DATA STRUTURES USING JAVA

**Graphs**

(creating graph, inserting vertices, BFS, DFS)

Program:

package graph;

import java.util.\*;

public class graph

{

public class Node

{

int data;

Node next;

Node(int d)

{

data=d;

next=null;

}

}

Node front,rear;

Node a[]=new Node[6];

public void enqueue(int d)

{

Node nn=new Node(d);

if(front==null)

{

front=nn;

rear=nn;

}

else

{

rear.next=nn;

rear=nn;

}

}

public int dequeue()

{

int d=front.data;

Node h=front;

front=front.next;

h.next=null;

return d;

}

public boolean isempty()

{

return (front==null)?true:false;

}

public void insert(int src, int dest)

{

Node nn=new Node(dest);

nn.next=a[src];

a[src]=nn;

nn=new Node(src);

nn.next=a[dest];

a[dest]=nn;

}

public void bfs(int k)

{

boolean[] b=new boolean[6];

b[k]=true;

enqueue(k);

while(!isempty())

{

int d=dequeue();

System.***out***.print(d+" ");

Node temp=a[d];

while(temp!=null)

{

int data=temp.data;

if(!b[data])

{

b[data]=true;

enqueue(data);

}

temp=temp.next;

}

}

}

boolean b[]=new boolean[6];

public void dfs(int s)

{

b[s]=true;

System.***out***.print(s+" ");

Node temp=a[s];

while(temp!=null)

{

int m=temp.data;

if(!b[m])

{

dfs(m);

}

temp=temp.next;

}

}

public static void main(String args[])

{

graph g=new graph();

g.insert(0,1);

g.insert(0, 2);

g.insert(1,3);

g.insert(2,3);

g.insert(2,4);

g.insert(3,4);

g.insert(3,5);

g.insert(4,5);

g.bfs(0);

System.***out***.println();

g.dfs(0);

}

}

**Trees**

(creating tree, inserting node, searching element in tree, inorder, preorder, postorder)

Program:

package treesandoperations;

import java.util.\*;

public class tree

{

public class Node

{

int data;

Node left;

Node right;

public Node(int d)

{

data=d;

left=null;

right=null;

}

}

Node root=null;

public void insert(int d)

{

root=insertrec(root,d);

}

public Node insertrec(Node temp, int d)

{

if(temp==null)

{

Node nn=new Node(d);

return nn;

}

else if(d>temp.data)

{

temp.right=insertrec(temp.right,d);

}

else if(d<temp.data)

{

temp.left=insertrec(temp.left,d);

}

return temp;

}

public void inorder(Node temp)

{

if(temp!=null)

{

inorder(temp.left);

System.***out***.print(temp.data+" ");

inorder(temp.right);

}

}

public void preorder(Node temp)

{

if(temp!=null)

{

System.***out***.print(temp.data+" ");

preorder(temp.left);

preorder(temp.right);

}

}

public void postorder(Node temp)

{

if(temp!=null)

{

postorder(temp.left);

postorder(temp.right);

System.***out***.print(temp.data+" ");

}

}

public void search(int key)

{

if(searchRec(root,key))

{

System.***out***.println("element found");

}

else

{

System.***out***.println("element not found");

}

}

public boolean searchRec(Node root, int key) {

if (root == null)

{

return false;

}

else if (root.data == key)

{

return true;

}

else if (root.data > key)

{

return searchRec(root.left, key);

}

else

{

return searchRec(root.right, key);

}

}

public static void main(String args[])

{

tree t=new tree();

Scanner sc=new Scanner(System.***in***);

int n=sc.nextInt();

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

{

int k=sc.nextInt();

t.insert(k);

}

System.***out***.println("enter element to search:");

int key=sc.nextInt();

t.search(key);

System.***out***.println("inorder:");

t.inorder(t.root);

System.***out***.println();

System.***out***.println("preorder");

t.preorder(t.root);

System.***out***.println();

System.***out***.println("postorder");

t.postorder(t.root);

}

}

**Queue**

(enqueue, dequeue, isempty)

Programs:

package Queue\_and\_operations;

import java.util.\*;

public class queue

{

public class Node

{

int data;

Node next;

Node(int d)

{

data=d;

next=null;

}

}

Node front=null;

Node rear=null;

public void enqueue(int d)

{

Node nn=new Node(d);

if(front==null)

{

front=nn;

rear=nn;

}

else

{

rear.next=nn;

rear=nn;

}

}

public void dequeue()

{

System.***out***.println(front.data+" ");

Node h=front;

front=front.next;

h.next=null;

}

public void isempty()

{

if(front==null)

{

System.***out***.println("Queue is empty");

}

else

{

System.***out***.println("something is there");

}

}

public void display()

{

Node t=front;

if(t!=null)

{

System.***out***.println(t.data+" ");

t=t.next;

}

}

public static void main(String args[])

{

queue q=new queue();

Scanner sc=new Scanner(System.***in***);

int n=sc.nextInt();

int k;

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

{

k=sc.nextInt();

q.enqueue(k);

}

q.display();

q.dequeue();

q.display();

q.isempty();

}

}

**Stack**

(push, pull, isempty)

Program:

package Stack\_and\_opearations;

import java.util.\*;

public class CreatingStack

{

public class Node

{

int data;

Node next;

Node(int d)

{

data=d;

next=null;

}

}

Node top=null;

public void push(int d)

{

Node nn=new Node(d);

if(top==null)

{

top=nn;

}

else

{

nn.next=top;

top=nn;

}

}

public void pop()

{

if(top==null)

{

System.***out***.println("stack is empty");

}

else

{

System.***out***.println(top.data+" ");

top=top.next;

}

}

public void isempty()

{

if(top==null)

{

System.***out***.print("stack is empty");

}

else

{

System.***out***.print("Something is there");

}

}

public void display()

{

Node t=top;

while(t!=null)

{

System.***out***.print(t.data+" ");

t=t.next;

}

System.***out***.println();

}

public static void main(String args[])

{

CreatingStack cs=new CreatingStack();

Scanner sc=new Scanner(System.***in***);

System.***out***.println("enter the no of elements to insert");

int n=sc.nextInt();

int k;

System.***out***.println("enter the elements");

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

{

k=sc.nextInt();

cs.push(k);

}

System.***out***.println("Stack after pushing elements:");

cs.display();

System.***out***.println("the poped element is:");

cs.pop();

System.***out***.println("satack after popping element is:");

cs.display();

}

}

**Linkedlist**

* **Single linkedlist**

**(insert at beginning, insert at end)**

Program:

package Linkedlist;

import java.util.\*;

public class circularLinkedList

{

public class Node

{

String data;

Node next;

Node(String str)

{

data=str;

next=null;

}

}

Node head=null;

Node tail=null;

public void insertdata(String str)

{

Node nn=new Node(str);

if(head==null)

{

head=nn;

tail=nn;

nn.next=head;

}

else

{

tail.next=nn;

tail=nn;

tail.next=head;

}

}

public void insertatbegining(String str)

{

Node nn=new Node(str);

if(head==null)

{

head=nn;

tail=nn;

tail.next=head;

}

else

{

nn.next=head;

head=nn;

tail.next=head;

}

}

public void insertatend(String str)

{

Node nn=new Node(str);

if(head==null)

{

head=nn;

tail=nn;

tail.next=head;

}

else

{

tail.next=nn;

tail=nn;

tail.next=head;

}

}

public void display()

{

Node current=head;

if(head==null)

{

System.***out***.println("empty linkedlist");

}

else {

do

{

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

current=current.next;

}while(current!=head);

System.***out***.println();

}

}

public void deleteatbegining()

{

if(head==null)

{

System.***out***.println("linkedlist is empty");

}

if(head==tail)

{

head=null;

tail=null;

}

else

{

head=head.next;

tail.next=head;

}

}

public void deleteatend()

{

if(head==null)

{

System.***out***.println("Linkedlist is empty");

}

if(head==tail)

{

head=null;

tail=null;

}

else

{

tail.next=null;

}

}

public static void main(String args[])

{

circularLinkedList cll=new circularLinkedList();

Scanner sc=new Scanner(System.***in***);

int n=sc.nextInt();

String k="";

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

{

k=sc.nextLine();

cll.insertdata(k);

}

System.***out***.println("Linkedlist before inserting");

cll.display();

String s=sc.nextLine();

System.***out***.println("Linkedlist after inserting");

cll.insertatbegining(s);

cll.display();

String s1=sc.nextLine();

System.***out***.println("Linkedlist after inserting at end");

cll.insertatend(s1);

cll.display();

}

}

* **Double Linkedlist**

**(insert at beginning, insert at end, delete at beginning, delete at end)**

Program:

package Linkedlist;

import java.util.\*;

public class doublelinkedlist

{

public class Node

{

String data;

Node prev;

Node next;

Node(String str)

{

data=str;

prev=null;

next=null;

}

}

Node head=null;

Node tail=null;

public void insertdata(String str)

{

Node nn=new Node(str);

if(head==null)

{

head=nn;

tail=nn;

head.prev=null;

tail.next=null;

}

else

{

tail.next=nn;

nn.prev=tail;

tail=nn;

tail.next=null;

}

}

public void insertbegin(String str)

{

Node nn=new Node(str);

if(head==null)

{

head=nn;

tail=nn;

head.prev=null;

tail.next=null;

}

else

{

nn.next=head;

head.prev=nn;

head=nn;

head.prev=null;

}

}

public void insertatend(String str)

{

Node nn=new Node(str);

if(head==null)

{

head=nn;

tail=nn;

head.prev=null;

tail.next=null;

}

else

{

nn.prev=tail;

tail.next=nn;

tail=nn;

tail.next=null;

}

}

public void deleteatbegining()

{

if(head==null)

{

System.***out***.println("linkedlist is empty");

return;

}

if(head==tail)

{

head=null;

tail=null;

}

else

{

head=head.next;

head.prev=null;

}

}

public void deleteatend()

{

if(tail==null)

{

System.***out***.println("linkedlist is empty");

return;

}

if(head==tail)

{

head=null;

tail=null;

}

else

{

tail=tail.prev;

tail.next=null;

}

}

public void display()

{

Node current=head;

if(head==null)

{

System.***out***.println("Linkedlist is empty");

}

//System.out.println("Doubly linkedlist is:");

while(current!=null)

{

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

current=current.next;

}

System.***out***.println();

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.***in***);

doublelinkedlist dll=new doublelinkedlist();

int n=sc.nextInt();

sc.nextLine();

String k="";

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

{

k=sc.nextLine();

dll.insertdata(k);

}

System.***out***.println("Linkedlist befor inserting element at begining:");

dll.display();

String s=sc.nextLine();

System.***out***.println("Linkedlist after inserting element at begining");

dll.insertbegin(s);

dll.display();

String s1=sc.nextLine();

System.***out***.println("Linkedlist after inserting element at end");

dll.insertatend(s1);

dll.display();

System.***out***.println("Linkedlist after deleting at beging:");

dll.deleteatbegining();

dll.display();

System.***out***.println("Linkedlist after delecting at end");

dll.deleteatend();

dll.display();

}

}