**Implementing Stack using two Queues**

Making the push operation costly

Consider 2 queues q1 and q2. This method makes sure that newly entered element is always at the front of ‘q1’, so that pop operation just dequeues from ‘q1’. ‘q2’ is used to put every new element at front of ‘q1’.

Push Operation

1. Enqueue x to queue2.
2. One by one dequeue everything from from queue1 and enqueue into queue2.
3. Swap the names of queue1 and queue2.

Pop Operation

Dequeue an item from queue1 and return it.

public class StackUsingTwoQueues

{

Queue<Integer> queue1;

Queue<Integer> queue2;

StackUsingTwoQueues()

{

queue1=new LinkedList<Integer>();

queue2=new LinkedList<Integer>();

}

public void push(int i)

{

if(queue1.size()==0)

queue1.add(i);

else

{

int sizeOfQueue1 = queue1.size();

queue2.add(i);

for(int j=0 ; j<sizeOfQueue1 ; j++)

queue2.add(queue1.remove());

Queue<Integer> q = new LinkedList<Integer>();

q=queue1;

         queue1 = queue2;

         queue2 = q;

}

}

public int pop()

{

if(queue1.size()==0)

System.out.println("Underflow ");

return queue1.remove();

}

public static void main(String[] args)

{

StackUsingTwoQueues stack = new StackUsingTwoQueues();

stack.push(20);

stack.push(40);

stack.push(70);

stack.push(50);

stack.push(90);

stack.push(110);

stack.push(30);

System.out.println("Removed element : "+ stack.pop());

stack.push(170);

System.out.println("Removed element : "+ stack.pop());

}

}

Output:-

Element removed from LinkedList: 75

Removed element : 30

Removed element : 170

**Implementing Stack Using Single Queue**

The idea is to keep newly inserted element always at front, keeping order of previous elements same. Below are complete steps.

1. Enqueue x to queue.
2. One by one dequeue all items from queue and enqueue them.

public class stack

{

Queue<Integer> q = new LinkedList<Integer>();

void push(int val)

{

int size = q.size();

q.add(val);

for (int i = 0; i < size; i++)

{

int x = q.remove();

q.add(x);

}

}

int pop()

{

if (q.size()==0)

{

System.out.println(“Underflow”);

return -1;

}

int x = q.remove();

return x;

}

public static void main(String[] args)

{

stack s = new stack();

s.push(10);

s.push(20);

System.out.println("Top element :" + s.peek());

s.pop();

s.push(30);

s.pop();

System.out.println("Top element :" + s.peek());

}

}

Output:- 20

10