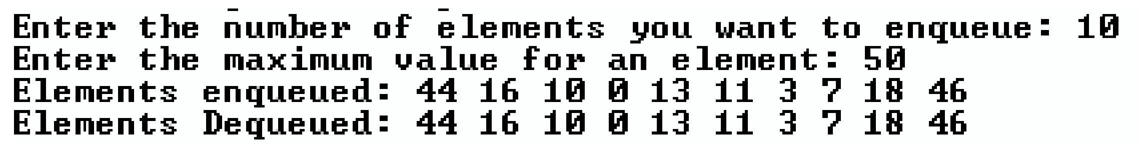
```java
import java.util.*;

// implementing a Queue as a Doubly Linked List


class Node{

    private int data;
    private Node nextNodePtr;
    private Node prevNodePtr;

    public Node(){}

    public void setData(int d){
        data = d;
    }

    public int getData(){
        return data;
    }

    public void setNextNodePtr(Node nodePtr){
        nextNodePtr = nodePtr;
    }

    public Node getNextNodePtr(){
        return nextNodePtr;
    }

    public void setPrevNodePtr(Node nodePtr){
        prevNodePtr = nodePtr;
    }

    public Node getPrevNodePtr(){
        return prevNodePtr;
    }

}

class Queue{

    private Node headPtr;
    private Node tailPtr;

    public  Queue(){
        headPtr = new Node();
        tailPtr = new Node();
        headPtr.setNextNodePtr(null);
        tailPtr.setPrevNodePtr(null);
    }

    public Node getHeadPtr(){
        return headPtr;
    }

    public  Node getTailPtr(){
        return tailPtr;
    }

    public  boolean isEmpty(){

        if (headPtr.getNextNodePtr() == null)
            return true;

        return false;
    }


    public  void enqueue(int data){

        Node newNodePtr = new Node();
        newNodePtr.setData(data);
        newNodePtr.setNextNodePtr(null);

        Node lastNodePtr = tailPtr.getPrevNodePtr();

        if (lastNodePtr == null){

            headPtr.setNextNodePtr(newNodePtr);
            newNodePtr.setPrevNodePtr(null);

        }
        else{

            lastNodePtr.setNextNodePtr(newNodePtr);
            newNodePtr.setPrevNodePtr(lastNodePtr);

        }

        tailPtr.setPrevNodePtr(newNodePtr);

    }


    public  int dequeue(){

        Node firstNodePtr = headPtr.getNextNodePtr();
        Node nextNodePtr = null;

        int poppedData = -100000; //empty queue

        if (firstNodePtr != null){
            nextNodePtr = firstNodePtr.getNextNodePtr();
            poppedData = firstNodePtr.getData();
        }
        else
            return poppedData;

        if (nextNodePtr != null){
            nextNodePtr.setPrevNodePtr(null);
            headPtr.setNextNodePtr(nextNodePtr);
        }
        else{
            headPtr.setNextNodePtr(null);
            tailPtr.setPrevNodePtr(null);
        }

        return poppedData;

    }


    public  int peek(){

        Node firstNodePtr = headPtr.getNextNodePtr();

        if (firstNodePtr != null)
            return firstNodePtr.getData();
        else
            return -100000; //empty queue

    }


};

class DoublyLinkedList{

    public static void main(String[] args){

    Queue queue = new Queue();

    Scanner input = new Scanner(System.in);

    int queueSize;

    System.out.print("Enter the number of elements you want to enqueue: ");
    queueSize = input.nextInt();;

    Random randGen = new Random(System.currentTimeMillis());

    int maxValue;

    System.out.print("Enter the maximum value for an element: ");
    maxValue = input.nextInt();

    System.out.print("Elements enqueued: ");
    for (int i = 0; i < queueSize; i++){

        int value = randGen.nextInt(maxValue);
        queue.enqueue(value);
        System.out.print(value + " ");
    }

    System.out.println();

    System.out.print("Elements Dequeued: ");
    while (!queue.isEmpty()){

        System.out.print(queue.dequeue() + " ");
    }

    System.out.println();

    }

}
```



```
Enter the number of elements you want to enqueue: 10
Enter the maximum value for an element: 50
Elements enqueued: 44 16 10 0 13 11 3 7 18 46
Elements Dequeued: 44 16 10 0 13 11 3 7 18 46
```