4.1 Write a method for the Queue class in the queue.java program (Listing 4.4) that displays the contents of the queue. Note that this does not mean simply displaying the contents of the underlying array. You should show the queue contents from the first item inserted to the last, without indicating to the viewer whether the sequence is broken by wrapping around the end of the array. Be careful that one item and no items display properly, no matter where front and rear are.

public void display()

    { int startFront = front;

        for (int j = front ;j <nItems; j++ )

        {

        System.out.println(queArray[j]);

         if (j == nItems-1 )

        {       j=0;

                System.out.println(queArray[j]);

        }

      if (j==startFront-1)

          return;

       }

    }

4.2 Create a Deque class based on the discussion of deques (double-ended queues) in this chapter. It should include insertLeft(), insertRight(), removeLeft(), removeRight(), isEmpty(), and isFull() methods. It will need to support wraparound at the end of the array, as queues do.

package Queue;

import java.util.ArrayList;

import java.util.List;

public class Deque {

    private List<Integer> deque = new ArrayList<Integer>();

    public void insertLeft(int item){

        System.out.println("adding at left: "+item);

        deque.add(0,item);

        System.out.println(deque);

    }

    public void insertRight(int item){

        System.out.println("adding at right: "+item);

        deque.add(item);

        System.out.println(deque);

    }

    public void removeLeft(){

        if(deque.isEmpty()){

            System.out.println("Array is empty");

            return;

        }

        int rem = deque.remove(0);

        System.out.println("removed from left: "+rem);

        System.out.println(deque);

    }

    public void removeRight(){

        if(deque.isEmpty()){

            System.out.println("Array is empty!");

            return;

        }

        int rem = deque.remove(deque.size()-1);

        System.out.println("removed from left: "+rem);

        System.out.println(deque);

    }

    public int peakLeft(){

        int item = deque.get(0);

        System.out.println("Element at first: "+item);

        return item;

    }

    public int peakright(){

        int item = deque.get(deque.size()-1);

        System.out.println("Element at right: "+item);

        return item;

    }

    public static void main(String a[]){

        Deque deque = new Deque();

        deque.insertLeft(34);

        deque.insertRight(45);

        deque.removeLeft();

        deque.removeLeft();

        deque.removeLeft();

        deque.insertLeft(21);

        deque.insertLeft(98);

        deque.insertRight(5);

        deque.insertLeft(43);

        deque.removeRight();

    }

}

4.3 Write a program that implements a stack class that is based on the Deque class in Programming Project 4.2. This stack class should have the same methods and capabilities as the StackX class in the stack.java program (Listing 4.1).

package Stack;

import java.util.ArrayList;

import java.util.List;

public class Stack

{

 private List<Integer> stack = new ArrayList<Integer>();

 public void push(int number)

 {

     stack.add(number);

     System.out.println(stack);

 }

 public void pop()

 {

     if(stack.isEmpty())

     {

         System.out.println("Array is empty");

         return;

     }

     stack.remove(stack.size()-1);

     System.out.println(stack);

 }

 public static void main(String args[])

 {

    Stack stack = new Stack();

    stack.push(1);

    stack.pop();

    stack.push(2);

    stack.push(3);

    stack.push(4);

    stack.push(5);

    stack.pop();

    stack.pop();

 }

}