|  |  |
| --- | --- |
| Student Name | Uzair Hussain |
| Roll Number | 21SW085 |
| Section # | 3rd or III |
| Lab # | 7th – Singly LinkedList |

**Task#01**

**Code:**

import java.util.Scanner;

//<<<DSA Task>>>

class Music {

    String song;

    Music nextSong;

    Scanner s = new Scanner(System.in);

    public Music (String song){

        this.song=song;

    }

    int menu() {

        System.out.println("\n\tWelcome to Fire Air Music Player");

        System.out.println("\n\t\tOperations: ");

        System.out.println("1. Display all songs");

        System.out.println("2. Search a song");

        System.out.println("3. Add a new song");

        System.out.println("4. Update a song");

        System.out.println("5. Delete a song");

        System.out.println("0: Exit");

        System.out.print("\nEnter Your Choice: ");

        int Ch = s.nextInt();

        return Ch;

    }

    void display() {

        for(Music temp=this;temp!=null;temp=temp.nextSong){

            System.out.println("=> "+temp.song);

        }

    }

    int total() {

        int total\_songs=0;

        for(Music temp=this;temp!=null;temp=temp.nextSong){

            total\_songs++;

        }

        return total\_songs;

    }

    public Music add(String song,int pos){

        Music temp=this;

        Music temp1=this;

        Music data\_node=new Music(song);

        if(pos==1){

            data\_node.nextSong=temp;

            return data\_node;

        }

        if(pos-1<=total() && pos>0){

           int counter=2;

            for(temp=temp1;temp!=null;temp=temp.nextSong){

              if(counter<pos){

                counter++;

              }

              else if(counter==pos){

                 break;

              }

            }

            data\_node.nextSong=temp.nextSong;

            temp.nextSong=data\_node;

            temp=temp1;

            return temp;

        }

        System.out.println("position is invalid ");

        return temp;

      }

    int search(String songname) {

        Music tempMusic=this;

        Music tempMusic1=this;

        int counter=0;

        for(tempMusic=tempMusic1;tempMusic!=null;tempMusic=tempMusic.nextSong){

            counter++;

            if(tempMusic.song.equalsIgnoreCase(songname))

            {

               System.out.println("The song is present ");

                      return counter;

            }

        }

        System.out.println("The song is not present");

        return -1;

    }

    Music delete(String songname) {

        Music tempMusic=this;

        if(tempMusic.song.equalsIgnoreCase(songname)){

            System.out.println("The song is deleted successfully");

            return tempMusic.nextSong;

        }

        for(Music temp=tempMusic.nextSong;temp!=null;temp=temp.nextSong){

             if(temp.song.equalsIgnoreCase(songname))

            {

                if(temp.nextSong==null){

                    tempMusic.nextSong=null;

                    System.out.println("The song is deleted successfully");

                    return this;

                }

                tempMusic.nextSong=temp.nextSong;

                System.out.println("The song is deleted successfully");

            }

          tempMusic=tempMusic.nextSong;

        }

        return this;

    }

    void update(String songname,String newname) {

        Music tempMusic=this;

        Music tempMusic1=this;

        for(tempMusic=tempMusic1;tempMusic!=null;tempMusic=tempMusic.nextSong){

            if(tempMusic.song.equalsIgnoreCase(songname))

            {

                    tempMusic.song=newname;

                    System.out.println("The song is updated successfully");

                    return;

            }

        }

            System.out.println("The song is not updated ");

    }

}

class Task1\_FireAir {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String[] song = { "Dil Dil Pakistan", "Memories", "Pasoori","hello" };

        Music musicList=new Music(song[0]);

        Music temp=musicList;

        for(int i=1;i<song.length;i++){

            temp.nextSong=new Music(song[i]);

            temp=temp.nextSong;

        }

        int c = 0;

        do {

            c = musicList.menu();

            switch(c){

                case 1:{

                    musicList.display();

                    break;

                }

                case 2:{

                    System.out.print("Enter song name to search : ");

                    sc.nextLine();

                    String n=sc.nextLine();

                    musicList.search(n);

                    break;

                }

                case 3:{

                    System.out.print("Enter the song name : ");

                    sc.nextLine();

                    String m=sc.nextLine();

                    System.out.print("Enter position : ");

                    int pos=sc.nextInt();

                    musicList=musicList.add(m,pos);

                    break;

                }

                case 4:{

                    System.out.print("Enter name of song to update : ");

                    sc.nextLine();

                    String mi=sc.nextLine();

                    System.out.print("Enter new name of song : ");

                    String  e=sc.nextLine();

                    musicList.update(mi,e);

                    break;

                }

                case 5:{

                    System.out.print("Enter name of song to delete : ");

                    String so=sc.nextLine();

                    musicList.delete(so);

                    break;

                }

                case 0:{

                    continue;

                }

                default:{

                    System.out.println("The number is incorrect ");

                }

            }

            System.out.print("\nPress:\n1 to continue \n0 to exit \n\nChoice:");

            c = sc.nextInt();

        } while (c >= 1 && c <= 4);

    }

}

**Output 1:**

**Text

Description automatically generated**

**Task 2:**

**Code 2:**

class Task2\_TourRoute{

    static class Node{     //Node Class

        String Location;

        Node next;

        Node(String L){     //Node class Constructor

            Location=L;

        }

    }

    public static void main(String[] args) {

        //Routes Array

        String[] R1={"Thandi Sarak", "Wadhu Wah", "Abdullah Mall Rd", "Hyderabad Bypass", "Jamshoro", "Mehran University Campus Rd"};

        String[] R2={"Dr. Ziauddin Rd", "CMH Rd", "NH5/AH2", "Hyderabad Bypass", "Jamshoro", "Mehran University Campus Rd"};

        //Estimated Distance and Time Taken(in mins) for both routes

        float

        Estimated\_Distance2=16.3f,

        Estimated\_Distance1=14.8f,

        Estimated\_Mins1=28,

        Estimated\_Mins2=32;

        //Node Objects

        Node Route1=new Node(R1[0]);

        Node Route2=new Node(R2[0]);

        //Insertion in Route1

        Node t=Route1;

        for(int i=1;i<R1.length;i++){

            t.next=new Node(R1[i]);

            t=t.next;

        }

        //Insertion in Route2

        t=Route2;

        for(int i=1;i<R2.length;i++){

            t.next=new Node(R2[i]);

            t=t.next;

        }

        //Displaying Route1

        System.out.println("\nRoute No.1 \t Estimated Distance: " + Estimated\_Distance1 + " Km \t Estimated Time: " + Estimated\_Mins1 + " mins\n");

        for(Node n=Route1;n!=null;n=n.next){

            if(n.next==null){

                System.out.println(n.Location);

                System.out.println("Arrived at Destination\n");

                break;

            }

            System.out.print(n.Location + " -> ");

        }

        //Displaying Route2

        System.out.println("\nRoute No.2 \t Estimated Distance: " + Estimated\_Distance2 + " Km \t Estimated Time: " + Estimated\_Mins2 + " mins\n");

        for(Node n=Route2;n!=null;n=n.next){

            if(n.next==null){

                System.out.println(n.Location);

                System.out.println("Arrived at Destination\n");

                break;

            }

            System.out.print(n.Location + " -> ");

        }

        //Displaying fastest route

        if(Estimated\_Mins1>Estimated\_Mins2){

            System.out.println("\nThe Fastest Route is Route No. 1");

        }

        else if(Estimated\_Mins2>Estimated\_Mins1){

            System.out.println("\nThe Fastest Route is Route No. 2\n");

        }

    }

}

**Output:**

**Text

Description automatically generated**

**Task 3:**

**Code 3:**

class Task3\_Generic\_linkedlist<T extends Comparable<T>> {

    Node<T> head;

    int size;

    static class Node<T> {      // generic node class passing general data type of T

        T data;

        Node<T> next;

        public Node(T data) {

            this.data = data;

            this.next = null;

        }

    }

    public Task3\_Generic\_linkedlist() {         // constructor to initialize head and size value

        head = null;

        size = 0;

    }

    public boolean isEmpty() {                  // method to check status of linkedlist

        return head == null;

    }

    public void getSize() {                      // it checks the size of linkedlist

        System.out.println("Size of linkedlist is: "+size);

    }

    public void add(T data) {                   // adding elements in linkedlist, it adds at last generally

        Node<T> newNode = new Node<T>(data);

        newNode.next = head;

        head = newNode;

        size++;

    }

    public void insertAtFirst(T data) {         // specific method to add element at start of linkedlist by adding new node

        Node<T> newNode = new Node<T>(data);

        newNode.next = head;

        head = newNode;

        size++;

    }

    public void insertAtLast(T data) {          // specific method to add element at last of linkedlist by adding new node

        Node<T> newNode = new Node<T>(data);

        if (isEmpty()) {

            head = newNode;

        } else {

            Node<T> temp = head;

            while (temp.next != null) {

                temp = temp.next;

            }

            temp.next = newNode;

        }

        size++;

    }

    public void insertAtPosition(T data, int position) {        // it insert at particular position by giving index

        Node<T> newNode = new Node<T>(data);

        if (position == 0) {

            newNode.next = head;

            head = newNode;

        } else {

            Node<T> temp = head;

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

                temp = temp.next;

            }

            newNode.next = temp.next;

            temp.next = newNode;

        }

        size++;

    }

    public void deleteFirst() {             // it deletes the very first elememnt in linkedlist

        if (!isEmpty()) {

            head = head.next;

            size--;

        }

    }

    public void deleteLast() {          // it deletes the last element

        if (!isEmpty()) {

            if (head.next == null) {

                head = null;

            } else {

                Node<T> temp = head;

                while (temp.next.next != null) {

                    temp = temp.next;

                }

                temp.next = null;

            }

            size--;

        }

    }

    public void deleteAtPosition(int position) {            // it deletes the element at particular postion

        if (isEmpty()) {

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

            return;

        }

        if (position < 0 || position >= size) {

            System.out.println("Invalid position");

            return;

        }

        if (position == 0) {

            head = head.next;

        } else {

            Node<T> current = head;

            int count = 0;

            while (count < position - 1) {

                current = current.next;

                count++;

            }

            current.next = current.next.next;

        }

        size--;

    }

    public void deleteValue(T value) {          //it deletes by value/element by searching

        Node<T> current = head;

        Node<T> previous = null;

        while (current != null) {

            if (current.data.equals(value)) {

                if (previous != null) {

                    previous.next = current.next;

                } else {

                    head = current.next;

                }

                break;

            }

            previous = current;

            current = current.next;

        }

    }

    public boolean search(T value) {            // it searches the element in linkeedlist

        Node<T> temp = head;

        while (temp != null) {

            if (temp.data.equals(value)) {

                return true;

            }

            temp = temp.next;

        }

        return false;

    }

    public T findLargest() {

        if (head == null) {

            throw new IllegalStateException("List is empty");

        }

        T max = head.data;

        Node<T> current = head.next;

        while (current != null) {

            if (current.data.compareTo(max) > 0) {

                max = current.data;

            }

            current = current.next;

        }

        return max;

    }

    public T findLowest() {

        if (head == null) {

            throw new IllegalStateException("List is empty");

        }

        T min = head.data;

        Node<T> current = head.next;

        while (current != null) {

            if (current.data.compareTo(min) < 0) {

                min = current.data;

            }

            current = current.next;

        }

        return min;

    }

    public void display() {             //method to display linkedlist

        Node<T> current = head;

        System.out.print(" [ ");

       for(current=head;current!=null;current=current.next){

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

       }

       System.out.println(" ] ");

    }

    public static void main(String[] args) {

        Task3\_Generic\_linkedlist<Integer> t=new Task3\_Generic\_linkedlist<Integer>();

        t.getSize();

        t.add(2);

        t.add(7);

        t.add(8);

        // checking list state (empty or having some elements)

        System.out.println(t.isEmpty());

        // displaying

        t.display();

        // inserting at first and last

        t.insertAtFirst(1);

        t.insertAtLast(10);

        // inserting at specific postition

        t.insertAtPosition(4, 2);

        //displaying list again

        t.display();

        // checking size of list

        t.getSize();

        //finding largest element in list

        System.out.println(t.findLargest());

        //finding lowest element in list

        System.out.println(t.findLowest());

        //searching by number

        System.out.println(t.search(7));

        //deleting at specific position and deleting value directly

        t.deleteAtPosition(2);

        //deleting first and last

        t.deleteFirst();

        t.deleteLast();

        //displaying list again

        t.display();

        //deleting value directly

        t.deleteValue(2);

        //displaying list again

        t.display();

    }

}

**Output 3:**

**Text

Description automatically generated**

**Github Repository for all Lab Tasks: (from lab 1 to continue)**

[**https://github.com/UzairHussain193/DSA\_LABS\_21SW**](https://github.com/UzairHussain193/DSA_LABS_21SW)

**The End!**