**Binary Tree (Array implementation)**

**Case 1:** (0—n-1)

if (say)father=p;

then left\_son=(2\*p)+1;

and right\_son=(2\*p)+2;

**Case 2:** 1—n

if (say)father=p;

then left\_son=(2\*p);

and right\_son=(2\*p)+1;

import java.io.\*;

import java.lang.\*;

import java.util.\*;

public class Tree {

public static void main(String[] args)

{

Array\_imp obj = new Array\_imp();

obj.Root("A");

obj.set\_Left("B", 0);

obj.set\_Right("C", 0);

obj.set\_Left("D", 1);

obj.set\_Right("E", 1);

obj.set\_Right("F", 2);

obj.print\_Tree();

}

}

class Array\_imp {

static int root = 0;

static String[] str = new String[10];

public void Root(String key) { str[0] = key; }

public void set\_Left(String key, int root)

{

int t = (root \* 2) + 1;

if (str[root] == null) {

System.out.printf(

"Can't set child at %d, no parent found\n",

t);

}

else {

str[t] = key;

}

}

public void set\_Right(String key, int root)

{

int t = (root \* 2) + 2;

if (str[root] == null) {

System.out.printf(

"Can't set child at %d, no parent found\n",

t);

}

else {

str[t] = key;

}

}

public void print\_Tree()

{

for (int i = 0; i < 10; i++) {

if (str[i] != null)

System.out.print(str[i]);

else

System.out.print("-");

}

}

}

Linked Representation of Tree.

class Node {

int data;

Node left;

Node right;

public Node(int data) {

    this.data = data;

}

}

class BinaryTree {

Node root;

public void addNode(int data) {

    Node newNode = new Node(data);

    if (root == null) {

    root = newNode;

    } else {

    Node focusNode = root;

    Node parent;

    while (true) {

        parent = focusNode;

        if (data < focusNode.data) {

        focusNode = focusNode.left;

        if (focusNode == null) {

            parent.left = newNode;

            return;

        }

        } else {

        focusNode = focusNode.right;

        if (focusNode == null) {

            parent.right = newNode;

            return;

        }

        }

    }

    }

}

public void preOrderTraversal(Node focusNode) {

    if (focusNode != null) {

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

    preOrderTraversal(focusNode.left);

    preOrderTraversal(focusNode.right);

    }

}

}

class Main {

public static void main(String[] args) {

    BinaryTree tree = new BinaryTree();

    tree.addNode(50);

    tree.addNode(25);

    tree.addNode(75);

    tree.addNode(12);

    tree.addNode(37);

    tree.addNode(43);

    tree.addNode(30);

    tree.preOrderTraversal(tree.root);

}

}