Class Demo

Problem statement: Find the nth element from end of Linked List.

Method of solving:

1. In this method we need to traverse the list TWO times.

Algorithm:

* 1. Traverse through whole linked list and find the length of list.
  2. While traversing second time, we go till (length-n)th node and retrieve data from that node.

1. In this method we use TWO pointers and traverse through the list only ONCE.

Algorithm:

* 1. Initialize two pointers ptr1 and ptr2 and point them to head.
  2. Move ptr1 to n-1 nodes.
  3. Now traverse through the list until ptr1 points to null, incrementing both ptr1 and ptr2.
  4. The node next to the node that ptr2 will be pointing to will be the nth node from last.

Method a implementation:

public class LinkedListFull

{

private Node head;

private static class Node

{

private int value;

private Node next;

Node(int value)

{

this.value=value;

}

}

public void addLast(Node node)

{

if(head==null){

head=node;}

else

{

Node tmp=head;

while(tmp.next!=null)

tmp=tmp.next;

tmp.next=node;

}

}

public int listLength()

{

int len=0;

Node tmp=head;

while(tmp.next!=null)

{

len++;

tmp=tmp.next;

}

return len;

}

public Node nthFromLast(Node node,int len,int n)

{

Node tmp=head;

for(int i=0;i<=len-n;i++)

{

tmp=tmp.next;

}

return tmp;

}

public void printList()

{

Node tmp=head;

while(tmp!=null)

{

System.out.print(tmp.value+" ");

tmp=tmp.next;

}

//System.out.print(tmp.value+" ");

System.out.println();

}

public static void main(String[] args)

{

LinkedListFull list=new LinkedListFull();

Node head=new Node(5);

list.addLast(head);

list.addLast(new Node(4));

list.addLast(new Node(10));

list.addLast(new Node(1));

list.addLast(new Node(2));

list.addLast(new Node(3));

list.addLast(new Node(6));

list.printList();

int length=list.listLength();

Node res=list.nthFromLast(head,length,3);

System.out.println(res.value);

}

}

method b implementation:

public class LinkedList

{

private Node head;

private static class Node

{

private int value;

private Node next;

Node(int value)

{

this.value=value;

}

}

public void addLast(Node node)

{

if(head==null)

head=node;

else

{

Node tmp=head;

while(tmp.next!=null)

{

tmp=tmp.next;

}

tmp.next=node;

}

}

public void printList()

{

Node tmp=head;

while(tmp!=null)

{

System.out.print(tmp.value+" ");

tmp=tmp.next;

}

System.out.println();

}

public Node nthFromLast(Node node,int n)

{

Node ptr1=head;

Node ptr2=head;

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

{

ptr1=ptr1.next;

}

while(ptr1.next!=null)

{

ptr1=ptr1.next;

ptr2=ptr2.next;

}

return ptr2.next;

}

public static void main(String[] args)

{

LinkedList list=new LinkedList();

Node head=new Node(5);

list.addLast(head);

list.addLast(new Node(4));

list.addLast(new Node(10));

list.addLast(new Node(1));

list.addLast(new Node(2));

list.addLast(new Node(3));

list.addLast(new Node(6));

list.printList();

Node res=list.nthFromLast(head,3);

System.out.println(res.value);

}

}