

Name : [AZHAR Ali]

CMS : [023-23-0314]

SEC : [A]

LAB 4

DSA LAB



1. **Stack using array:** Understand provided code and implement all required methods in Stack. Stack Code is given below:

```
1 class StackUsingArray
2 {
3     private int arr[];
4     private int top;
5     private int capacity;
6     int size;
7
8     StackUsingArray(int cap)
9     {
10    capacity=cap;
11    arr = new int[capacity];
12    top = -1;
13    size=0;
14    }
15
16    public void push(int x)
17    {
18        if(!isFull()){
19            arr[++top]=x;
20            size++;
21            System.out.println("Inserted "+x);
22        }
23        else{
24            System.out.println("Stack OverFlow");
25        }
26    }
27
28    // Utility function to pop top element from the stack and
29    //check for stack underflow
30    public int pop()
31    {
32        if(isEmpty()){
33            System.out.println("Stack is Empty");
34            return 0;
35        }
36        else {
37            System.out.println("Removed "+arr[top]);
38            size--;
39            return arr[top--];
40        }
41    }
42
43    // Utility function to return top element in a stack
44    public int top()
45    {
46        return arr[top];
47    }
48    // Utility function to return the size of the stack
49    public int size()
50    {
51        return size;
52    }
53    // Utility function to check if the stack is empty or not
54    public Boolean isEmpty()
55    {
56        return size==0;
57    }
58    // Utility function to check if the stack is full or not
59    public Boolean isFull()
60    {
61        return (size==capacity);
62    }
63    public static void main (String[] args)
64    {
65        StackUsingArray stack = new StackUsingArray(3);
66        stack.push(1); // Inserting 1 in the stack
67        stack.push(2); // Inserting 2 in the stack
68        stack.pop(); // removing the top 2
69        stack.pop(); // removing the top 1
70        stack.push(3); // Inserting 3 in the stack
71        System.out.println("Top element is: " + stack.top());
72        System.out.println("Stack size is " + stack.size());
73        stack.pop(); // removing the top 3
74        // check if stack is empty
75        if (stack.isEmpty())
76            System.out.println("Stack Is Empty");
77        else
78            System.out.println("Stack Is Not Empty");
79    }
80 }
```

```
arali/.config/Code/U
Inserted 1
Inserted 2
Removed 2
Removed 1
Inserted 3
Top element is: 3
Stack size is 1
Removed 3
Stack Is Empty
```

```
○ azharali@fedora:~/Se
```

2. **Stack using Linked list:** Understand provided code and implement all required methods in Stack. Stack Code is given below:

```
1 // Define Node class
2 class Node {
3     int data;
4     Node next;
5
6     public Node() {
7         this.data = 0;
8         this.next = null;
9     }
10
11     public Node(int data) {
12         this.data = data;
13         this.next = null;
14     }
15 }
16
17 // Implement Stack using Linked List
18 class StackUsingLinkedList {
19     private Node top;
20
21     public StackUsingLinkedList() {
22         this.top = null;
23     }
24
25     // Utility function to add an element x in the stack
26     public int push(int x) {
27         Node newNode = new Node(x);
28         newNode.next = top;
29         top = newNode;
30         return x;
31     }
32
33     // Utility function to check if the stack is empty or not
34     public boolean isEmpty() {
35         if (top == null) {
36             return true;
37         } else {
38             return false;
39         }
40     }
41
42     // Utility function to return top element in a stack
43     public int top() {
44         if (top == null) {
45             return -1;
46         }
47         return top.data;
48     }
49
50     // Utility function to remove top element from the stack
51     public int pop() {
52         if (top == null) {
53             return 0;
54         }
55         else {
56             int k = top.data;
57             top = top.next;
58             return k;
59         }
60     }
61 }
62 }
```

```

62 }
63 // Test Stack implementa
64 class StackImpl {
65     public static void main(String[] args) {
66         StackUsingLinkedList stack = new StackUsingLinkedList();
67         System.out.println("inserted "+stack.push(1));
68         System.out.println("inserted "+stack.push(2));
69         System.out.println("inserted "+stack.push(3));
70
71
72         if(stack.top()!=-1){
73             System.out.println("Top element is " + stack.top());}
74         else{
75             System.out.println("Stack is Empty");
76         }
77
78         int d1 = stack.pop();
79         if(d1==0){
80             System.out.println("Stack Empty");
81         }
82         else{
83             System.out.println("Removed "+d1);
84         }
85         int d2 = stack.pop();
86         if(d2==0){
87             System.out.println("Stack Empty");
88         }
89         else{
90             System.out.println("Removed "+d2);
91         }
92         int d3 = stack.pop();
93         if(d3==0){
94             System.out.println("Stack Empty");
95         }
96         else{
97             System.out.println("Removed "+d3);
98         }
99         // Test pop on empty stack
100         int d4 = stack.pop();
101         if(d4==0){
102             System.out.println("Stack Empty");
103         }
104         else{
105             System.out.println("Removed "+d4);
106         }
107
108         if(stack.isEmpty()){
109             System.out.println("Stack Empty");
110         }
111         else{
112             System.out.println("Stack is not empty");
113         }
114
115     }
116 }
117 }

```

```

Inserted 1
Inserted 2
Inserted 3
Inserted 4
Front element is 1
Removed 1
Removed 2
Removed 3
Removed 4
Queue is empty
azharali@fedora:~/Semester 3/DSA LAB/Lab 4$

```

3. Queue using array: Understand provided code and implement all required methods in Queue. Queue Code is given below:

```
1 class QueueUsingArray
2 {
3     private int arr[];
4     private int front;
5     private int rear;
6     private int count;
7     // Constructor to initialize queue
8     QueueUsingArray(int cap)
9     {
10        arr = new int[cap];
11        front = 0;
12        rear = 0;
13        count = 0;
14    }
15    // Utility function to remove front element from the queue and check for Queue Underflow
16    public void dequeue()
17    {
18        if(count==0){
19            System.out.println("Queue Is Empty");
20        }
21        else{
22            System.out.println("Removed "+arr[front]);
23            count--;
24            arr[(front++)%(arr.length)]=0;
25        }
26    }
27
28    public void enqueue(int item)
29    {
30        System.out.println("Inserted "+item);
31        count++;
32        arr[(rear++)%(arr.length)]=item;
33    }
34
35    public int peek()
36    {
37        return arr[front];
38    }
39    // Utility function to return the size of the queue
40    public int size()
41    {
42        return count;
43    }
44    // Utility function to check if the queue is empty or not
45    public Boolean isEmpty()
46    {
47        return count<arr.length;
48    }
49    // Utility function to check if the queue is empty or not
50    public Boolean isFull()
51    {
52        return count==arr.length;
53    }
54 }
55
56
57 class Main
58 {
59     // main function
60     public static void main (String[] args)
61     {
62         // create a queue of capacity 5
63         QueueUsingArray q = new QueueUsingArray(5);
64         q.enqueue(1);
65         q.enqueue(2);
66         q.enqueue(3);
67         System.out.println("Front element is: " + q.peek());
68         q.dequeue();
69         System.out.println("Front element is: " + q.peek());
70         System.out.println("Queue size is " + q.size());
71         q.dequeue();
72         System.out.println("Front element is: " + q.peek());
73         q.dequeue();
74         if (q.isEmpty())
75             System.out.println("Queue Is Empty");
76         else
77             System.out.println("Queue Is Not Empty");
78     }
79 }
```

```
Inserted 1
Inserted 2
Inserted 3
Front element is: 1
Removed 1
Front element is: 2
Queue size is 2
Removed 2
Front element is: 3
Removed 3
Queue Is Empty
```

azharali@fedora:~/Semester 3/DS

4. Queue using Linked list: Understand provided code and implement all required methods in Queue. Queue Code is given below:

```
1 class QueueUsingLinkedList{
2     Node rear = null, front = null;
3
4     // Utility function to remove front element from the queue and check for Queue Underflow
5 public int dequeue()
6 { if(front==null){
7     System.out.println("Queue UnderFlow");
8     return -1;
9 }
10    int k = front.data;
11    front=front.next;
12    System.out.println("Removed "+k);
13    return k;
14 }
15 // Utility function to add an item in the queue
16 public void enqueue(int item)
17 {
18     Node temp = new Node(item);
19     if(front==null && rear==null){
20         rear=temp;
21         front=temp;
22     }
23     else{
24
25         rear.next=temp;
26         rear=temp;
27     }
28     System.out.println("Inserted "+item);
29 }
30 // Utility function to return top element in a queue
31 public int peek()
32 {
33     if(front==null){
34         System.out.println("QUEUE EMPTY");
35         return 0;
36     }
37     else{
38         return front.data;
39     }
40 }
41 // Utility function to check if the queue is empty or not
42 public boolean isEmpty()
43 {
44     return front==null;
45 }
46
47 public static void main(String[] args)
48 {
49     QueueUsingLinkedList q = new QueueUsingLinkedList();
50     q.enqueue(1);
51     q.enqueue(2);
52     q.enqueue(3);
53     q.enqueue(4);
54     System.out.printf("Front element is %d\n", q.peek());
55     q.dequeue();
56     q.dequeue();
57     q.dequeue();
58     q.dequeue();
59     if (q.isEmpty()) {
60         System.out.println("Queue is empty");
61     } else {
62         System.out.println("Queue is not empty");
63     }
64 }
65 }
66
67 class Node
68 {
69     int data;
70     Node next;
71
72     public Node(int data)
73     {
74
75         this.data = data;
76         this.next = null;
77     }
78 }
```

```
Inserted 1
Inserted 2
Inserted 3
Inserted 4
Front element is 1
Removed 1
Removed 2
Removed 3
Removed 4
Queue is empty
```

azharali@fedora:~/Semes

5. **Queue using two Stacks:** Understand provided code and implement all required methods in Queue Class. Sample Code is given below:

```
1 class QueueUsingTwoStacks {
2     StackUsingLinkedList s1, s2;
3
4     QueueUsingTwoStacks() {
5         s1 = new StackUsingLinkedList();
6         s2 = new StackUsingLinkedList();
7     }
8
9     public void enqueue(int data) {
10         s1.push(data);
11     }
12
13     public int dequeue() {
14         if (s2.isEmpty()) {
15             while (!s1.isEmpty()) {
16                 s2.push(s1.pop());
17             }
18         }
19         if (s2.isEmpty()) {
20             throw new RuntimeException("Queue is empty");
21         }
22         return s2.pop();
23     }
24
25     public static void main(String[] args) {
26         int[] keys = {1, 2, 3, 4, 5};
27         QueueUsingTwoStacks q = new QueueUsingTwoStacks();
28
29         for (int key : keys) {
30             System.out.println("Inserted "+key);
31             q.enqueue(key);
32         }
33
34         System.out.println("Removed "+q.dequeue());
35         System.out.println("Removed "+q.dequeue());
36     }
37 }
```

```
onMessages -cp /ho
Inserted 1
Inserted 2
Inserted 3
Inserted 4
Inserted 5
Removed 1
Removed 2
○ azharali@fedora:~.
```

6. think about the inverse of task 05 (Stack using queue) and implement all the required methods.

```
1 public class StackUsingQueue {
2     private QueueUsingLinkedList Q1, Q2;
3
4     public StackUsingQueue() {
5         Q1 = new QueueUsingLinkedList();
6         Q2 = new QueueUsingLinkedList();
7     }
8
9     public void Push(int data) {
10         Q1.enqueue(data);
11     }
12
13     public int Pop() {
14         if (Q1.isEmpty()) {
15             throw new RuntimeException("Stack is empty");
16         }
17
18         while (Q1.size() > 1) {
19             Q2.enqueue(Q1.dequeue());
20         }
21
22         int topElement = Q1.dequeue();
23
24         QueueUsingLinkedList temp = Q1;
25         Q1 = Q2;
26         Q2 = temp;
27
28         return topElement;
29     }
30
31     public static void main(String[] args) {
32         StackUsingQueue stack = new StackUsingQueue();
33
34         stack.Push(1);
35         stack.Push(2);
36         stack.Push(3);
37         stack.Push(4);
38         stack.Push(5);
39
40         System.out.println(stack.Pop()); // 5
41         System.out.println(stack.Pop()); // 4
42         System.out.println(stack.Pop()); // 3
43         System.out.println(stack.Pop()); // 2
44         System.out.println(stack.Pop()); // 1
45
46     }
47 }
48
49
50
51 }
```

```
home/azharali/.config/Code/User/workspaceStorage/9
5
4
3
2
1
azharali@fedora:~/Semester 3/DSA LAB/Lab 4$
```


