

Mohammad Ali Jinnah University

Chartered by Government of Sindh - Recognized by HEC

**Lab Linked List 2**

**Name:** Muhamad Fahad

**Id:** FA19-BSSE-0014

**Subject:** Data Structures and Algorithms Lab (CS 2511)

**Section:** AM

**Teacher:** MUHAMMAD MUBASHIR KHAN

**Date:** Tuesday, December 22, 2020

**Task:**

**Implement Singly Linked List with following operations:**

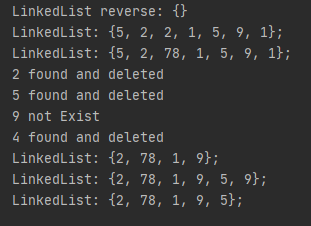
**1) Insertion**

**2) Insertion at index**

**3) Deletion by value**

**4) Printing list**

**Output:**

****

**Code:**

public class PracticeQuestion {  
 public static void main(String[] args) {  
 Linkedlist list = new Linkedlist();  
 System.*out*.println(list.*Displayreverse*()); //Q2  
  
 list.*insert*(1); // Q3  
 list.*insertAtstart*(2); //Q4  
 list.*insert*(5,2); //Q5 at any postion or mid  
 list.*insert*(9,2);  
 list.*insert*(1); // Q3  
 list.*insertAtstart*( 2); //Q4  
 list.*insert*( 5,0); //Q5 at any postion or mid  
  
 System.*out*.println(list.*Display*());  
  
 list.*update*( 2,78); //Q5 at any postion or mid  
  
 System.*out*.println(list.*Display*());  
  
 list.*deleteFront*();  
 list.*deleteByValue*(5);  
 list.*deleteBykey*(9);  
 list.*delete*();  
 System.*out*.println(list.*Display*());  
  
 list.*insert*( 5);  
 list.*insert*( 9);  
 System.*out*.println(list.*Display*());  
  
 list.*deleteDuplicate*();  
 System.*out*.println(list.*Display*());  
  
 }  
}

**Main class(class in which object of the linked list used).**

**Linkedlist class**

package com.company.Linkedlist;  
  
import java.util.HashSet;  
  
public class Linkedlist {  
 private static Node *head*;  
  
 static class Node{  
 private int Value;  
 private Node pointer;  
  
 Node(int data){  
 Value = data;  
 pointer = null;  
 }  
 }  
  
 static int getLenght() {  
 int i = 0;  
 Node last = *head*;  
 while (last.pointer != null) {  
 i++;  
 last = last.pointer;  
 }  
 return i;  
 }  
 static boolean isEmpty() {  
 boolean condition = true;  
  
 if (*head* == null)  
 condition = false;  
  
 return condition;  
 }  
  
 // add the element in the linked list  
 static void insert(int data) {  
 Node new\_node = new Node(data);  
 new\_node.pointer = null;  
  
 if (!*isEmpty*())  
 *head* = new\_node;  
 else {  
 Node last = *head*;  
 while (last.pointer != null)  
 last = last.pointer;  
  
 last.pointer = new\_node;  
 }  
 }  
 static void insert( int data, int key) {  
 int size = *getLenght*();  
 Node new\_node = new Node(data);  
 Node prev = null;  
 Node current = *head*;  
  
 if(key == 0)  
 *insertAtstart*(data);  
 else if(key > size-1)  
 *insert*(data);  
 else{  
 for(int i = 0; i < key; i++)  
 current = (prev = current).pointer;  
  
 new\_node.pointer = current;  
 prev.pointer = new\_node;  
 }  
 }  
 static void insertAtstart(int data) {  
 Node new\_node = new Node(data);  
  
 if (!(*isEmpty*()))  
 *head* = new\_node;  
 else {  
 new\_node.pointer = *head*;  
 *head* = new\_node;  
 }  
 }  
  
 // delete the element in the linked list  
 static void deleteByValue(int key){  
 Node currNode = *head*,  
 prev = null;  
  
 if (currNode != null && currNode.Value == key) {  
 *head* = currNode.pointer;  
 System.*out*.println(key + " found and deleted");  
 return;  
 }  
  
 while (currNode != null && currNode.Value != key)  
 currNode = (prev = currNode).pointer;  
  
  
 if (currNode != null) {  
 prev.pointer = currNode.pointer;  
 System.*out*.println(key + " found and deleted");  
 }  
  
 if (currNode == null)  
 System.*out*.println(key + " not found");  
  
 }  
 static void deleteBykey(int key){  
 int size = *getLenght*();  
 Node currNode = *head*,  
 prev = null;  
  
 if (size < key) {  
 System.*out*.println(key + " not Exist");  
 return;  
 }  
  
 if (key == 0){  
 *head* = currNode.pointer;  
 System.*out*.println((currNode.pointer).Value + " found and deleted");  
 return;  
 }  
  
 for (int i=0; i<key; i++)  
 currNode = (prev = currNode).pointer;  
  
 prev.pointer = currNode.pointer;  
 System.*out*.println(key + " found and deleted");  
  
 }  
 static void delete(){  
 *deleteBykey*(*getLenght*());  
 }  
 static void deleteFront(){  
 *deleteBykey*(0);  
 }  
 static void deleteDuplicate(){  
 HashSet<Integer> hs = new HashSet<>();  
  
 Node current = *head*;  
 Node prev = null;  
 while (current != null) {  
 if (hs.contains(current.Value)) prev.pointer = current.pointer;  
 else {  
 hs.add(current.Value);  
 prev = current;  
 }  
 current = current.pointer;  
 }  
 }  
  
 // update the element in the linked list  
 static void update( int index, int value){  
 Node currNode = *head*;  
 if (*getLenght*() < index) {  
 System.*out*.println("Index not Exist! ");  
 return;  
 }  
  
 for (int i = 0; i < index; i++)  
 currNode = currNode.pointer;  
  
 currNode.Value = value;  
 }  
  
 // search the element in the linked list  
 static Boolean Search( int key) {  
 Node currNode = *head*;  
 Boolean condition = false;  
  
 if (currNode == null)  
 return condition;  
  
  
 while (currNode.Value != key)  
 currNode = currNode.pointer;  
  
 if (currNode != null) condition = true;  
 else System.*out*.println(key + " not Exist(404 Error)");  
  
 return condition;  
 }  
  
 //Sorting the Element in the  
 static void sortList() {  
 Node current = *head*, index = null;  
 int temp;  
  
 if(current == null) {  
 return;  
 }  
 else {  
 while(current != null) {  
 index = current.pointer;  
  
 while(index != null) {  
 if(current.Value > index.Value) {  
 temp = current.Value;  
 current.Value = index.Value;  
 index.Value = temp;  
 }  
 index = index.pointer;  
 }  
 current = current.pointer;  
 }  
 }  
 }  
  
 //Merge Two linked list in the element  
 static Linkedlist Merge(Linkedlist list1,Linkedlist list2){  
 Linkedlist list = new Linkedlist();  
 int l1 = *getLenght*(),l2 = *getLenght*();  
 Node current = list1.*head*;  
  
 for (int i = 0; i <= (l1+l2)+1; i++) {  
 list.*insert*(current.Value);  
 if (l1 != i) current = current.pointer;  
 else current = list2.*head*;  
 }  
 return list;  
 }  
  
 //count the odd and even nodes  
 static int countOdd(){  
 int count = 0;  
 Node current = *head*;  
 while (current.pointer != null){  
 if (current.Value % 2 == 0)  
 count++;  
 current = current.pointer;  
 }  
 return count;  
 }  
 static int countEven(){  
 int count = 0;  
 Node current = *head*;  
 while (current.pointer != null){  
 if (current.Value % 2 != 0)  
 count++;  
 current = current.pointer;  
 }  
 return count;  
 }  
  
 //swap the number  
 static void swap(Node n1,Node n2){  
 int temp = n1.Value;  
 n1.Value = n2.Value;  
 n2.Value = temp;  
 }  
 static void swapAdj(){  
 int count = 0;  
 Node current = *head*;  
 while (current.pointer != null) {  
 *swap*(current,(current = current.pointer));  
 current = current.pointer;  
 }  
 }  
  
  
 //Display methods  
 static String Display(){  
 Node currNode = *head*;  
 String display = "LinkedList: {";  
  
 while (currNode != null) {  
 display += currNode.Value + ", ";  
 currNode = currNode.pointer;  
 }  
 display += "\b\b};";  
 return display;  
 }  
 static String Displayreverse (){  
 Node currNode = *head*;  
 String display = "}";  
  
 while (currNode != null) {  
 display += currNode.Value + " ";  
 currNode = currNode.pointer;  
 }  
 display += "{";  
  
 display = "LinkedList reverse: " +(new StringBuilder(display)).reverse();  
 return display;  
 }  
  
}