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

```
public class deleteMid{

//Represent a node of the singly linked list
class Node{
int data;
Node next;
public Node(int data)
{
this.data = data;
this.next = null;
}
}

//Represent the head and tail of the singly linked list
public Node head = null;
public Node tail = null;

public int size;
//addNode() will add a new node to the list
 public void addNode(int data) {
//Create a new node
 Node newNode = new Node(data);
//Checks if the list is empty
if(head == null) {
 //If list is empty, both head and tail will point to new node
head = newNode;
tail = newNode;
}
else {
//newNode will be added after tail such that tail's next will point to newNode
tail.next = newNode;
//newNode will become new tail of the list
tail = newNode;
}
size++;
}
//deleteFromMid() will delete a node from the middle of the list
void deleteFromMid() {
Node temp, current;
//Checks if the list is empty
if(head == null) {
System.out.println("List is empty");
return;
}
else {
//Store the mid position of the list
 int count = (size % 2 == 0) ? (size/2) : ((size+1)/2);
//Checks whether the head is equal to the tail or not, if yes then the list has only one node.
if( head != tail ) {
 //Initially, temp will point to head
temp = head;
current = null;
//Current will point to node previous to temp
```

```
 //If temp is pointing to node 2 then current will point to node 1.
for(int i = 0; i< count-1; i++){
current = temp;
temp = temp.next;
}
if(current != null) {
 //temp is the middle that needs to be removed.
 //So, current node will point to node next to temp by skipping temp.
current.next = temp.next;
//Delete temp
 temp = null;
 }
 //If current points to NULL then, head and tail will point to node next to temp.
 else {
head = tail = temp.next;
//Delete temp
temp = null;
}
}
//If the list contains only one element
 //then it will remove it and both head and tail will point to NULL
else {
head = tail = null;
}
}
size--;
}
//display() will display all the nodes present in the list
public void display() {
//Node current will point to head
Node current = head;
if(head == null) {
System.out.println("List is empty");
 return;
      }
      while(current != null) {
         //Prints each node by incrementing pointer
System.out.print(current.data + " ");
         current = current.next;
      }
System.out.println();
   }

public static void main(String[] args) {

deleteMidsList = new deleteMid();

      //Adds data to the list
sList.addNode(1);
sList.addNode(2);
sList.addNode(3);
sList.addNode(4);

      //Printing original list
System.out.println("Original List: ");
sList.display();
```

```
        while(sList.head != null) {
sList.deleteFromMid();
            //Printing updated list
System.out.println("Updated List: ");
sList.display();
        }
    }
}
```

Output:

```
Original List:
1 2 3 4
Updated List:
1 3 4
Updated List:
1 4
Updated List:
4
Updated List:
List is empty
```