|  |  |
| --- | --- |
| **Ex.No: 1**  **Date:** | **Implementation of stack ADT using Linked List** |

**AIM:**

To write a Java program for the implementation of stack using Linked list.

**FLOW CHART:**

DEFINE STRUCTURE FOR STACK POINTER

DEFINE PUSH()

DEFINE POP()

DEFINE DISPLAY()

READ CHOICE ch

Switch()

EXIT ()

DISPLAY ()

POP ()

PUSH ()

BREAK

BREAK

BREAK

BREAK

/\* STACK ADT\*/

import java.io.\*;

import java.lang.\*;

class Node

{

int data;

Node next;

}

class Stack

{

private Node top;

public Stack()

{

this.top = null;

}

public void push(int x)

{

Node node = new Node();

if (node == null)

{

System.out.print("\nHeap Overflow");

return;

}

System.out.println("Inserting " + x);

node.data = x;

node.next = top;

top = node;

}

public int peek()

{

if (top!=null) {

return top.data;

}

else {

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

return -1;

}

}

public void pop()

{

if (top == null)

{

System.out.print("\nStack Underflow");

return;

}

System.out.println("Removing " + peek());

top = (top).next;

}

void display()

{

if (top == null) {

System.out.printf("\nStack Underflow\n");

System.exit(1);

}

else {

Node node = top;

while (node != null) {

System.out.printf("%d->", node.data);

node = node.next;

}

}

}

}

class stackadt

{

public static void main(String[] args)throws IOException

{

Stack stack = new Stack();

int ch,s;

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

do

{

System.out.println("\nEnter your choice:\n\t1.PUSH \n\t2.POP \n\t3.DISPLAY

\n\t4.EXIT");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter the item to be inserted:");

s=Integer.parseInt(br.readLine());

stack.push(s);

System.out.println("The top element is " + stack.peek());

break;

case 2:

stack.pop();

break;

case 3:

stack.display();

break;

}

} while(ch!=4);

}

}

**OUTPUT:**

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

1

Enter the item to be inserted:

4

Inserting 4

The top element is 4

Enter the item to be inserted:

4

Inserting 4

The top element is 4

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

1

Enter the item to be inserted:

9

Inserting 9

The top element is 9

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

1

Enter the item to be inserted:

6

Inserting 6

The top element is 6

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

3

6->9->4->

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

2

Removing 6

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

3

9->4->

Enter your choice:

1. PUSH

2. POP

3. DISPLAY

4. EXIT

4

RESULT:

Thus the program for **STACK ADT** using Linked list has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 2**  **Date:** | **Implementation of Queue ADT using Linked List** |

**AIM:**

To write a JAVA program for QUEUE ADT using Linked List.

**FLOW CHART:**

WHILE (1)

READ CHOICE USING C

DEFINE DISPLAY

DEFINE DEQUEUE

DEFINE ENQUEUE

DEFINE NODE FOR QUEUE USING LIST

SWITCH

EXIT

VIEW

DEL

ADD

READ CHOICE

BREAK

BREAK

BREAK

BREAK

/\*QUEUE ADT USING LINKED LIST\*/

import java.io.\*;

import java.lang.\*;

class Node

{

int data;

Node next;

public Node(int data)

{

this.data = data;

this.next = null;

}

}

class Queue

{

private static Node rear = null, front = null;

public static int dequeue()

{

if (front == null)

{

System.out.print("\nQueue Underflow");

System.exit(1);

}

Node temp = front;

System.out.printf("Removing %d\n", temp.data);

front = front.next;

if (front == null) {

rear = null;

}

int item = temp.data;

return item;

}

public static void enqueue(int item)

{

Node node = new Node(item);

System.out.printf("Inserting %d\n", item);

if (front == null)

{

front = node;

rear = node;

}

else {

rear.next = node;

rear = node;

}

}

public static int peek()

{

if (front != null) {

return front.data;

}

else {

System.exit(1);

}

return -1;

}

public static boolean isEmpty() {

return rear == null && front == null;

}

static void displayQueue(Queue q){

Node temp = q.front;

System.out.printf("\nElements in Circular Queue are: ");

while (temp.next != null) {

System.out.printf("%d ", temp.data);

temp = temp.next;

}

System.out.printf("%d", temp.data);

}

}

class queueadt

{

public static void main(String[] args)throws IOException

{

Queue q = new Queue();

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int ch,s;

do

{

System.out.println("\nEnteryour

choice:\n\t1.Enqueue\n\t2.Dequeue\n\t3.Display");

System.out.println("\nEnter your chice:");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter the element for queue:");

s=Integer.parseInt(br.readLine());

q.enqueue(s);

System.out.printf("The front element is %d\n", q.peek());

break;

case 2:

q.dequeue();

break;

case 3:

q.displayQueue(q);

break;

}

}while(ch!=4);

}

}

**OUTPUT:**

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

1

Enter the element for queue:

3

Inserting 3

The front element is 3

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

1

Enter the element for queue:

5

Inserting 5

The front element is 3

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

1

Enter the element for queue:

7

Inserting 7

The front element is 3

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

3

Elements in Circular Queue are: 3 5 7

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

2

Removing 3

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

3

Elements in Circular Queue are: 5 7

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

4

**RESULT:**

Thus the Java program for queue ADT using linked list has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 3**  **Date:** | **Implementation of CIRCULAR Queue ADT using List** |

**AIM:**

To write a JAVA program for CIRCULAR QUEUE ADT using Linked List.

**FLOW CHART:**

DEFINE LIST FOR CIRCULAR QUEUE NODE LIST

WHILE (1)

READ CHOICE USING C

DEFINE DISPLAY

DEFINE DEQUEUE

DEFINE ENQUEUE

READ CHOICE

BREAK

EXIT

DISPLAY

ENQUEUE

SWITCH

BREAK

BREAK

BREAK

DEQUEUE

/\* CIRCULAR QUEUE USING LINKED LIST\*/

import java.util.\*;

import java.io.\*;

class CQueue {

static class Node {

int data;

Node link;

}

static class Queue {

Node front, rear;

}

static void enQueue(Queue q, int value)

{

Node temp = new Node();

temp.data = value;

if (q.front == null)

q.front = temp;

else

q.rear.link = temp;

q.rear = temp;

q.rear.link = q.front;

}

static void deQueue(Queue q)

{

if (q.front == null) {

System.out.printf("Queue is empty");

}

int value; // Value to be dequeued

if (q.front == q.rear) {

value = q.front.data;

q.front = null;

q.rear = null;

}

else

{

Node temp = q.front;

value = temp.data;

q.front = q.front.link;

q.rear.link = q.front;

}

System.out.println("The Dequeue element is:"+value);

}

static void displayQueue(Queue q)

{

Node temp = q.front;

System.out.printf("\nElements in Circular Queue are: ");

while (temp.link != q.front) {

System.out.printf("%d ", temp.data);

temp = temp.link;

}

System.out.printf("%d", temp.data);

}

public static void main(String args[])throws IOException

{

Queue q = new Queue();

q.front = q.rear = null;

CQueue c=new CQueue();

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int ch,s;

do

{

System.out.println("\nEnteryour

choice:\n\t1.Enqueue\n\t2.Dequeue\n\t3.Display");

System.out.println("\nEnter your chice:");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter the element for queue:");

s=Integer.parseInt(br.readLine());

c.enQueue(q,s);

break;

case 2:

c.deQueue(q);

break;

case 3:

c.displayQueue(q);

break;

}

}while(ch!=4);

}

}

**OUTPUT:**

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

1

Enter the element for queue:

4

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

1

Enter the element for queue:

6

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

3

Elements in Circular Queue are: 2 4 6

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

2

The Dequeue element is: 2

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

3

Elements in Circular Queue are: 4 6

Enter your choice:

1. Enqueue

2. Dequeue

3. Display

Enter your choice:

4

**RESULT:**

Thus the java program for circular queue using linked list has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 4**  **Date:** | **Implementation of Infix expression into postfix expression** |

**AIM:**

To write a JAVA program for implementation of Infix expression into postfix expression.

**FLOW CHART:**

DEFINE gotOper(CHAR C)

EXCECUTE CONVERT EXPRESSION USING ITEM

READ INFIX EXPRESSION USING STR

DEFINE CONVERT

DEFINE doTrans()

DEFINE INPUTEXPR ()

DEFINE STRUCTURE FOR EXPRESSION

/\*Converting INFIX to POSTFIX Expression\*/

import java.io.\*;

public class InToPost {

private Stack theStack;

private String input;

private String output = "";

public InToPost(String in) {

input = in;

int stackSize = input.length();

theStack = new Stack(stackSize);

}

public String doTrans() {

for (int j = 0; j < input.length(); j++) {

char ch = input.charAt(j);

switch (ch) {

case '+':

case '-':

gotOper(ch, 1);

break;

case '\*':

case '/':

gotOper(ch, 2);

break;

case '(':

theStack.push(ch);

break;

case ')':

gotParen(ch);

break;

default:

output = output + ch;

break;

}

}

while (!theStack.isEmpty()) {

output = output + theStack.pop();

}

return output;

}

public void gotOper(char opThis, int prec1) {

while (!theStack.isEmpty()) {

char opTop = theStack.pop();

if (opTop == '(') {

theStack.push(opTop);

break;

} else {

int prec2;

if (opTop == '+' || opTop == '-')

prec2 = 1;

else

prec2 = 2;

if (prec2 < prec1) {

theStack.push(opTop);

break;

}

Else

output = output + opTop;

}

}

theStack.push(opThis);

}

public void gotParen(char ch) {

while (!theStack.isEmpty()) {

char chx = theStack.pop();

if (chx == '(')

break;

else output = output + chx;

}

}

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the INFIX string:");

String input =br.readLine();

String output;

InToPost theTrans = new InToPost(input);

output = theTrans.doTrans();

System.out.println("Postfix is " + output + '\n');

}

class Stack {

private int maxSize;

private char[] stackArray;

private int top;

public Stack(int max) {

maxSize = max;

stackArray = new char[maxSize];

top = -1;

}

public void push(char j) {

stackArray[++top] = j;

}

public char pop() {

return stackArray[top--];

}

public char peek() {

return stackArray[top];

}

public boolean isEmpty() {

return (top == -1);

}

}

}

**OUTPUT:**

C:\Program Files\Java\jdk1.8.0\_301\bin>javac InToPost.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java InToPost

Enter the INFIX string:

1\*2+(4-5)

Postfix is 12\*45-+

**RESULT:**

Thus the Java program to convert the Infix to postfix expression has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 5**  **Date:** | **Evaluate the POSTFIX Expression** |

**AIM:**

To write a JAVA program for evaluate the postfix expression.

**FLOW CHART:**

DEFINE CLASS FOR POSTFIX EXPRESSION

DEFINE POSTFIX

READ CHOICE

DEFINE PUSH(INT)

DEFINE POP()

DEFINE isdigit()

WHILE (1)

READ POST [I]

IF

POP

PUSH

RESULT

/\*Evaluate the postfix expression\*/

import java.util.\*;

class postfix1

{

double a[]=new double[20];

int top;

postfix1() { top=-1; }

void push(double x)

{

top++;

a[top]=x;

}

double pop()

{

if(top==-1) { System.out.println("\nImproper expression !"+(char)7); System.exit(1); }

return a[top--];

}

boolean isdigit(char x)

{

return(x>='0' & x<='9');

}

double val(double v1,double v2,char x)

{

switch(x)

{

case '+': return v1+v2;

case '-': return v1-v2;

case '\*': return v1\*v2;

case '/': return v1/v2;

case '$': return (Math.pow(v1,v2));

default : System.out.println("Improper Operation");

System.exit(1);

}

return 1;

}

double eval(char b[])

{

double v1,v2,v3,v;

for(int i=0;i<b.length;i++)

{

if(isdigit(b[i]))

push((double)b[i]-48);

else

{

v2=pop();

v1=pop();

v=val(v1,v2,b[i]);

push(v);

}

}

if(top!=0) { System.out.println("\nImproper expression"+(char)7); System.exit(1);

}

return pop();

}

}

class Postfix

{

public static void main(String args[])

{

postfix1 p=new postfix1();

Scanner scr=new Scanner(System.in);

System.out.print("Enter the post fix expression: ");

String str=scr.next();

char b[]=str.toCharArray();

System.out.print("\nValue of the expression is: "+p.eval(b));

}

}

**OUTPUT:**

C:\Program Files\Java\jdk1.8.0\_301\bin>javac Postfix.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java Postfix

Enter the post fix expression: 23-4+567\*+\*

Value of the expression is: 141.0

C:\Program Files\Java\jdk1.8.0\_301\bin>javac Postfix.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java Postfix

Enter the post fix expression: 138\*+

Value of the expression is: 25.0

**RESULT:**

Thus the Java program to evaluate the postfix expression has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 9**  **Date:** | **Breadth First Search for GRAPH** |

**AIM:**

To write a JAVA program to the implementation of BFS for a given graph.

**FLOW CHART:**

DEFINE CLASSFOR BFS

DISPLAY RESULT FOR BFS

TRAVERSE THE GRAPH USING BFS ALGORITHM

ENTER THE NO OF EDGES

ENTER THE NO OF VERTICES

DEFINE addEdge(int v,int w),

DFS(int s) methods.

/\*BREADTH-FIRST SEARCH GRAPH\*/

import java.io.\*;

import java.util.\*;

class bfstrue

{

private int V; // No. of vertices

private LinkedList<Integer> adj[]; //Adjacency Lists

bfstrue(int v)

{

V = v;

adj = new LinkedList[v];

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

adj[i] = new LinkedList();

}

void addEdge(int v,int w)

{

adj[v].add(w);

}

void BFS(int s)

{

boolean visited[] = new boolean[V];

LinkedList<Integer> queue = new LinkedList<Integer>();

visited[s]=true;

queue.add(s);

while (queue.size() != 0)

{

s = queue.poll();

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

Iterator<Integer> i = adj[s].listIterator();

while (i.hasNext())

{

int n = i.next();

if (!visited[n])

{

visited[n] = true;

queue.add(n);

}

}

}

}

public static void main(String args[])throws IOException

{

bfstrue g = new bfstrue(8);

int edge,i,a,b;

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the no of edges in your graph:");

edge=Integer.parseInt(br.readLine());

for(i=1;i<=edge;i++)

{

System.out.println("Enter the node connecting with other edge:");

a=Integer.parseInt(br.readLine());

b=Integer.parseInt(br.readLine());

g.addEdge(a,b);

}

System.out.println("Following is Breadth First Traversal "+

"(starting from vertex 0)");

g.BFS(0);

}

}

**OUTPUT:**

Enter the no of edges in your graph:

6

Enter the node connecting with other edge:

0

1

Enter the node connecting with other edge:

0

2

Enter the node connecting with other edge:

1

3

Enter the node connecting with other edge:

1

4

Enter the node connecting with other edge:

2

5

Enter the node connecting with other edge:

2

6

Following is Breadth First Traversal (starting from vertex 0)

0 1 2 3 4 5 6

**RESULT:**

Thus the Java program for Breadth first search has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 10**  **Date:** | **Depth First Search for GRAPH** |

**AIM:**

To write a JAVA program to the implementation of DFS for a given graph.

**FLOW CHART:**

DEFINE CLASSFOR DFS

DEFINE addEdge(int v,int w),

DFS(int s) methods.

ENTER THE NO OF VERTICES

ENTER THE NO OF EDGES

TRAVERSE THE GRAPH USING DFS ALGORITHM

DISPLAY RESULT FOR DFS

/\*DEPTH FIRST TRAVERSAL\*/

import java.util.\*;

import java.io.\*;

class dfsgra

{

private int V; // No. of vertices

private LinkedList<Integer> adj[];

dfsgra(int v)

{

V = v;

adj = new LinkedList[v];

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

adj[i] = new LinkedList();

}

void addEdge(int v, int w)

{

adj[v].add(w); // Add w to v's list.

}

void DFSUtil(int v, boolean visited[])

{

visited[v] = true;

System.out.print(v + " ");

Iterator<Integer> i = adj[v].listIterator();

while (i.hasNext())

{

int n = i.next();

if (!visited[n])

DFSUtil(n, visited);

}

}

void DFS(int v)

{

boolean visited[] = new boolean[V];

DFSUtil(v, visited);

}

public static void main(String args[])throws IOException

{

dfsgra g = new dfsgra(8);

int edge,i,a,b;

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the no of edges in your graph:");

edge=Integer.parseInt(br.readLine());

for(i=1;i<=edge;i++)

{

System.out.println("Enter the node connecting with other edge:");

a=Integer.parseInt(br.readLine());

b=Integer.parseInt(br.readLine());

g.addEdge(a,b);

}

System.out.println(

"Following is Depth First Traversal "

+ "(starting from vertex 0)");

g.DFS(0);

}

}

**OUTPUT:**

Enter the no of edges in your graph:

6

Enter the node connecting with other edge:

0

1

Enter the node connecting with other edge:

0

2

Enter the node connecting with other edge:

1

3

Enter the node connecting with other edge:

1

4

Enter the node connecting with other edge:

2

5

Enter the node connecting with other edge:

2

6

Following is Depth First Traversal (starting from vertex 0)

0 1 3 4 2 5 6

**RESULT:**

Thus the Java program for Depth –first traversal has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 8**  **Date:** | **SEARCH an element in BINARY SEARCH TREE** |

**AIM:**

To write a JAVA program to search a key element in a binary search tree.

**FLOW CHART:**

DISPLAY RESULT FOR search

Search the node using given key in a tree

GET the values for Nodes

ENTER THE NO of nodes

DEFINE insert(node root,int key),

search(node root,int key,node parent) methods.

DEFINE CLASS FOR searchtree

/\*Search a node for a given key in BST\*/

import java.io.\*;

class Node

{

int data;

Node left = null, right = null;

Node(int data) {

this.data = data;

}

}

class searchtree

{

public static Node insert(Node root, int key)

{

if (root == null) {

return new Node(key);

}

if (key < root.data) {

root.left = insert(root.left, key);

}

else {

root.right = insert(root.right, key);

}

return root;

}

public static void search(Node root, int key, Node parent)

{

if (root == null)

{

System.out.print("Key Not found");

return;

}

if (root.data == key)

{

if (parent == null) {

System.out.print("The node with key " + key + " is root node");

}

else if (key < parent.data)

{

System.out.print("The given key is the left node of the node " +

"with key " + parent.data);

}

else {

System.out.print("The given key is the right node of the node " +

"with key " + parent.data);

}

return;

}

if (key < root.data) {

search(root.left, key, root);

}

else {

search(root.right, key, root);

}

}

public static void main(String[] args)throws IOException

{

int[] keys = new int[10];

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter how many nodes in tree");

int n=Integer.parseInt(br.readLine());

Node root = null;

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

System.out.println("Enter the node value:");

keys[i]=Integer.parseInt(br.readLine());

}

for (int key:keys){

root = insert(root, key);

}

System.out.println("Enter the node to search:");

int x=Integer.parseInt(br.readLine());

search(root, x, null);

}

}

**OUTPUT:**

C:\Program Files\Java\jdk1.8.0\_301\bin>javac searchtree.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java searchtree

Enter how many nodes in tree

5

Enter the node value:

23

Enter the node value:

10

Enter the node value:

15

Enter the node value:

34

Enter the node value:

45

Enter the node value:

19

Enter the node to search:

19

The given key is the right node of the node with key 15

C:\Program Files\Java\jdk1.8.0\_301\bin>javac searchtree.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java searchtree

Enter how many nodes in tree

5

Enter the node value:

23

Enter the node value:

10

Enter the node value:

15

Enter the node value:

34

Enter the node value:

45

Enter the node value:

19

Enter the node to search:

24

Key Not found

C:\Program Files\Java\jdk1.8.0\_301\bin>

**RESULT:**

Thus the java program to search a key element in a binary search tree has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 6**  **Date:** | **INSERT an element in BINARY SEARCH TREE** |

**AIM:**

To write a JAVA program to insert an element in a binary search tree.

**FLOW CHART:**

DISPLAY node in inorder

Insert the node using given key in a tree

GET the values for Nodes

ENTER THE NO of nodes

DEFINE insert(node root,int key),

inorder(root) methods.

DEFINE CLASS FOR inserttree

import java.io.\*;

import java.lang.\*;

class Node

{

int data;

Node left = null, right = null;

Node(int data) {

this.data = data;

}

}

class inserttree

{

public static Node insert(Node root, int key)

{

if (root == null) {

return new Node(key);

}

if (key < root.data) {

root.left = insert(root.left, key);

}

else {

root.right = insert(root.right, key);

}

return root;

}

public static void inorder(Node root)

{

if (root == null) {

return;

}

inorder(root.left);

System.out.print(root.data +" ");

inorder(root.right);

}

public static void main(String[] args)throws IOException

{

int n;

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter how many nodes in tree");

n=Integer.parseInt(br.readLine());

int[] keys = new int[n];

Node root = null;

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

{

System.out.println("Enter the "+i+" node value:");

keys[i]=Integer.parseInt(br.readLine());

}

for (int key:keys)

{

root = insert(root, key);

}

System.out.println("The inserted nodes in inorder:");

inorder(root);

}

}

**OUTPUT:**

C:\Program Files\Java\jdk1.8.0\_301\bin>javac inserttree.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java inserttree

Enter how many nodes in tree

5

Enter the 0 node value:

34

Enter the 1 node value:

23

Enter the 2 node value:

12

Enter the 3 node value:

16

Enter the 4 node value:

17

The inserted nodes in inorder:

12 16 17 23 34

**RESULT:**

Thus the java program for inserting node in a Binary search tree has been executed successfully.

|  |  |
| --- | --- |
| **Ex.No: 7**  **Date:** | **DELETE an element in BINARY SEARCH TREE** |

**AIM:**

To write a JAVA program to delete an element in a binary search tree.

**FLOW CHART:**

DEFINE insert(node root, int key),

inorder(root),

deleteNode(Node root, int key) methods.

DEFINE CLASS FOR DELtree

ENTER THE NO of nodes

Insert the node using given key in a tree

GET the values for Nodes

DISPLAY node in inorder

s

/\* DELETE node in a tree\*/

import java.io.\*;

class Node

{

int data;

Node left = null, right = null;

Node(int data)

{

this.data = data;

}

}

class DELtree

{

public static void inorder(Node root)

{

if (root == null)

{

return;

}

inorder(root.left);

System.out.print(root.data + " ");

inorder(root.right);

}

public static Node findMaximumKey(Node ptr)

{

while (ptr.right != null)

{

ptr = ptr.right;

}

return ptr;

}

public static Node insert(Node root, int key)

{

if (root == null)

{

return new Node(key);

}

if (key < root.data)

{

root.left = insert(root.left, key);

}

else

{

root.right = insert(root.right, key);

}

return root;

}

public static Node deleteNode(Node root, int key)

{

if (root == null)

{

System.out.println("The node value not in the tree"); return null;

}

if (key < root.data)

{

root.left = deleteNode(root.left, key);

}

else if (key > root.data)

{

root.right = deleteNode(root.right, key);

}

else

{

if (root.left == null && root.right == null)

{

return null;

}

else if (root.left != null && root.right != null)

{

Node predecessor = findMaximumKey(root.left);

root.data = predecessor.data;

root.left = deleteNode(root.left, predecessor.data);

}

else

{

Node child = (root.left != null) ? root.left: root.right;

root = child;

}

}

return root;

}

public static void main(String[] args)throws IOException

{

int n,x;

int[] keys = new int[5];

BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

System.out.println("Enter the no of nodes to be inserted:");

n=Integer.parseInt(br.readLine());

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

{

System.out.println("Enther the nodes"+i);

keys[i]=Integer.parseInt(br.readLine());

}

Node root = null;

for (int key: keys)

{

root = insert(root, key);

}

inorder(root);

System.out.println("\nEnter the value of the node to be deleted");

x=Integer.parseInt(br.readLine());

root = deleteNode(root, x);

System.out.println("After delete the node the tree");

inorder(root); }

}

**OUTPUT:**

C:\Program Files\Java\jdk1.8.0\_301\bin>javac DELtree.java

C:\Program Files\Java\jdk1.8.0\_301\bin>java DELtree

Enter the no of nodes to be inserted:

5

Enter the nodes0

23

Enter the nodes1

43

Enter the nodes2

12

Enter the nodes3

25

Enter the nodes4

65

12 23 25 43 65

Enter the value of the node to be deleted

12

After delete the node the tree

23 25 43 65

**RESULT:**

Thus the Java program for implementing delete node in a binary search tree has been executed successfully.