Data Structures & Applications
Fall 2020
Lab 03 – Stacks, Queues

Instructor: Saif Hassan **Date: 5th October 2020**

Instructions:

- At the end of this Lab, you will have to submit all files on LMS.
- File format should be **.zip/.rar** file containing required **.java** files and additional if required.
- File Name should be your **CMSID_Name_Lab03.zip.**
- Create a project named lab03_dsa and perform following tasks.

Stack

Note: Keep this code with you till the course ends.

Task 01: (Stack using array)

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

```
1.  import java.util.*;
2.
3.  class Stack
4.  {
5.      private int arr[];
6.      private int top;
7.      private int capacity;
8.
9.      // Constructor to initialize stack
10.     Stack(int size)
11.     {
12.         arr = new int[size];
13.         capacity = size;
14.         top = -1;
15.     }
16.
17.     // Utility function to add an element x in the stack and check for stack overflow
18.     public void push(int x)
19.     {
20.         // Write your code here
21.     }
22.
23.     // Utility function to pop top element from the stack and check for stack underflow

24.     public int pop()
25.     {
26.         // Write your code here
27.     }
28.
```

```
29.     // Utility function to return top element in a stack
30.     public int peek()
31.     {
32.         // Write your code here
33.     }
34.
35.     // Utility function to return the size of the stack
36.     public int size()
37.     {
38.         // Write your code here
39.     }
40.
41.     // Utility function to check if the stack is empty or not
42.     public Boolean isEmpty()
43.     {
44.         // Write your code here
45.     }
46.
47.     // Utility function to check if the stack is full or not
48.     public Boolean isFull()
49.     {
50.         // Write your code here
51.     }
52.
53.     public static void main (String[] args)
54.     {
55.         Stack stack = new Stack(3);
56.
57.         stack.push(1);      // Inserting 1 in the stack
58.         stack.push(2);      // Inserting 2 in the stack
59.
60.         stack.pop();        // removing the top 2
61.         stack.pop();        // removing the top 1
62.
63.         stack.push(3);      // Inserting 3 in the stack
64.
65.         System.out.println("Top element is: " + stack.peek());
66.         System.out.println("Stack size is " + stack.size());
67.
68.         stack.pop();        // removing the top 3
69.
70.         // check if stack is empty
71.         if (stack.isEmpty())
72.             System.out.println("Stack Is Empty");
73.         else
74.             System.out.println("Stack Is Not Empty");
75.     }
76. }
```

After implementing all the methods, run the code. Your output should be like as follows:

Output

```
Inserting 1
Inserting 2
Removing 2
Removing 1
Inserting 3
Top element is: 3
Stack size is 1
Removing 3
Stack Is Empty
```

Task 02:(Stack using Linked list)

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

```
1.  import java.util.*;
2.
3.  // A linked list node
4.  class Node
5.  {
6.      int data;       // integer data
7.      Node next;      // pointer to the next node
8.  };
9.
10. class Stack
11. {
12.     private Node top;
13.
14.     public Stack() {
15.         this.top = null;
16.     }
17.
18.     // Utility function to add an element x in the stack
19.     public void push(int x) // insert at the beginning
20.     {
21.         // Write your code here
22.     }
23.
24.     // Utility function to check if the stack is empty or not
25.     public boolean isEmpty()
26.     {
27.         // Write your code here
28.     }
29.
30.     // Utility function to return top element in a stack
31.     public int peek()
32.     {
33.         // Write your code here
34.     }
35.
36.     // Utility function to pop top element from the stack and check for Stack underflow
37.     public void pop() // remove at the beginning
38.     {
39.         // Write your code here
40.     }
```

```
41. }
42.
43. class StackImpl
44. {
45.     public static void main(String[] args)
46.     {
47.         Stack stack = new Stack();
48.
49.         stack.push(1);
50.         stack.push(2);
51.         stack.push(3);
52.
53.         System.out.println("Top element is " +  stack.peek());
54.
55.         stack.pop();
56.         stack.pop();
57.         stack.pop();
58.
59.         if (stack.isEmpty()) {
60.             System.out.print("Stack is empty");
61.         } else {
62.             System.out.print("Stack is not empty");
63.         }
64.     }
65. }
```

After implementing all the methods, run the code. Your output should be like as follows:

Output

```
Inserting 1
Inserting 2
Inserting 3
Top element is 3
Removing 3
Removing 2
Removing 1
Stack is empty
```

Queue

Note: Keep this code with you till the course ends.

Task 03: (Queue using array)

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

```
1.  import java.util.*;
2.
3.  // Class for queue
4.  class Queue
5.  {
6.      private int arr[];
7.      private int front;
8.      private int rear;
9.      private int capacity;
10.     private int count;
11.
12.     // Constructor to initialize queue
13.     Queue(int size)
14.     {
15.         arr = new int[size];
16.         capacity = size;
17.         front = 0;
18.         rear = -1;
19.         count = 0;
20.     }
21.
22.     // Utility function to remove front element from the queue and check for Queue Unde
    rflow
23.     public void dequeue()
24.     {
25.         // Write your code here
26.     }
27.
28.     // Utility function to add an item to the queue and check for queue overflow
29.     public void enqueue(int item)
30.     {
31.         // Write your code here
32.     }
33.
34.     // Utility function to return front element in the queue and check for Queue Underf
    low
35.     public int peek()
36.     {
37.         // Write your code here
38.     }
39.
40.     // Utility function to return the size of the queue
41.     public int size()
42.     {
43.         // Write your code here
44.     }
45.
46.     // Utility function to check if the queue is empty or not
47.     public Boolean isEmpty()
48.     {
49.         // Write your code here
```

```
    }

    // Utility function to check if the queue is empty or not
    public Boolean isFull()
    {
        // Write your code here
    }
}

class Main
{
    // main function
    public static void main (String[] args)
    {
        // create a queue of capacity 5
        Queue q = new Queue(5);

        q.enqueue(1);
        q.enqueue(2);
        q.enqueue(3);

        System.out.println("Front element is: " + q.peek());
        q.dequeue();
        System.out.println("Front element is: " + q.peek());

        System.out.println("Queue size is " + q.size());

        q.dequeue();
        q.dequeue();

        if (q.isEmpty())
            System.out.println("Queue Is Empty");
        else
            System.out.println("Queue Is Not Empty");
    }
}
```

After implementing all the methods, run the code. Your output should be like as follows:

Output

```
Inserting 1
Inserting 2
Inserting 3
Front element is: 1
Removing 1
Front element is: 2
Queue size is 2
Removing 2
Removing 3
Queue Is Empty
```

Task 04:(Queue using Linked list)

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

```
// A linked list node
class Node
{
    int data;       // integer data
    Node next;      // pointer to the next node

    public Node(int data)
    {
        // set the data in allocated node and return the node
        this.data = data;
        this.next = null;
    }
}

class Queue
{
    private static Node rear = null, front = null;

    // Utility function to remove front element from the queue and check for Queue Underflow
    public static int dequeue()     // delete at the beginning
    {
        // Write your code here
    }

    // Utility function to add an item in the queue
    public static void enqueue(int item)    // insertion at the end
    {
        // Write your code here
    }

    // Utility function to return top element in a queue
    public static int peek()
    {
        // Write your code here
    }

    // Utility function to check if the queue is empty or not
    public static boolean isEmpty()
    {
        // Write your code here
    }
}

class Main {
    public static void main(String[] args)
    {
        Queue q = new Queue();
        q.enqueue(1);
        q.enqueue(2);
        q.enqueue(3);
        q.enqueue(4);

        System.out.printf("Front element is %d\n", q.peek());

        q.dequeue();
        q.dequeue();
```

```
57.         q.dequeue();
58.         q.dequeue();
59.
60.         if (q.isEmpty()) {
61.             System.out.print("Queue is empty");
62.         } else {
63.             System.out.print("Queue is not empty");
64.         }
65.     }
66. }
```

After implementing all the methods, run the code. Your output should be like as follows:

Output

```
Inserting 1
Inserting 2
Inserting 3
Inserting 4
Front element is 1
Removing 1
Removing 2
Removing 3
Removing 4
Queue is empty
```

Queue using two Stacks

Question 5: Understand provided code and implement all required methods in Queue Class. Sample Code is given below:

```
1.  import java.util.*;
2.
3.  // Implement Queue using two stacks
4.  class Queue {
5.      private Stack s1, s2;
6.
7.      // Constructor
8.      Queue() {
9.          s1 = new Stack();
10.         s2 = new Stack();
11.     }
12.
13.     // Enqueue an item to the queue
14.     public void enqueue(int data)
15.     {
16.         // Write your code here
17.     }
18.
19.     // Dequeue an item from the queue
20.     public int dequeue()
21.     {
22.         // Write your code here
23.     }
24.
25.     public static void main(String[] args) {
26.         int[] keys = { 1, 2, 3, 4, 5 };
27.         Queue q = new Queue();
28.
29.         // insert above keys
30.         for (int key : keys) {
31.             q.enqueue(key);
32.         }
33.
34.         System.out.println(q.dequeue());    // print 1
35.         System.out.println(q.dequeue());    // print 2
36.     }
37. }
```

`After implementing all the methods, run the code.`

Bonus Task: Think about the inverse of task 05 (Stack using queue)

Specifications, notes, and hints

Your program needs to meet the following specifications:

- Submit all files `LinkedList`.java and additional files if applicable.
- When commenting your code use `Javadoc` style comments at the beginning of each method.
- Put comments at the top of the file (`Java` File) with your name, S_ID, S_Name, date and course, and a short (one or two line) description of what the program does. Make sure your code runs on machine.
- Submit your `source code` files via the classroom by the due date (remember the course syllabus for the late policy).