Stack and Queue implementation using linked lists:

Stack:

Code:

```java
import java.util.Scanner;

/**
 * Stack_implementation_using_Linked_List
 */
public class Stack_implementation_using_Linked_List {
    public static class Node {
        int data;
        Node next = null ;

        Node(int value){
            this.data = value;
        }

        @Override
        public String toString() {
            return "" + this.data;
        }

    }

    public static class Stack{
        int lenght =0;
        Node top;



        @Override
    public String toString() {
        String temp = "";

        Node current = this.top;
        while (current != null) {
            temp = temp + current.data+ " -- " ;
            current =current.next;
        }

        return temp + "Stack End";
    }
```

```
    public void push(int i){
        Node temp = new Node(i);
        if (this.top==null) {
            this.top=temp;
            this.lenght++;
            return ;
        }
        temp.next=this.top;
        top = temp;
        this.lenght++;
    }

    public int pop(){
        if(this.top == null){
            System.out.println("Stack Underflow");
            return -1;
        }
        Node temp = this.top;
        top = top.next;
        this.lenght--;
        return temp.data;
    }

    public void peek(){
        System.out.println("top = "+this.top);
    }









    }
    public static void main(String[] args) {
       Stack a = new Stack();
       Scanner sc = new Scanner(System.in);
       boolean l = true;
       System.out.println("Empty stack initailized perform your operations :");
       while (l) {
            System.out.println();
           System.out.print("1) Push ");
           System.out.print("2) Pop ");
           System.out.print("3) Peek ");
           System.out.print("4) Is Stack empty ");
           System.out.print("5) Display ");
           System.out.print("6) Exit \n");
           System.out.print("Enter your choice : ");
        int ch = sc.nextInt();
        System.out.println("OUTPUT:");
        switch (ch) {
```

```
            case 1:
                System.out.print("Enter int to push: ");
                int i = sc.nextInt();
                a.push(i);
                break;
            case 2:
                int poped = a.pop();
                System.out.println(poped + " got removed");
                break;
            case 3:
                a.peek();
                break;
            case 4:
                if (a.lenght == 0) {
                    System.out.println("Stack is empty");
                } else {
                    System.out.println("Stack is not empty");
                }
                break;

            case 6:
                System.out.println("Bye!");
                l=false;
                break;

            case 5:
                System.out.println(a);
                break;
            default:
                System.out.println("Invalid choice");
         }

       }
       sc.close();
    }

}
```

Output:

```
Empty stack initailized perform your operations :

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 4
OUTPUT:
Stack is empty

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 3
OUTPUT:
top = null

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 2
OUTPUT:
Stack Underflow
-1 got removed

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 1
OUTPUT:
Enter int to push: 10

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 3
OUTPUT:
top = 10

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 4
OUTPUT:
Stack is not empty

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 5
OUTPUT:
10 -- Stack End

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 1
OUTPUT:
Enter int to push: 190

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 5
OUTPUT:
190 -- 10 -- Stack End

1) Push 2) Pop 3) Peek 4) Is Stack empty 5) Display 6) Exit
Enter your choice : 2
OUTPUT:
190 got removed
```

Queue:

Code:

```
import java.util.Scanner;

public class Queue_implementation_using_Linked_List {
    public static void main(String[] args) {
         Queue a = new Queue();
       Scanner sc = new Scanner(System.in);
       boolean l = true;
       System.out.println("Empty Queue initailized perform your operations :");
       while (l) {
            System.out.println();
           System.out.print("1) Enqueue ");
           System.out.print("2) Dequeue ");
           System.out.print("3) Peek ");
           System.out.print("4) Is Queue empty ");
           System.out.print("5) Display ");
           System.out.print("6) Exit \n");
           System.out.print("Enter your choice : ");
        int ch = sc.nextInt();
        System.out.println("OUTPUT:");
        switch (ch) {
            case 1:
                System.out.print("Enter int to Enqueue: ");
                int i = sc.nextInt();
                a.Enqueue(i);
                break;
            case 2:
                int Dequeueed = a.Dequeue();
                System.out.println(Dequeueed + " got removed");
                break;
            case 3:
                a.peek();
                break;
            case 4:
            System.out.println(a.length != 0 ? "The Queue is Not Empty":"The Queue is
Empty");
                break;

            case 6:
                System.out.println("Bye!");
                l=false;
                break;

            case 5:
                System.out.println(a);
                break;
            default:
                System.out.println("Invalid choice");
        }
```

```
    }
    sc.close();
}

public static class Queue {
    Node top = null;
    Node bottom = null;
    int length = 0;


    public static class Node {
        int data;
        Node next;

        Node(int a){
            this.data = a;
            this.next = null;
        }

    }

    public void Enqueue(int i ){
        Node a = new Node(i);

        if (this.top == null) {
            top = a;
            bottom = a;
            this.length++;
            return;
        }

        this.bottom.next =a;
        this.bottom =a;
        this.length++;

    }

    public int Dequeue(){
        if(top == null){
            System.out.println("INVALID INDEX QUEUE UNDERFLOW");
            return -1;
        }
        else if (top == bottom) {
            Node temp = this.top;
            this.top = this.bottom = null;
            this.length--;
            return temp.data;
        }
```

```
            Node temp = this.top;
            this.top = this.top.next;
            this.length--;
            return temp.data;

        }

        public void peek(){

            if (this.top != null) {
                System.out.println(this.top.data);
            } else {
                System.out.println("Top doesn't exist. The Queue is empty.");
            }

        }

        public String toString() {
            String temp = "Queue Start -- ";

            Node current = this.top;
            while (current != null) {
                temp = temp + current.data+ " -- " ;
                current =current.next;
            }

            return temp + "Queue End";
        }



    }


}
```

Output:

```
Empty Queue initailized perform your operations :

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 5
OUTPUT:
Queue Start -- Queue End

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 4
OUTPUT:
The Queue is Empty

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 3
OUTPUT:
Top doesn't exist. The Queue is empty.

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 2
OUTPUT:
INVALID INDEX QUEUE UNDERFLOW
-1 got removed

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 1
OUTPUT:
Enter int to Enqueue: 100

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 1
OUTPUT:
Enter int to Enqueue: 1000

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 3
OUTPUT:
100

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice :
4
OUTPUT:
The Queue is Not Empty

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 5
OUTPUT:
Queue Start -- 100 -- 1000 -- Queue End

1) Enqueue 2) Dequeue 3) Peek 4) Is Queue empty 5) Display 6) Exit
Enter your choice : 2
OUTPUT:
100 got removed
```