﻿public class MyList {

Node head;

static int sum=0;

public MyList(){

head = new Node (0,null);

}

public MyList(int [] a){

head = new Node (0,null);

Node copyHead = head;

copyHead.setElement(a[0]);

if (a.length!=0){

for(int i=1; i<a.length; i++){

Node newNode = new Node(a[i],null);

copyHead.next = newNode;

copyHead = newNode;

}

}

}

public MyList(MyList a){

}

public void showList(){

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

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

}

}

public boolean isEmpty(){

if (head==null) return true;

return false;

}

public void clear(){

if (head!=null) head =null;

}

public boolean checkElement (int newElement){

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

if (n.element == newElement){

return true;

System.out.println("Added");

}

}

System.out.println("Not Added");

return false;

}

public void insert(int newElement){

if (checkElement(newElement)){

System.out.println("Duplicate");

}

else{

System.out.println("Real");

}

}

public void insert(int newElement, int index){

if(index<0){

System.out.println("wrong index");

}

Node node = new Node(newElement,null);

Node temp = head;

if(index==0){

node.next = head;

head = node;

}

else{

for(int i=0; i<index-1 ;i++){

temp=temp.next;

}

node.next = temp.next;

temp.next = node;

}

Node currNode = temp.head;

System.out.print("LinkedList: ");

// Traverse through the LinkedList

while (currNode != null) {

// Print the data at current node

System.out.print(currNode.data + " ");

// Go to next node

currNode = currNode.next;

}

}

public void insertAfter(Node prev\_node, int new\_data) {

if (prev\_node == null)

{

System.out.println("The given previous node cannot be null");

return;

}

Node new\_node = new Node(new\_data);

new\_node.next = prev\_node.next;

prev\_node.next = new\_node;

}

public Node copyList(){

Node copyHead = null;

Node copyTail= null;

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

Node newNode = new Node(n.element, null);

if(copyHead==null){

copyHead=newNode;

copyTail=newNode;

}

else{

copyTail.next=newNode;

copyTail=newNode;

}

}

return copyHead;

}

public Node nodeAt(Node head, int size, int index){

if (index<0 || index>=size){

return null;

}

Node n=head;

for (int i=0; i<index; i++, n=n.next);

return n;

}

public Node reverseList(Node head){

Node copyHead = null;

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

Node newNode = new Node (n.element, null);

if(copyHead==null){

copyHead = newNode;

}

else{

newNode.next = copyHead;

copyHead = newNode;

}

}

return copyHead;

}

void reverseList(int i) {

}

public Node rotateLeft (Node head){

Node oldHead = head;

head = head.next;

Node tail= head;

while(tail.next!=null)

tail=tail.next;

tail.next=oldHead;

oldHead.next=null;

return head;

}

}

public class Node {

int element;

Node next;

Node head;

String data;

public Node (int e, Node n){

this.element = e;

this.next = n;

}

public Node(int new\_data) {

}

public int getElement(){

return element;

}

public void setElement(int v){

element = v;

}

}