Remove At Index In Linked List

Question

1. You are given a partially written LinkedList class.  
2. Here is a list of existing functions:  
2.1 addLast - adds a new element with given value to the end of Linked List  
2.2. display - Prints the elements of linked list from front to end in a single line. All elements are separated by space  
2.3. size - Returns the number of elements in the linked list.  
2.4. removeFirst - Removes the first element from Linked List.   
2.5. getFirst - Returns the data of first element.   
2.6. getLast - Returns the data of last element.   
2.7. getAt - Returns the data of element available at the index passed.   
2.8. addFirst - adds a new element with given value in front of linked list.  
2.9. addAt - adds a new element at a given index.  
2.10. removeLast - removes the last element of linked list.  
3. You are required to complete the body of removeAt function. The function should remove the element available at the index passed as parameter. If the size is 0, should print "List is empty". If the index is inappropriate print "Invalid arguments". Also consider the case when list has a single element.  
4. Input and Output is managed for you.

Input Format

Input is managed for you

Output Format

Output is managed for you

Constraints

None

Sample Input

addFirst 10  
getFirst  
addAt 0 20  
getFirst  
getLast  
display  
size  
addAt 2 40  
getLast  
addAt 1 50  
addFirst 30  
removeAt 2  
getFirst  
removeAt 0  
removeAt 1  
addAt 2 60  
display  
size  
removeAt 0  
removeAt 1  
getFirst  
quit

Sample Output

10  
20  
10  
20 10   
2  
40  
30  
20 40 60   
3  
40

import java.io.\*;

import java.util.\*;

public class Main {

public static class Node {

int data;

Node next;

}

public static class LinkedList {

Node head;

Node tail;

int size;

void addLast(int val) {

Node temp = new Node();

temp.data = val;

temp.next = null;

if (size == 0) {

head = tail = temp;

} else {

tail.next = temp;

tail = temp;

}

size++;

}

public int size() {

return size;

}

public void display() {

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

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

}

System.out.println();

}

public void removeFirst() {

if (size == 0) {

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

} else if (size == 1) {

head = tail = null;

size = 0;

} else {

head = head.next;

size--;

}

}

public int getFirst() {

if (size == 0) {

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

return -1;

} else {

return head.data;

}

}

public int getLast() {

if (size == 0) {

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

return -1;

} else {

return tail.data;

}

}

public int getAt(int idx) {

if (size == 0) {

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

return -1;

} else if (idx < 0 || idx >= size) {

System.out.println("Invalid arguments");

return -1;

} else {

Node temp = head;

for (int i = 0; i < idx; i++) {

temp = temp.next;

}

return temp.data;

}

}

public void addFirst(int val) {

Node temp = new Node();

temp.data = val;

temp.next = head;

head = temp;

if (size == 0) {

tail = temp;

}

size++;

}

public void addAt(int idx, int val) {

if (idx < 0 || idx > size) {

System.out.println("Invalid arguments");

} else if (idx == 0) {

addFirst(val);

} else if (idx == size) {

addLast(val);

} else {

Node node = new Node();

node.data = val;

Node temp = head;

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

temp = temp.next;

}

node.next = temp.next;

temp.next = node;

size++;

}

}

public void removeLast() {

if (size == 0) {

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

} else if (size == 1) {

head = tail = null;

size = 0;

} else {

Node temp = head;

for (int i = 0; i < size - 2; i++) {

temp = temp.next;

}

tail = temp;

tail.next = null;

size--;

}

}

public void removeAt(int idx) {

if(idx==0){

removeFirst();

}else if(idx<0 || idx>=size){

System.out.println("Invalid arguments");

}else if(idx == size-1){

removeLast();

}else{

Node temp = head;

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

temp=temp.next;

}

Node rem=temp.next;

temp.next=rem.next;

rem.next=null;

size--;

}

}

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

LinkedList list = new LinkedList();

String str = br.readLine();

while (str.equals("quit") == false) {

if (str.startsWith("addLast")) {

int val = Integer.parseInt(str.split(" ")[1]);

list.addLast(val);

} else if (str.startsWith("size")) {

System.out.println(list.size());

} else if (str.startsWith("display")) {

list.display();

} else if (str.startsWith("removeFirst")) {

list.removeFirst();

} else if (str.startsWith("getFirst")) {

int val = list.getFirst();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("getLast")) {

int val = list.getLast();

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("getAt")) {

int idx = Integer.parseInt(str.split(" ")[1]);

int val = list.getAt(idx);

if (val != -1) {

System.out.println(val);

}

} else if (str.startsWith("addFirst")) {

int val = Integer.parseInt(str.split(" ")[1]);

list.addFirst(val);

} else if (str.startsWith("addAt")) {

int idx = Integer.parseInt(str.split(" ")[1]);

int val = Integer.parseInt(str.split(" ")[2]);

list.addAt(idx, val);

} else if (str.startsWith("removeLast")) {

list.removeLast();

} else if (str.startsWith("removeAt")) {

int idx = Integer.parseInt(str.split(" ")[1]);

list.removeAt(idx);

}

str = br.readLine();

}

} }