package com.lesson4.linkedlist;

public class LinkedList {

Node head;

static class Node{

int data;//Store Data

Node next;//Store Link

Node(int data){

this.data=data;

this.next=null;

}

}

//Method to insert nodes in linked list

public static LinkedList insert(LinkedList list,int data)

{

// Create a new node with given data

Node new\_node = new Node(data);

new\_node.next = null;

// If the Linked List is empty, then make the new node as head

if (list.head == null) {

list.head = new\_node;

} else {

// Else traverse till the last node and insert the new\_node there

Node last = list.head;

while (last.next != null) {

last = last.next;

}

// Insert the new\_node at last node

last.next = new\_node;

}

return list;

}

public static void printList(LinkedList list) {

Node currNode = list.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;

}

System.out.println();

}

// Method to delete a node in the LinkedList by KEY

public static LinkedList deleteByKey(LinkedList list, int key) {

// Store head node

Node currNode = list.head, prev = null;

if (currNode != null && currNode.data == key) {

list.head = currNode.next; // Changed head

System.out.println(key + " found and deleted");

return list;

}

while (currNode != null && currNode.data != key) {

prev = currNode;

currNode = currNode.next;

}

if (currNode != null) {

prev.next = currNode.next;

System.out.println(key + " found and deleted");

}

if (currNode == null) {

System.out.println(key + " not found");

}

return list;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

LinkedList list = new LinkedList();

list.insert(list, 10);

list.insert(list, 20); list.insert(list, 30); list.insert(list, 40);

list.insert(list, 50);

printList(list);

deleteByKey(list, 10);

printList(list);

}

}