3. Java program to delete a node from the middle of the singly linked list

In this program, we will create a singly linked list and delete a node from the middle of the list. To accomplish this task, we will calculate the size of the list and then divide it by 2 to get the mid-point of the list. Node temp will point to head node. We will iterate through the list till midpoint is reached. Now, the temp will point to middle node and node current will point to node previous to temp. We delete the middle node such that current's next node will point to temp's next node.

public class Main{

class Node{

int data;

Node next;

public Node(int data)

{

this.data = data;

this.next = null;

}

}

public Node head = null;

public Node tail = null;

public int size;

public void addNode(int data) {

Node newNode = new Node(data);

if(head == null) {

head = newNode;

tail = newNode;

}

else {

tail.next = newNode;

tail = newNode;

}

size++;

}

void deleteFromMid() {

Node temp, current;

if(head == null) {

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

return;

}

else {

int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);

if( head != tail ) {

temp = head;

current = null;

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

current = temp;

temp = temp.next;

}

if(current != null) {

current.next = temp.next;

temp = null;

}

else {

head = tail = temp.next;

temp = null;

}

}

else {

head = tail = null;

}

}

size--;

}

public void display() {

Node current = head;

if(head == null) {

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

return;

}

while(current != null) {

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

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

Main sList = new Main();

sList.addNode(1);

sList.addNode(2);

sList.addNode(3);

sList.addNode(4);

System.out.println("Original List: ");

sList.display();

while(sList.head != null) {

sList.deleteFromMid();

System.out.println("Updated List: ");

sList.display();

}

}

}

Ouput:

