

AUGUST 2022													
1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31											
M	T	W	T	F	S	S	M	T	W	T	F	S	S

July

30th Wk • 205-160

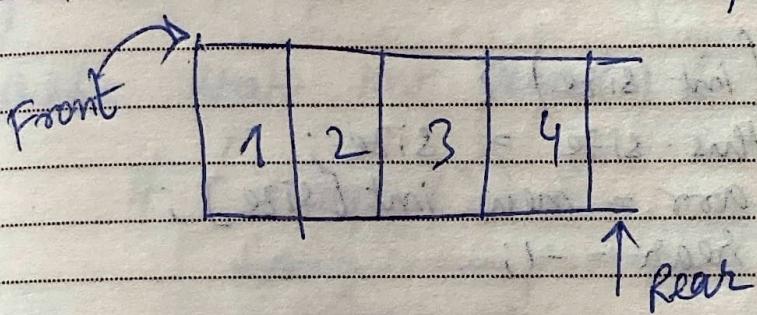
24

Sunday

In stack, we used to add elements one on top of the other. And we saw that it followed a LIFO structure.



However, in Queue we follow a front - Rear data structure. We add elements from the Rear.



- When we pop out elements from a Queue, we do that from the front. So, we pop out 1, then 2, ... so on.
- Here, in Queue we follow FIFO (First In First Out) structure.

Operations

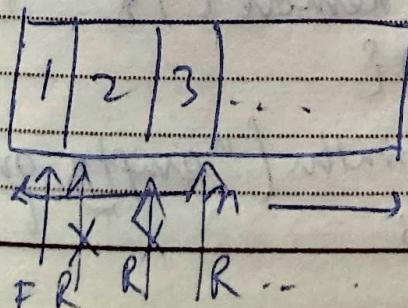
(Note: In array, add is O(1). But remove and peek is O(n). i.e., not optimised to implement Queue by array)

Enqueue → Add → adding elements

Dequeue → Remove → removing elements

Front → Peek → to view value of front elements

Implementation 1 (Queue using Array)



Size = n (fixed)

Initially, Front (F) = -1, R = -1

then F = 0, R = 0, then F = 0, R = 1, and so on.

25

July

31st Wk • 206-159

Monday

2022

215

MTWTFSSM	TWTFS
11 12 13 14 15 16 17	1 2 3 4 5 6 7
25 26 27 28 29 30 31	8 9 10 18 19 20 21
	22 23 24

* Queue Using Array *

```

8 Public class QueueB QueueB {
9     static class Queue {
10         static int arr[];
11         static int size;
12         static int rear = -1;
13
14         Queue (int size) {
15             this.size = size;
16             arr = new int [size];
17             rear = -1;
18         }
19
20         public static boolean isEmpty() {
21             return (rear == -1);
22         }
23
24         public static boolean isFull() {
25             return (rear == size - 1);
26         }
27
28         public static void add (int data) {
29             if (isFull ()) {
30                 System.out.println ("overflow");
31                 return;
32             }
33             arr[++rear] = data;
34         }
35
36         // O(n)
37         public static int remove () {
38             if (isEmpty ()) {
39                 System.out.println ("empty queue");
40                 return -1;
41             }
42         }
43     }
44 }

```

```
31 W T F S M T W I F  
int front = arr[0];  
for (int i = 0; i < rear; i++)  
{  
    arr[i] = arr[i+1];  
}  
rear--;  
return front;  
}
```

```
    public static int peek() {
```

if (isEmpty ()) {

```
isempty() {  
    System.out.println("empty queue");  
    return -1;  
}
```

return -i;

3

return arr[0];

3

```
public static void main (String args[]) {
```

frame q = new frame (5);

q.add('1');

g. add (2);
g. add (3);

```
System.out.println(q.remove());
```

```
System.out.println(q.peek());
```

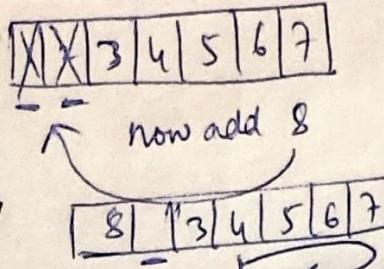
3

27

Circular Queue
July

31st Wk • 208-157

Wednesday



2022

217

JULY

11	12	13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31							

M T W T F S S M T W T F S

* * Circular Queue using array * *

8 Public class QueueB {

9 Static class Queue {

10 Static int arr[];

11 Static int size;

12 Static int front = -1;

13 Static int rear = -1;

14 Queue (int size) {

15 this.size = size;

16 arr = new int [size];

17 Public static boolean isEmpty() {

18 return (rear == -1 && front == -1);

19 Public static boolean isFull() {

20 return ((rear + 1) % size == front);

21 Public static void add (int data) {

22 If (isFull ()) {

23 System.out.println ("overflow");

24 return;

25 }

26 If it's the 1st element, if not full

27 If (front == -1) {

28 front = 0;

29 rear = (rear + 1) % size;

30 arr[rear] = data;

31

2022

AUGUST											
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31					
M	T	W	T	F	S	S	M	T	W	T	F

July

31st Wk • 209-156

28

Thursday

```

1     public static int remove() {
2         if (isEmpty())
3             System.out.println("empty queue");
4         return -1;
5     }
6     int res = arr[front];
7     if only 1 element is present
8         if (front == rear) {
9             front = rear = -1;
10        } else {
11            front = (front + 1) % size;
12        }
13        return res;
14    }
15    public static int peek() {
16        if (isEmpty())
17            System.out.println("empty queue");
18        return -1;
19    }
20    return arr[front];
21}
22
23    public static void main (String args[]) {
24        Queue q = new Queue(13);
25        q.add(1);
26        q.add(2);
27        q.add(3);
28        System.out.println(q.remove());
29        q.add(4);
30        System.out.println(q.remove());
31        while (!q.isEmpty()) {
32            System.out.println(q.remove());
33        }
34    }

```

29

July

31st Wk • 210-155

Friday

For linked list also,
for add, remove, peek $\rightarrow O(1)$.

2022

219

SUN	MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7	8	9	10	11	12	13	14
25	26	27	28	29	30	31							
							M	T	W	T	F	S	S

* * Queue using Linked List * *

```

8 Public class QueueB {
9     static class Node {
10         int data;
11         Node next;
12         Node (int data) {
13             this.data = data;
14             next = null;
15         }
16     }
17
18     static class Queue {
19         static Node head = null;
20         static Node tail = null;
21
22         Public static boolean isEmpty () {
23             return (head == null && tail == null);
24
25         Public static void add (int data) {
26             Node newNode = new Node (data);
27             if (isEmpty ()) {
28                 tail = head = newNode;
29             } else {
30                 tail.next = newNode;
31                 tail = newNode;
32             }
33         }
34
35         Public static int remove () {
36             if (!isEmpty ()) {
37                 System.out.println ("empty queue");
38                 return -1;
39             }
40         }
41     }
42 }
```

2022

AUGUST	
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30
31	

31st Wk • 211-154

July

30

Saturday

```

int front = head.data;
8 //single node
if (head == tail) {
9   tail = null;
10  head = head.next;
11  return front;
12}

11 public static int peek () {
12   if (isEmpty ()) {
13     System.out.println ("empty queue");
14     return -1;
15   }
16   return head.data;
17 }

18 public static void main (String args[]) {
19   Queue q = new Queue ();
20   q.add (1);
21   q.add (2);
22   q.add (3);
23   q.add (4);
24   q.add (5);

25   while (! q.isEmpty ()) {
26     System.out.println (q.peek ());
27     q.remove ();
28   }
29 }

```

31

July

31st Wk • 212-153

Sunday

2022

22

JULY											
SUN	MON	TUE	WED	THU	FRI	SAT	1	2	3	4	5
	11	12	13	14	15	16	17	18	19	20	21
	25	26	27	28	29	30	31				
M	T	W	T	F	S	S	M	T	W	T	F

Java Collection Framework - Queue

8 import java.util.*;

9 Public class QueueB {

10 Public static void main (String args[]){

// Queue implementation using JCF
 Queue <Integer> q = new ArrayDeque();

q.add(1);
 q.add(2);
 q.add(3);
 q.add(4);
 q.add(5);

2 while (! q.isEmpty()) {

3 System.out.println (q.peek());

4 q.remove();

5 }

6 * * Queue using 2 stacks * * push $\rightarrow O(n)$
 import java.util.*;

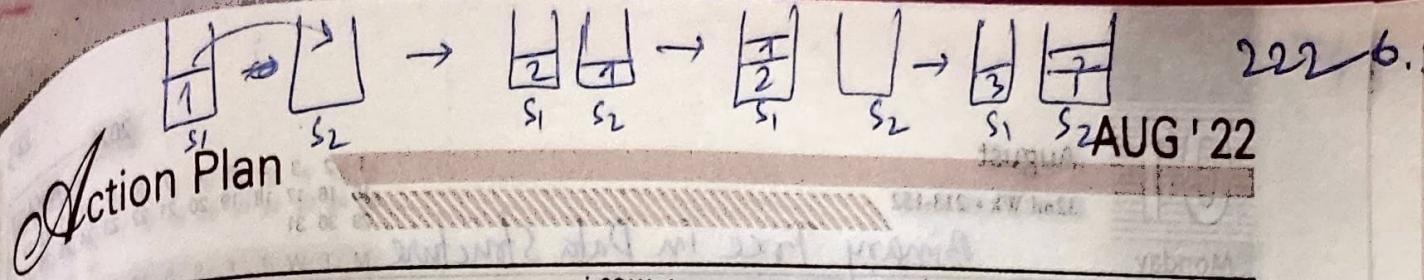
pop $\rightarrow O(n)$

7 Public class QueueB {

8 Static class Queue {

9 Static Stack <Integer> s1 = new Stack<>();

10 Static Stack <Integer> s2 = new Stack<>();



2226.

Action Plan

01 Mon	02 Tue	03 Wed	04 Thu
	Public static boolean isEmpty () { return s1.isEmpty(); }		
05 Fri	Public static void add (int data) { while (!s1.isEmpty ()) { s2.push (s2.pop()); // Pop from s2 & Push to s1. } s1.push (data); while (!s2.isEmpty ()) { s1.push (s2.pop()); // Pop from s1 & Push to s2. }	06 Sat	07 Sun
06 Tue	10 Wed	11 Thu	12 Fri
13 Sat	14 Sun	15 Mon	16 Tue
17 Wed	18 Thu	19 Fri	20 Sat
21 Sun	22 Mon	23 Tue	24 Wed
25 Thu	26 Fri	27 Sat	28 Sun
29 Mon	30 Tue	31 Wed	

AUG '22

```

02 Tue: Public static boolean isEmpty () {
    return s1.isEmpty();
}

06 Sat: Public static void add (int data) {
    while (!s1.isEmpty ()) {
        s2.push (s2.pop()); // Pop from s2 & Push to s1.
    }
    s1.push (data);
    while (!s2.isEmpty ()) {
        s1.push (s2.pop()); // Pop from s1 & Push to s2.
    }
}

10 Wed: 
11 Thu: 
12 Fri: 

14 Sun: 
15 Mon: 
16 Tue: 

18 Thu: Public static int remove () {
    return s1.pop();
}

22 Mon: Public static int peek () {
    return s1.peek();
}

26 Fri: Public static void main (String args[]) {
    Queue q = new Queue();
    q.add(1);
    q.add(2);
    q.add(3);
}

30 Tue: while (!q.isEmpty ()) {
    System.out.println (q.peek());
    q.remove();
}

```