public class TesterDoublyList {

public static void main(String [] args){

Object [] a1 = {10,20,30,40,50};

DoublyList h1 = new DoublyList(a1);

System.out.println(h1.countNode()); //print: 5

Node myNode = h1.nodeAt(2);

System.out.println(myNode.element);

int index = h1.indexOf(10);

System.out.println(index);

Object [] a2 = {10,20,30,40};

DoublyList h2 = new DoublyList(a2);

h2.insert(55,0);

System.out.println();

h2.insert(45,2);

System.out.println();

h2.insert(33,5);

Object [] a3 = {10,20,30,40,70};

DoublyList h3 = new DoublyList(a3);

System.out.println("Removed element: "+ h3.remove(3));

}

}

package testerdoublylist;

public class Node {

public Object element;

public Node next;

public Node prev;

public Node(Object e, Node n, Node p){

element = e;

next = n;

prev = p;

}

public Node(int new\_data) {

}

public Object getElement(){

return element;

}

public void setElement(int v){

element = v;

}

}

package testerdoublylist;

public class DoublyList {

public Node head;

public DoublyList(Object [] array){

head = new Node(null, null, null);

Node tail = head;

for(int i = 0; i<array.length; i++){

Node dc = new Node(array[i], null, null);

tail.next = dc;

dc.prev=tail;

tail=tail.next;

}

tail.next=head; // Making it circular

head.prev=tail;

}

public void showList(){

for(Node n=head; n!= null; n=n.next){

System.out.println(n.getElement());

}

}

/\* Counts the number of Nodes in the list \*/

public int countNode(){

int c=0;

for(Node n=head.next;n!=head;n=n.next){

c++;

}

return c;

}

public Node nodeAt(int idx){

int i=0;

for(Node n=head.next;n!=head;n=n.next,i++){

if(i==idx){

return n;

}

}

return null;

}

public int indexOf(Object elem){

int i=0;

for(Node n=head.next; n!=head; n=n.next, i++){

if(n.element == elem){

return i;

}

}

return -1;

}

/\* inserts Node containing the given element at the given index \*/

public void insert(Object elem, int idx){

int range = countNode();

if(idx>=0 && idx<=range){

int i=0;

Node temp = head.next;

while(i<idx){

temp = temp.next;

i++;

}

Node node = new Node(elem, temp, temp.prev);

temp.prev.next = node;

temp.prev = node;

}

else{

System.out.println("Type again");

}

}

public Object remove(int idx1){

int ne = countNode();

if(idx1>=0 && idx1<ne){

int i=0;

Node temp = head.next;

while(i<idx1){

i++;

temp = temp.next;

}

Node dc1 = temp;

Object ret = temp.element;

temp.prev.next = temp.next;

temp.next.prev = temp.prev;

dc1 = null;

return ret;

}

else{

System.out.println("Give an input");

return null;

}

}

}