Nonlinear Data Structure

http://smartlead.hallym.ac.kr

Instructor: **Jin Kim**
010-6267-8189(033-248-2318)

jinkim@hallym.ac.kr
Office Hours:

Lab2(이진트리)

http://smartlead.hallym.ac.kr

Instructor: **Jin Kim**
010-6267-8189(033-248-2318)

jinkim@hallym.ac.kr
Office Hours:

Objectives(실습목표)

- 이진트리를 표현해보자.
- 이진트리의 순회방법을 이해하자.
- 이진트리의 순회를 응용해보자.

Binary Tree

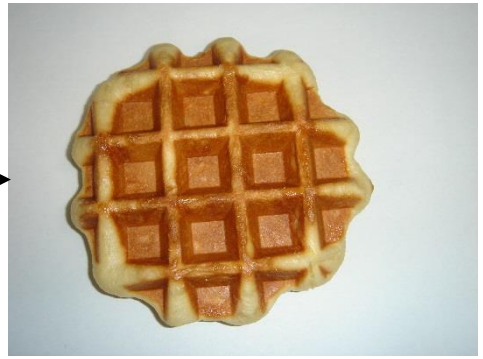
Class waffle (빵틀 와플)
object waffle1(객체 와플1)

Class waffle

Waffle waffle1= new waffle();//와플빵틀에서 waffle1을 찍어라

찍어라

waffle1



Class 붕어빵 (빵틀 붕어빵)
object 붕어빵1(객체 붕어빵1)

```
Class 붕어빵{
  int 머리;
  int 몸통;
  int 꼬리;
  붕어빵(){}
}
```

붕어빵 붕어빵1= new 붕어빵();//붕어빵틀에서 붕어빵1을 찍어라

붕어빵1

붕어빵1.머리
붕어빵1.몸통
붕어빵1.꼬리

Class 호두과자 (빵틀 호두과자)
object 호두과자1(객체 호두과자1)

```
Class 호두과자{
  int 껍질;
  int 속;
호두과자(){}
}
```

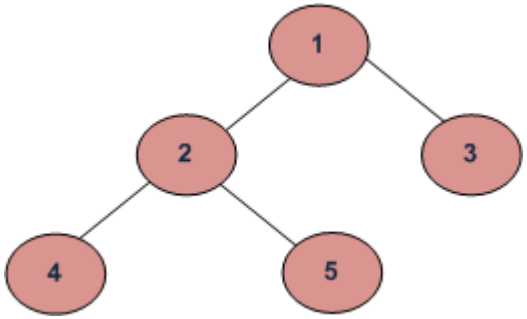
```
호두과자 호두과자1= new 호두과자();
//호두과자틀에서 호두과자1을 찍어라
```

호두과자1 →

호두과자1.껍질←밀가루

호두과자1.속 ← 팥

쉬운버전 프로그래밍



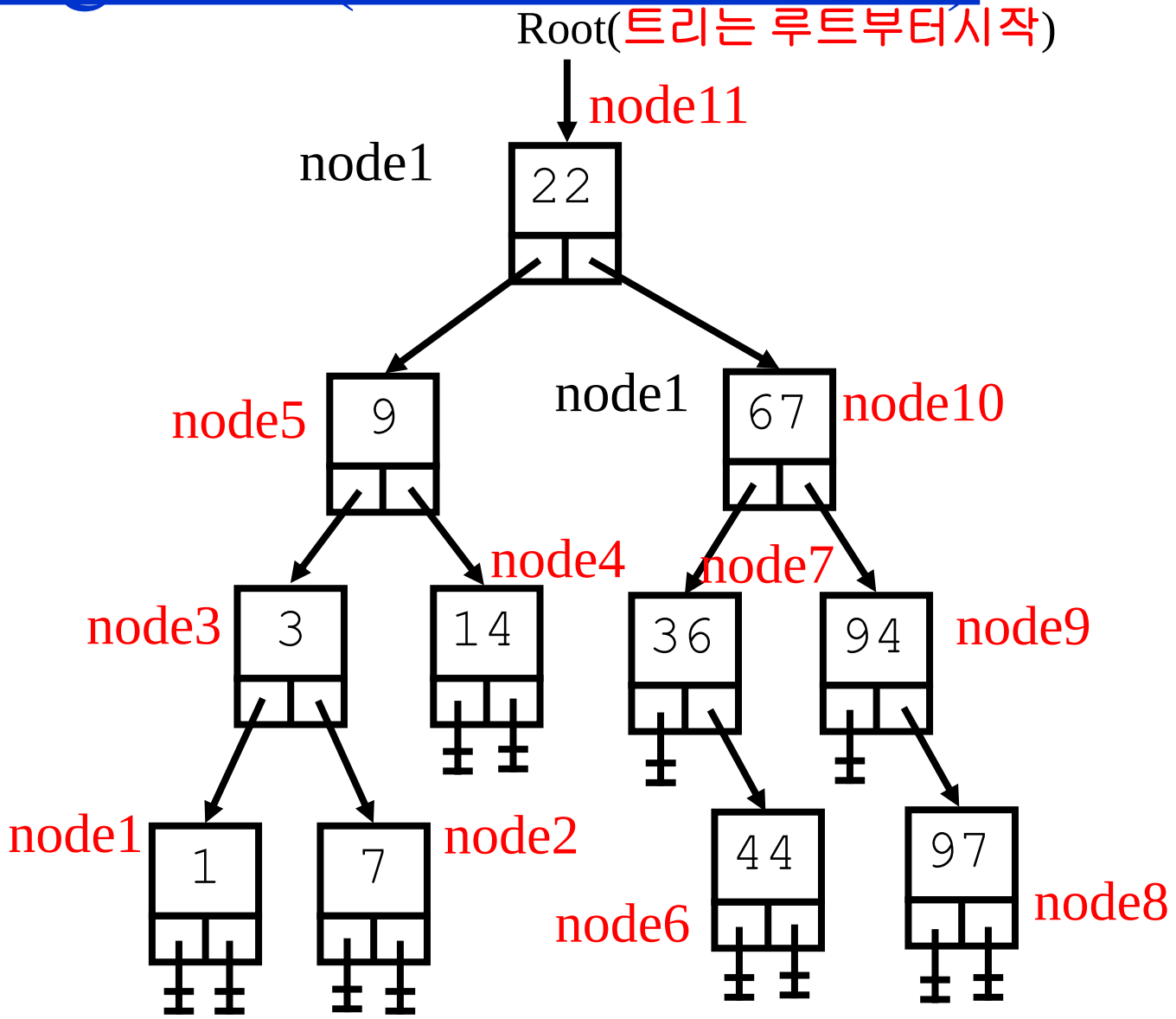
```
Public class BT {
    public  static  void  main(String[] args)
    {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);

        System.out.println(
            "Preorder traversal of binary tree is ");
        tree.printPreorder();

        System.out.println(
            "\nInorder traversal of binary tree is ");
        tree.printInorder();

        System.out.println(
            "\nPostorder traversal of binary tree is ");
        tree.printPostorder();
    }
}
```

```
class BinaryTree {
    // Root of Binary Tree
    Node root;

    BinaryTree() { root = null; }

    /* Given a binary tree, print its nodes according to the
      "bottom-up" postorder traversal. */
    void printPostorder(Node node)
    {
        if (node == null)
            return;

        // first recur on left subtree
        printPostorder(node.left);

        // then recur on right subtree
        printPostorder(node.right);

        // now deal with the node
        System.out.print(node.key + " ");
    }
}
```

```
class Node {
    int key;
    Node left, right;

    public Node(int item)
    {
        key = item;
        left = right = null;
    }
}
```

복잡한 버전 프로그래밍

Program1(let's make tree)



Binary tree tree1

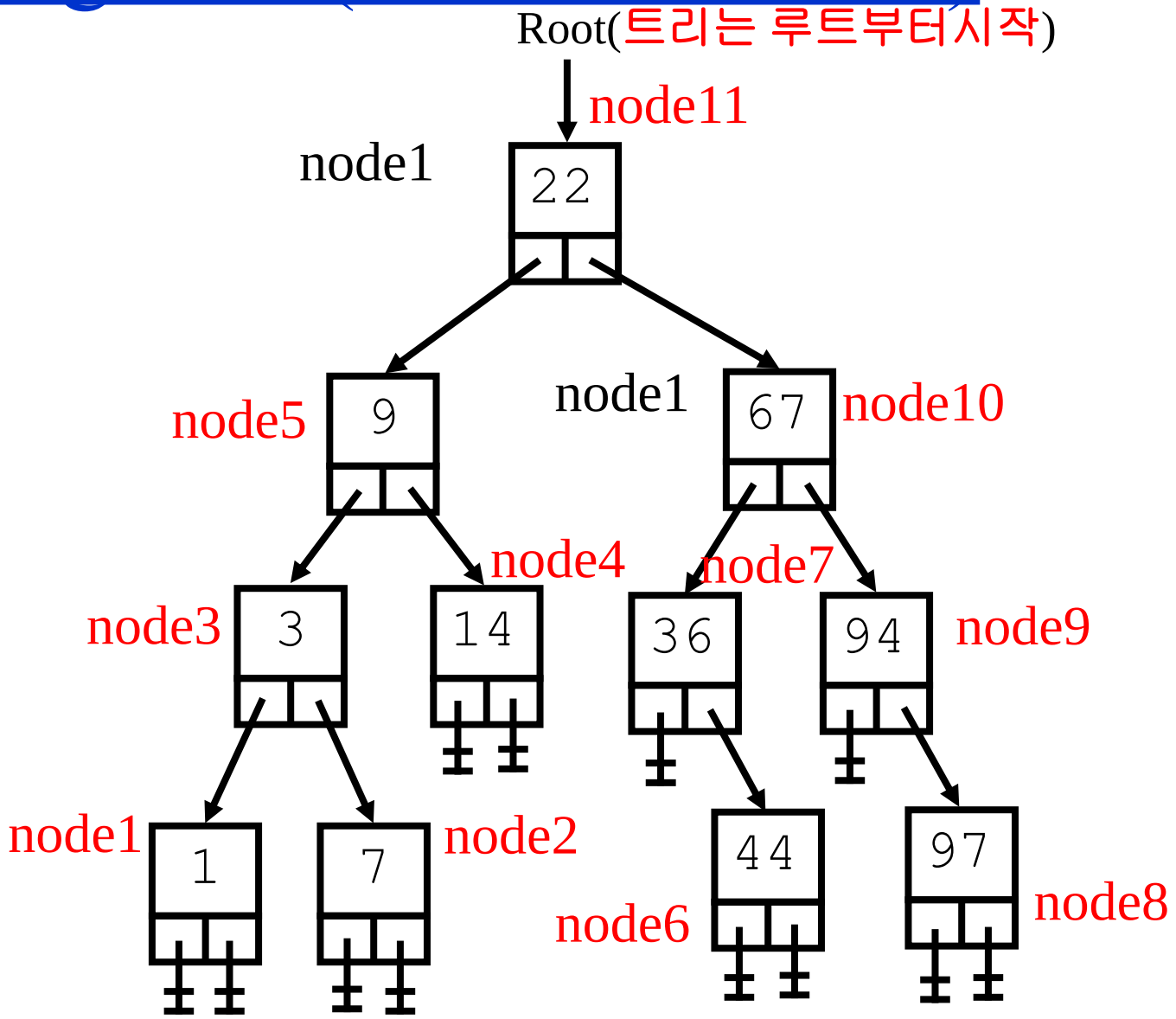
Class BinaryTree

```
Class BinaryTree {
        ….
}

  Public static void main(String[] ar){

  ….
     BinaryTree tree1 = new BinaryTree();
```

BinaryTree tree1 is built

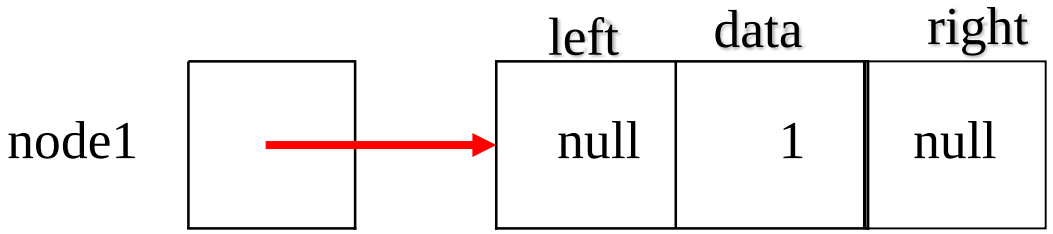
```
class TreeNode {
        int data;
        TreeNode left;
        TreeNode right;

        public TreeNode(int data1){//constructor
                data = data1;
                left=null;
                right=null;
        }
}
```

Program1(let's make tree)



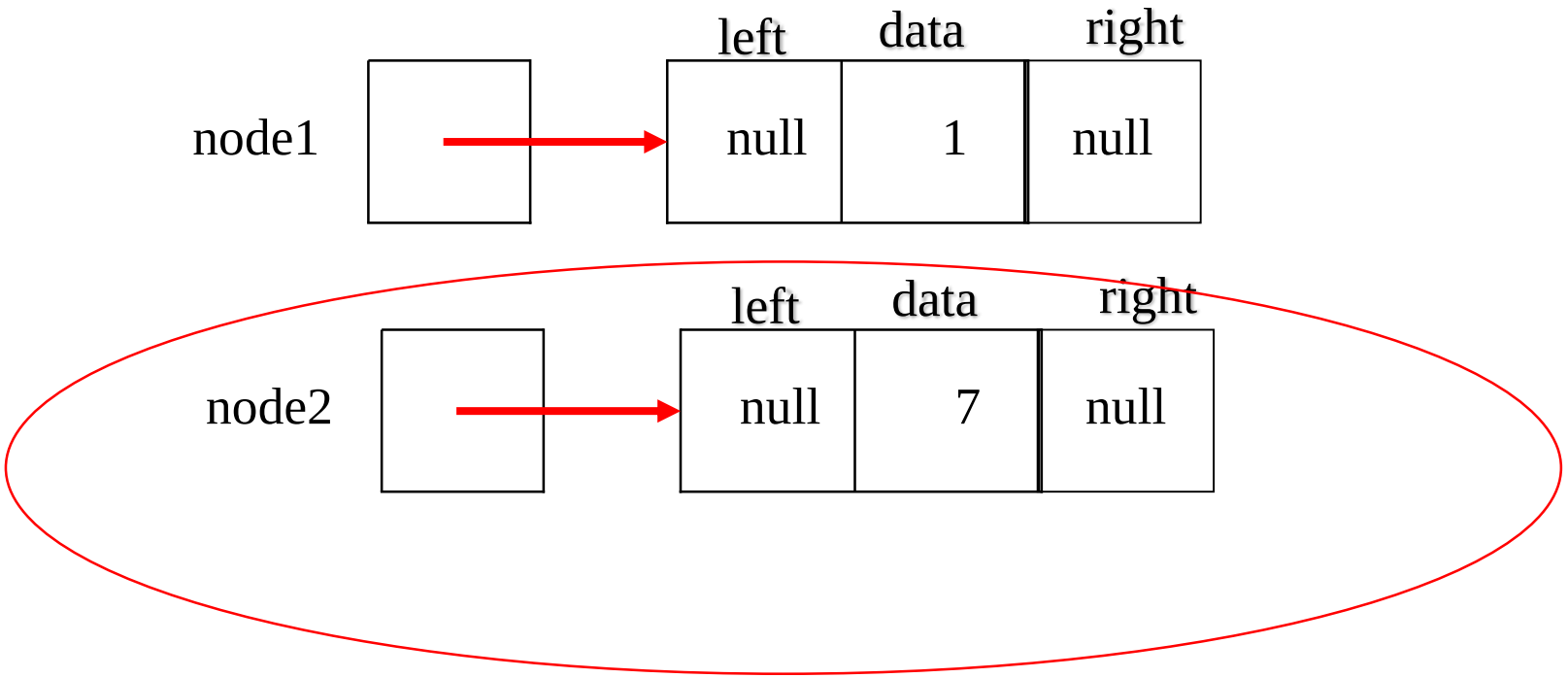
Binary tree tree1

TreeNode node1 = tree1.makenode(null,1,null);
실행후 결과 그림



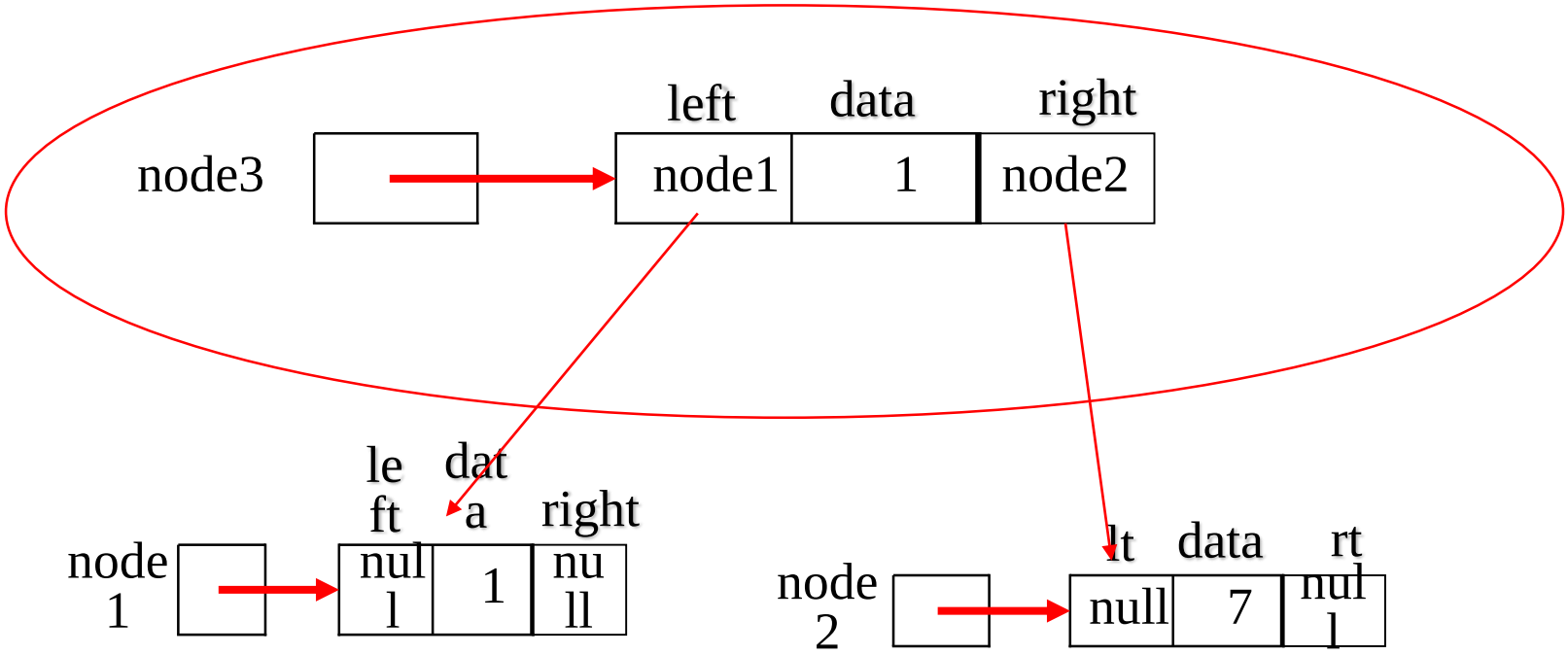
Node1은 TreeNode 빵틀에서 찍은 빵에 대한 주소를 가진다

TreeNode node2 = tree1.makenode(null,7,null);
실행후 결과 그림

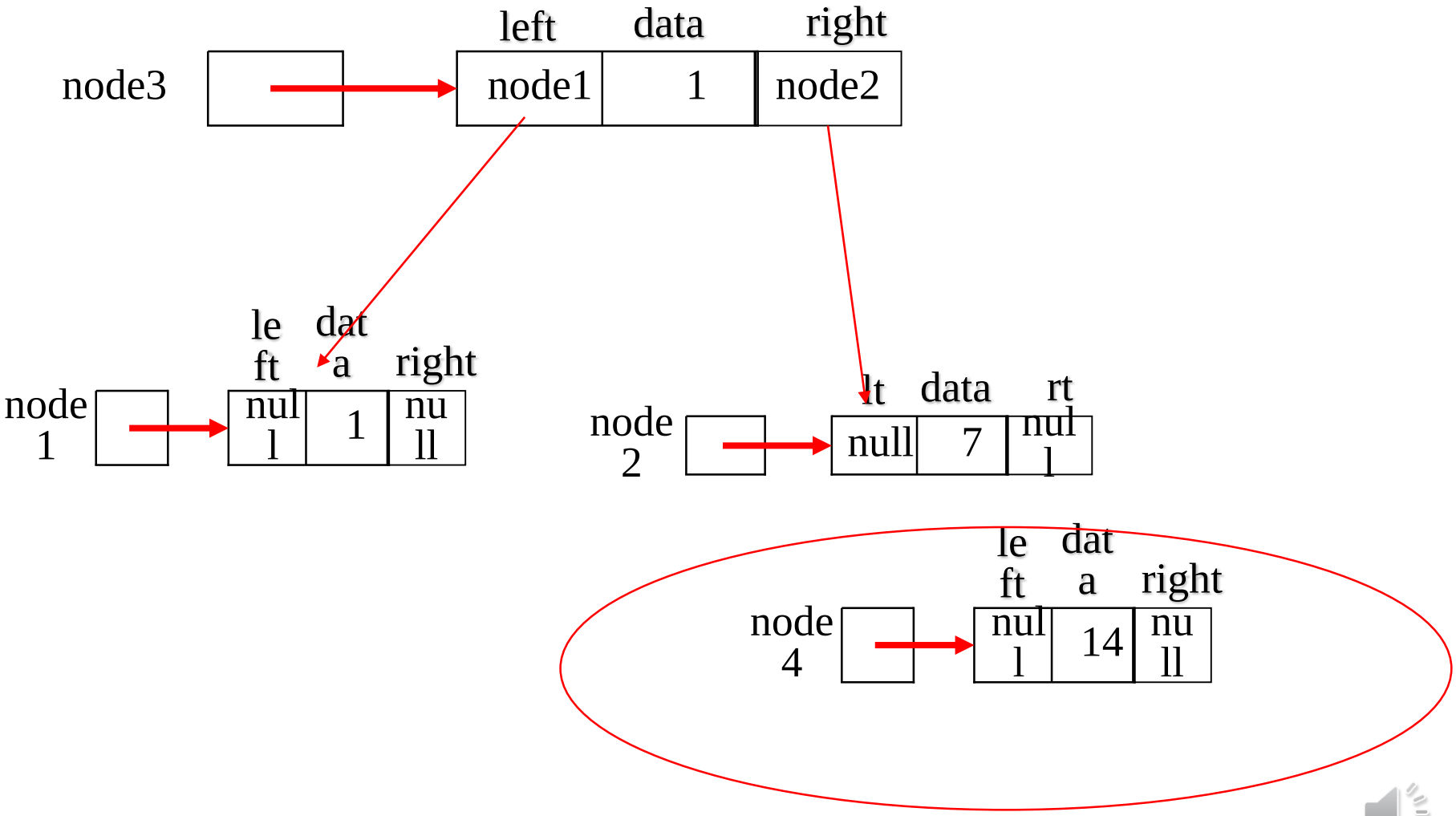


TreeNode node3 = tree1.makenode(node1,3,node2);

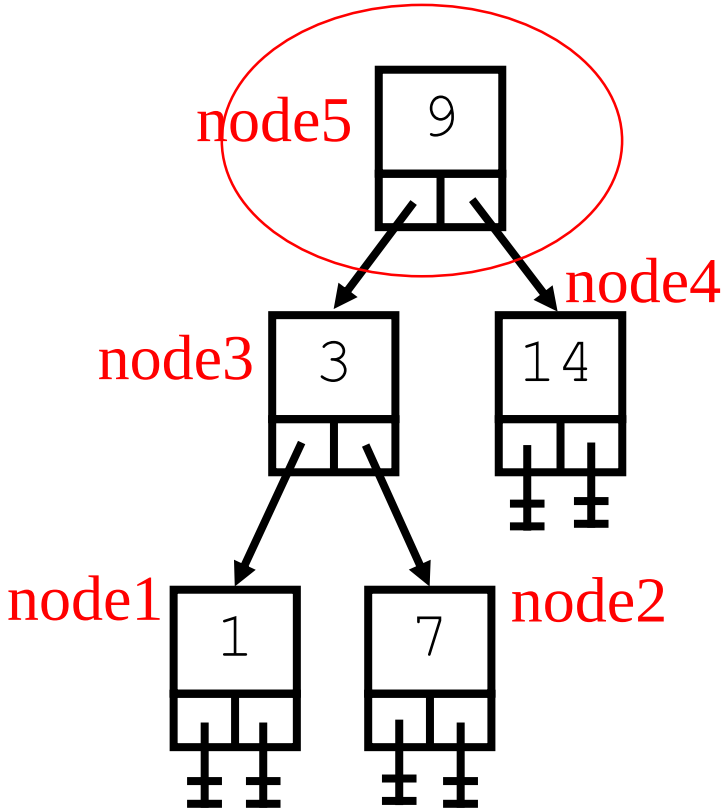
실행후 결과 그림



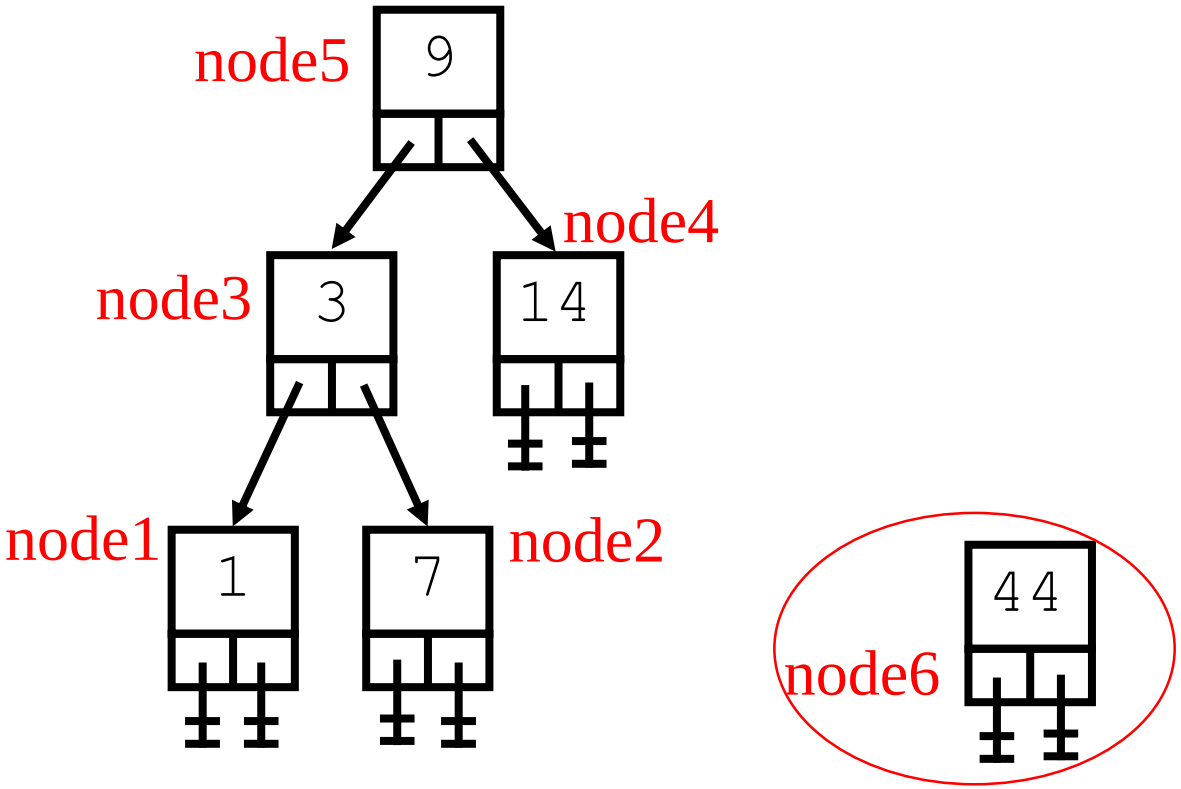
TreeNode node4 = tree1.makenode(null, 14,null);
실행후 결과 그림



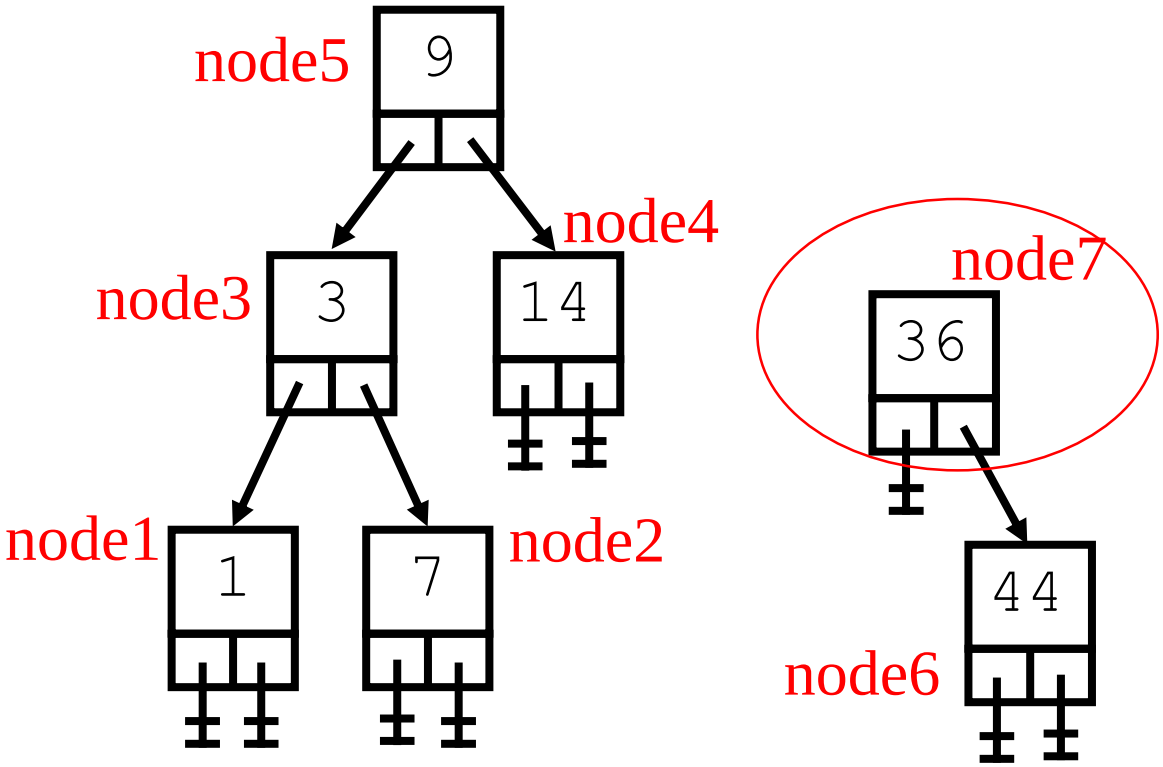
TreeNode node5 = tree1.makenode(null, 14,null);
실행후 결과 그림



TreeNode node6 = tree1.makenode(null, 44,null);
실행후 결과 그림

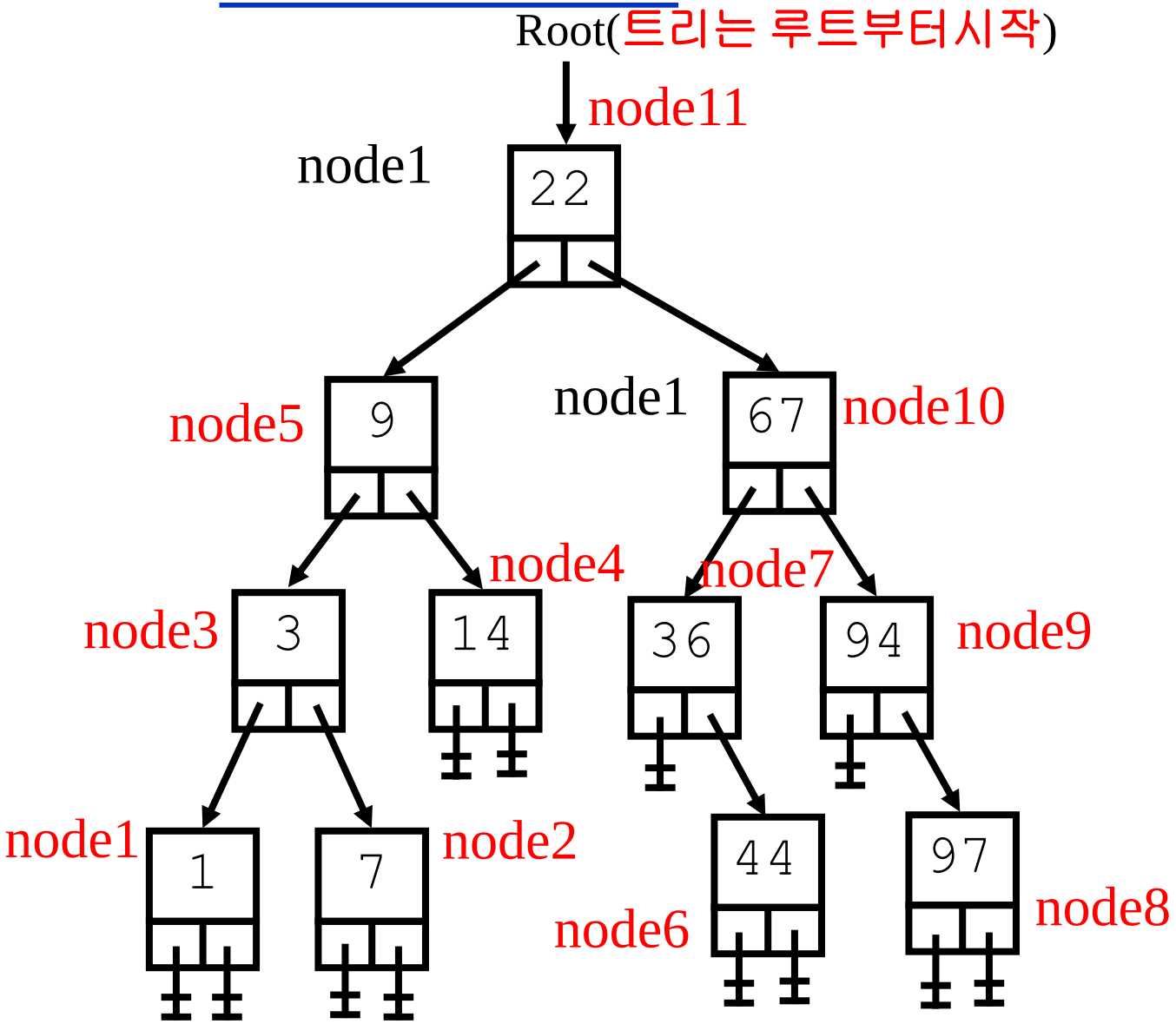


TreeNode node7= tree1.makenode(null, 36,node6);
실행후 결과 그림



Continue 계속 만든다

프로그램1


Root(트리는 루트부터시작)
node11
node1
22
node5
9
node1
67
node10
node4
node7
node3
3
14
36
94
node9
node1
1
7
node2
44
97
node6
node8

이진트리 A

```
public class BT {
        public void BT(){};  //생성자
        public static void main(String[] args){
                BinaryTree A = new BinaryTree();
                TreeNode node1 = A.makenode(null,1,null);
                TreeNode node2 = A.makenode(null,7,null);
                TreeNode node3 = A.makenode(node1,3,node2);
                TreeNode node4 = A.makenode(null,14,null);
                TreeNode node5 = A.makenode(node3,9,node4);
                TreeNode node6 = A.makenode(null,44,null);
                TreeNode node7 = A.makenode(null,36,node6);
                TreeNode node8 = A.makenode(null,97,null);
                TreeNode node9 = A.makenode(null,94,node8);
                TreeNode node10 = A.makenode(node7,67,node9);
                TreeNode node11 = A.makenode(node5,22,node10);
                System.out.println("Preorder");
                A.preorder(node11);
                System.out.println("\n");
                System.out.println("Postorder");
                A.postorder(node11);
                System.out.println("\n");
                System.out.println("Inorder");
                A.inorder(node11);
                System.out.println("\n");
                A.nodecount(node11);

        }
}
```

```
class BinaryTree {
        private TreeNode root;
        public static int count;
        public static int leafcount;
        public TreeNode makenode(TreeNode leftv, int data, TreeNode rightv){
                root = new TreeNode(data);
                root.left=leftv;
                root.right=rightv;
                return root;
        }
        public void preorder(TreeNode root){
                if(root!=null){
                        System.out.print(root.data+", ");
                        preorder(root.left);
                        preorder(root.right);
                }
        }
        public void postorder(TreeNode root){
        }
        public void inorder(TreeNode root){
        }
        public int nodecount(TreeNode root){   //대략적인 코드?
                if(root!=null){
                        this.count++;
                        nodecount(root.left);
                        nodecount(root.right);
                }
                return this. Count;
        }
        public int leafcount(TreeNode root){
                if(root!=null){
                        if(root.left==null)&&(root.right==null)){ this.leafcount++;}
                        nodecount(root.left);
                        nodecount(root.right);
                }
                return this. leafcount;

        }

}
```

```
class TreeNode {
        int data;
        TreeNode left;
        TreeNode right;

        public TreeNode(int data1){//constructor
                data = data1;
                left=null;
                right=null;
        }
}
```

0. 주어진 프로그램에 inorder, postorder,preorder method를 추가하라.

1.주어진 프로그램에 노드의 개수를 계산하는 count method를 추가하라(참고로 노드의 개수는 11이다)

2.주어진 프로그램에 외부노드의 개수를 계산하는 leafcount method를 추가하라.(참고로 해당그림의 외부노드의 개수는 5이다)

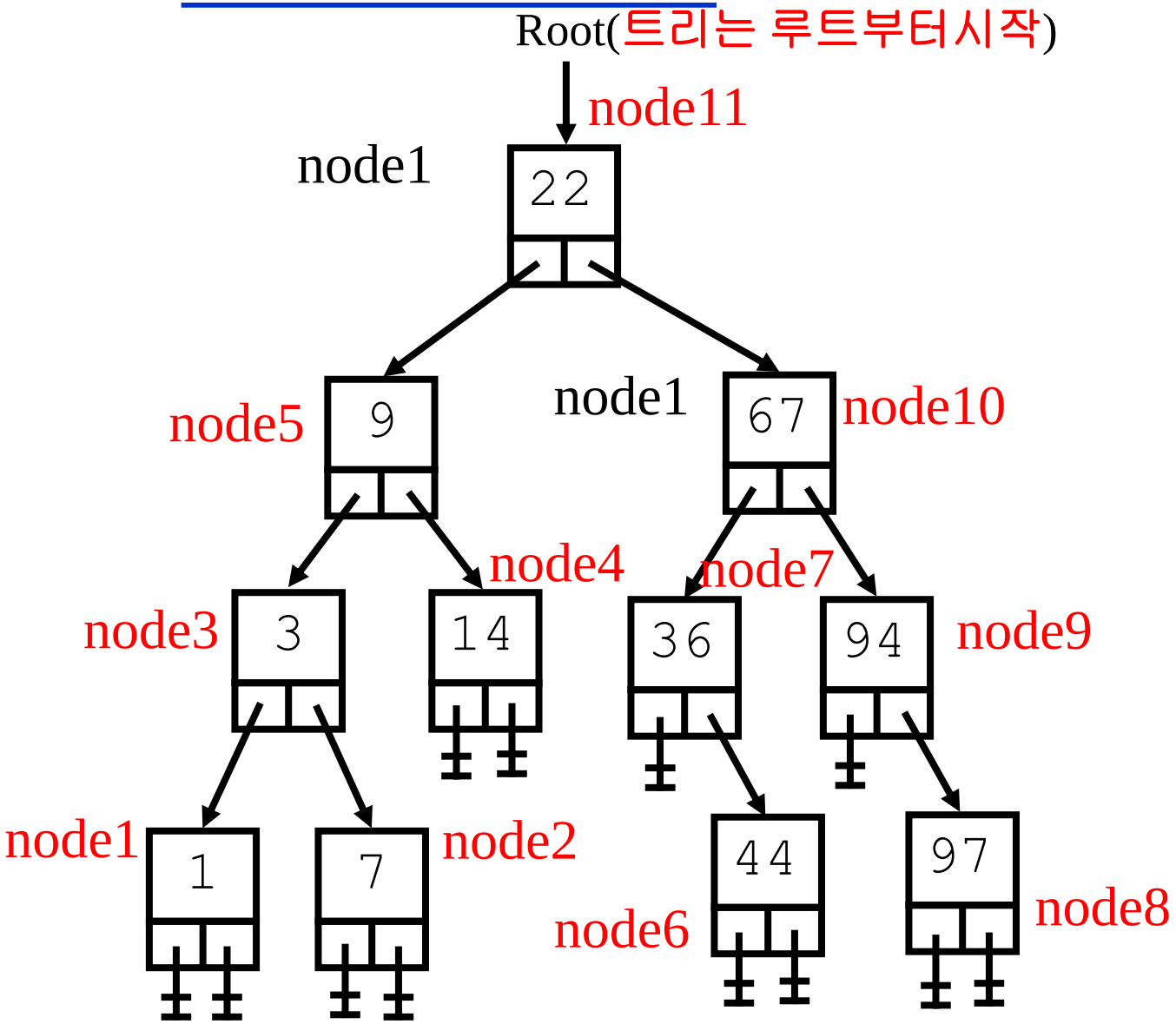
3.주어진 프로그램에 해당 트리의 깊이를 계산하는 depthcount method을 추가하라. 참고로 이 트리의 깊이는3이다. (해결한 자는)

프로그램을 학번+성명.zip으로 하여 업로드하라

Modification of inorder, postorder, preorder

- LDR (inorder)
- DLR(preorder)
- LRD(postorder)

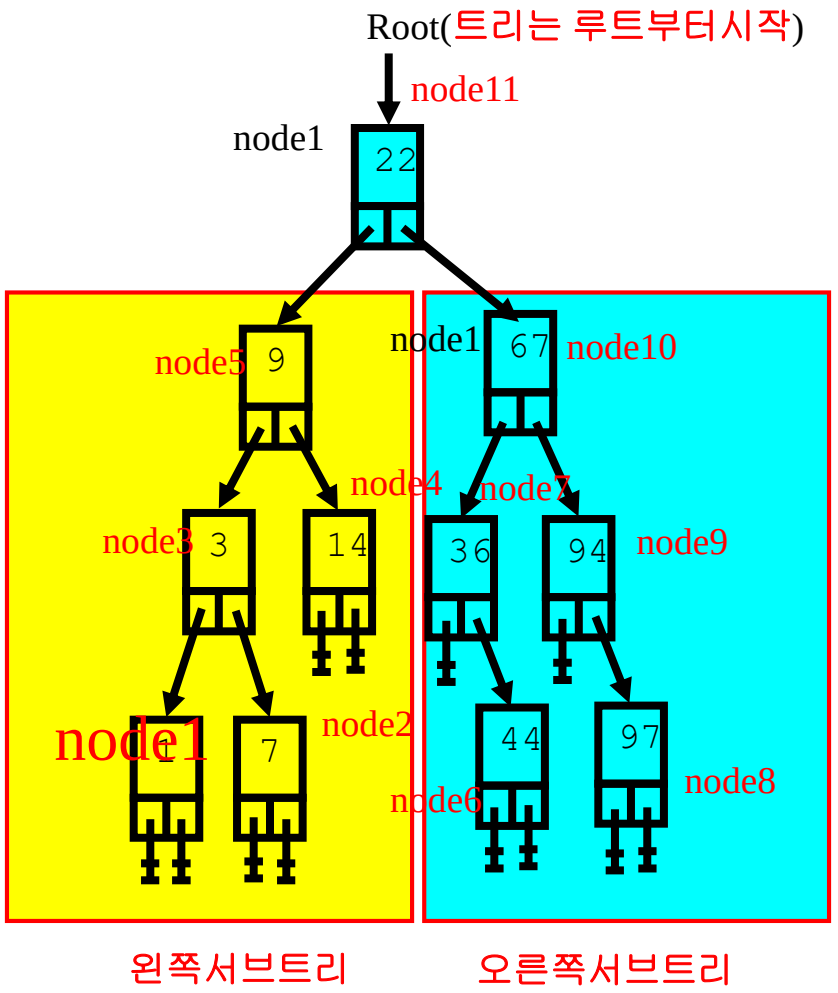
트리의 깊이


Root(트리는 루트부터시작)
node11
node1
22
node5
9
node1
67
node10
node4
node7
node3
3
14
36
94
node9
node1
1
7
node2
44
97
node6
node8

이진트리 A

트리의 깊이

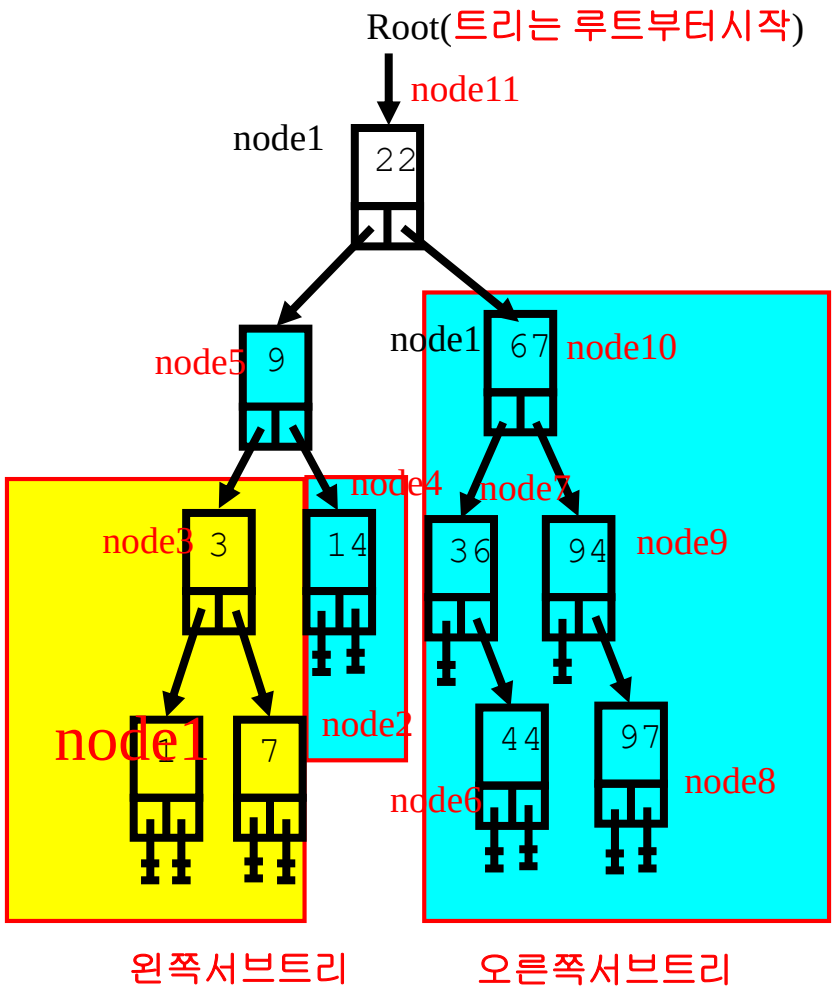
깊이(A)=1+max(깊이( ), 깊이( ))



이진트리 A

트리의 깊이

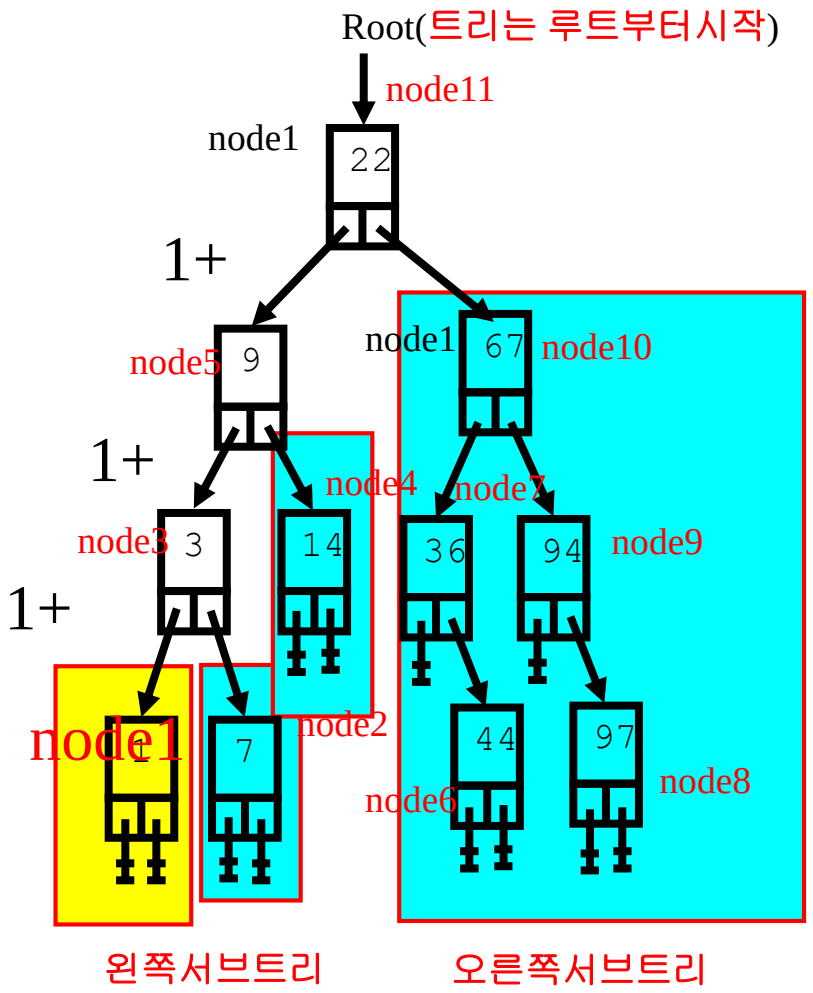
깊이(A)=1+max(깊이( ), 깊이( ))
=1+max(1+max(깊이( ), 깊이( )), 깊이( ))



이진트리 A

트리의 깊이

깊이(A)=1+max(깊이( ), 깊이( ))
=1+max(1+max(깊이( ), 깊이( )), 깊이( ))

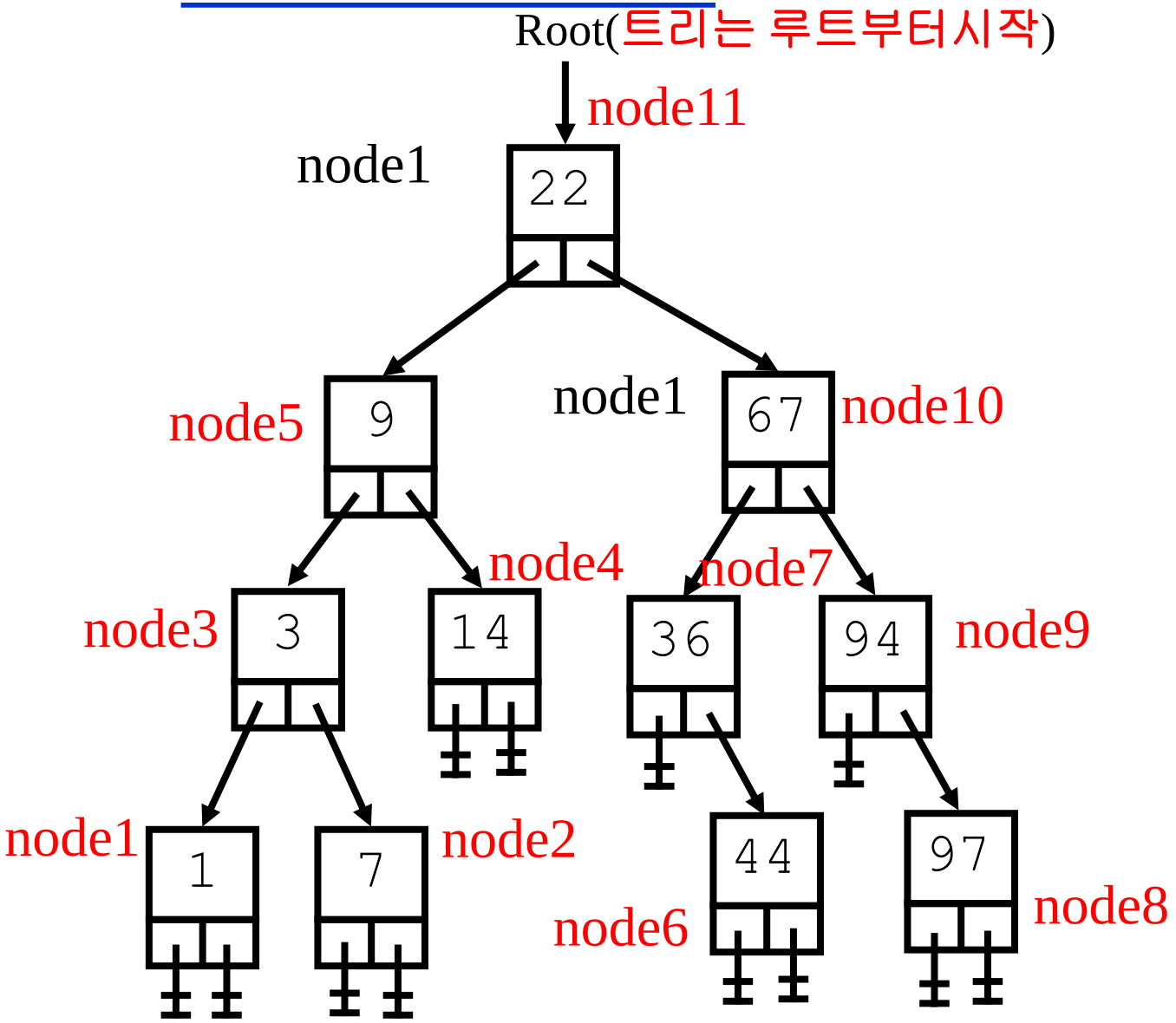


이진트리 A

depthcount

```
public int depthcount(TreeNode root) {
        int count = 0;
        if(root != null){
                count = 1 + Math.max(depthcount(root.left),depthcount(root.right));
        }
        return count;
}
```

노드의 개수


Root(트리는 루트부터시작)
node11
node1
22
node5
9
node1
67
node10
node4
node7
node3
3
14
36
94
node9
node1
1
7
node2
44
97
node6
node8

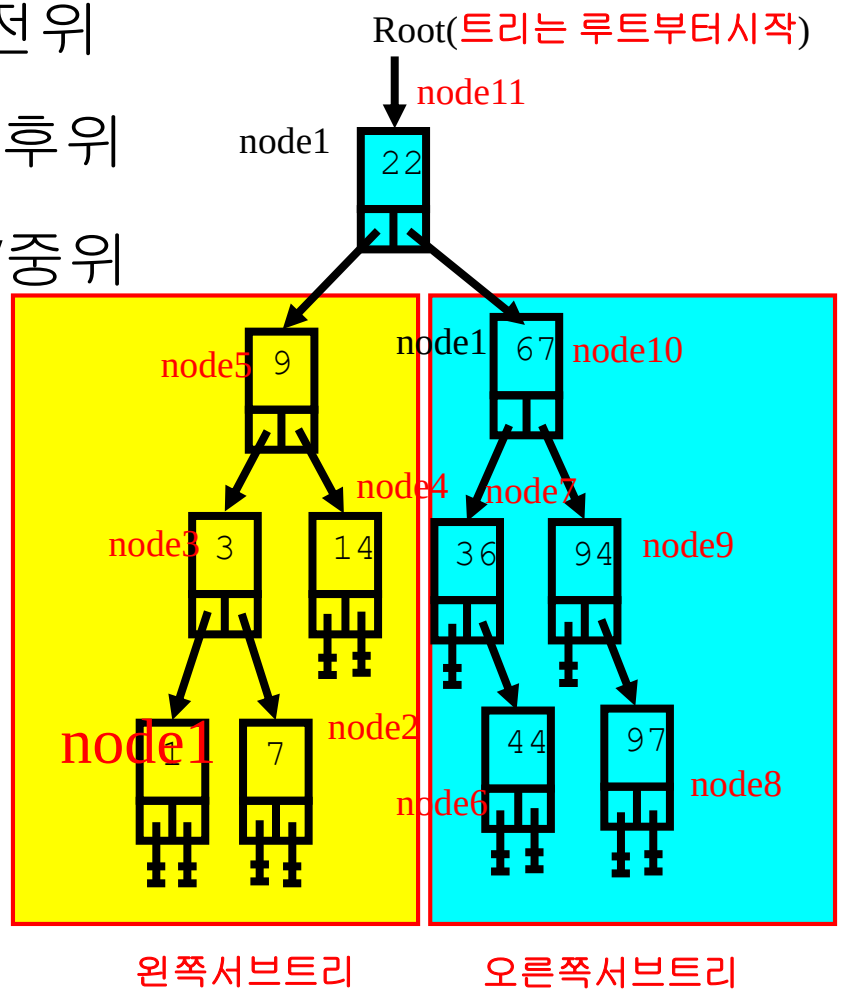
이진트리 A

트리의 깊이

개수(A)=1+ 개수( )+ 개수( ) //전위

개수(A)= (개수( ) + 개수( )+ 1 //후위

개수(A)= (개수( ) + 1+ 개수( ) //중위



이진트리 A

nodecount

```
public int nodecount(TreeNode root) {
        int count = 0;
        if (root == null)
                return 0;
        else
                count += 1 + nodecount(root.left) + nodecount(root.right);
        return count;
}
```

Upload your program(BT.java)(at smartlead.hallym.ac.kr)
*.java 파일만을 업로드하라.
다른 파일들은 필요없다.

Upload your program