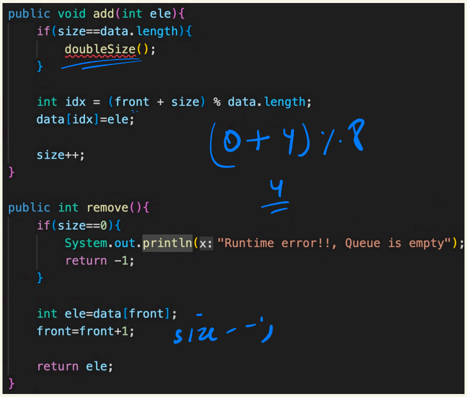
$i = \cancel{0} + \cancel{2}\ 3$

$idx = (3+\cancel{2}\ 3)\ \%\ 4 \Rightarrow \cancel{0} + 2\ ;$



```java
private void doubleSize(){
    int[] newData=new int[2*data.length];

    for(int i=0; i<size; i++){
        int idx=(front+i)%data.length;

        newData[i] = data[idx];
    }

    data=newData;
}
```

99, 101, 102, 103



```java
public void add(int ele){
    if(size==data.length){
        doubleSize();
    }

    int idx = (front + size) % data.length;
    data[idx]=ele;

    size++;
}

public int remove(){
    if(size==0){
        System.out.println(x: "Runtime error!!, Queue is empty");
        return -1;
    }

    int ele=data[front];
    front=front+1;

    return ele;
}
```

(0+4) % 8

4

size--;

add 57
✓ add 87
✓ add 54
✓ add 99
✓ peek -> return data(front)
✓ remove
✓ add 101
✓ remove
✓ remove
✓ add 102
✓ remove