1. **DEFINISI MASALAH**
2. Tambahkan method untuk menghitung banyaknya node pada Binary Tree
3. Tambahkan method untuk menghitung banyaknya daun
4. Tambahkan method untuk menghitung tinggi dari pohon
5. Tambahkan method panjang yang menerima suatu pohon biner dan menentukan berapa level yang dimiliki pohon tersebut.
6. Buatlah program Complete Binary Tree dengan menggunakan array.
7. **SOURCE CODE**

|  |  |
| --- | --- |
|  | Tree.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227 | package Praktikum9;  import java.util.Random;  import java.util.Scanner;  class Node {  int data;  Node nodeKiri;  Node nodeKanan;  public Node(int dt) {  data = dt;  nodeKiri = nodeKanan = null;  }  public void sisipDt(int dtSisip) {  if (dtSisip < data) {  if (nodeKiri == null) {  nodeKiri = new Node(dtSisip);  } else {  nodeKiri.sisipDt(dtSisip);  }  } else if (dtSisip > data) {  if (nodeKanan == null) {  nodeKanan = new Node(dtSisip);  } else {  nodeKanan.sisipDt(dtSisip);  }  }  }  }  public class Tree {  private Node root;  int jumlah, leave;  private int panjang;  public Tree() {  root = null;  }  public void sisipDtNode(int dtSisip) {  if (root == null) {  root = new Node(dtSisip);  } else {  root.sisipDt(dtSisip);  }  jumlah++;  }  public int getJumlah() { // menghitung jumlah node  return jumlah;  }  public void preorderTraversal() {  preorder(root);  }  private void preorder(Node node) {  if (node == null) {  return;  }  System.out.printf("%d ", node.data);  preorder(node.nodeKiri);  preorder(node.nodeKanan);  }  public void getDaun(Node node) { // hitung banyak daun  if (node.nodeKanan == null) {  if (node.nodeKiri == null) {  leave++;  } else {  getDaun(node.nodeKiri);  }  } else if (node.nodeKiri == null) {  if (node.nodeKanan == null) {  leave++;  } else {  getDaun(node.nodeKanan);  }  } else if (node.nodeKanan != null && node.nodeKiri != null) {  getDaun(node.nodeKiri);  getDaun(node.nodeKanan);  }  }  private int ceklevel(Node node) { // cek level pohon  if (node != null) {  int a = ceklevel(node.nodeKiri);  int b = ceklevel(node.nodeKanan);  if (a > b) {  return a = a + 1;  } else {  return b = b + 1;  }  } else {  return -1;  }  }  public int Level() {  return ceklevel(root);  }  public int Tinggi() { // tinggi pohon  return Level() + 1;  }  public void cekDaun() {  getDaun(root);  }  public void inorderTraversal() {  inorder(root);  }  private void inorder(Node node) {  if (node == null) {  return;  }  inorder(node.nodeKiri);  System.out.printf("%d ", node.data);  inorder(node.nodeKanan);  }  public void postorderTraversal() {  postorder(root);  }  private void postorder(Node node) {  if (node == null) {  return;  }  postorder(node.nodeKiri);  postorder(node.nodeKanan);  System.out.printf("%d ", node.data);  }  private int setPanjang(int p) {  this.panjang = p;  int a = this.Tinggi() - this.Level();  return a;  }  private int getPanjang() {  return panjang;  }  public boolean delete(int data) {  Node current = root;  Node parent = root;  boolean isLeftChild = true;  while (current.data != data) {  parent = current;  if (data < current.data) {  isLeftChild = true;  current = current.nodeKiri;  } else {  isLeftChild = false;  current = current.nodeKanan;  }  if (current == null) {  return false;  }  }  if (current.nodeKiri == null && current.nodeKanan == null) {  if (current == root) {  root = null;  } else if (isLeftChild) {  parent.nodeKiri = null;  } else {  parent.nodeKanan = null;  }  } else if (current.nodeKanan == null) {  if (current == root) {  root = current.nodeKiri;  } else if (isLeftChild) {  parent.nodeKiri = current.nodeKiri;  } else {  parent.nodeKanan = current.nodeKiri;  }  } else if (current.nodeKiri == null) {  if (current == root) {  root = current.nodeKanan;  } else if (isLeftChild) {  parent.nodeKiri = current.nodeKanan;  } else {  parent.nodeKanan = current.nodeKanan;  }  } else {  Node successor = getSuccessor(current);  if (current == root) {  root = successor;  } else if (isLeftChild) {  parent.nodeKiri = successor;  } else {  parent.nodeKanan = successor;  }  successor.nodeKiri = current.nodeKiri;  }  return true;  }  private Node getSuccessor(Node delNode) {  Node successorParent = delNode;  Node successor = delNode;  Node current = delNode.nodeKanan;  while (current != null) {  successorParent = successor;  successor = current;  current = current.nodeKiri;  }  if (successor != delNode.nodeKanan) {  successorParent.nodeKiri = successor.nodeKanan;  successor.nodeKanan = delNode.nodeKanan;  }  return successor;  }  public static void main(String args[]) {  Tree Tree = new Tree();  Scanner input = new Scanner(System.in);  int array[] = new int[24];  int nilai;  Random randomNumber = new Random();  System.out.println("sisip nilai data berikut : ");  // sisipDt 10 bilangan acak dari 0-99 ke dalam tree  for (int i = 0; i < 10; i++) {  nilai = randomNumber.nextInt(100);  System.out.print(nilai + " ");  Tree.sisipDtNode(nilai);  }  System.out.println("");  System.out.print("Masukan Nilai Data Yang Akan Dihapus\t: ");  nilai = input.nextInt();  Tree.delete(nilai);  System.out.println("\nPreorder traversal");  Tree.preorderTraversal();  System.out.println("\n\nInorder traversal");  Tree.inorderTraversal();  System.out.println("\n\nPostorder traversal");  Tree.postorderTraversal();  System.out.println("\n\nJumlah node pada binary tree adalah :");  System.out.println(Tree.getJumlah());  Tree.cekDaun();  System.out.print("\nJumlah daun : " + Tree.leave);  System.out.println("\nJumlah level : " + Tree.Level());  System.out.println("Jumlah tinggi : " + Tree.Tinggi());  }  } |

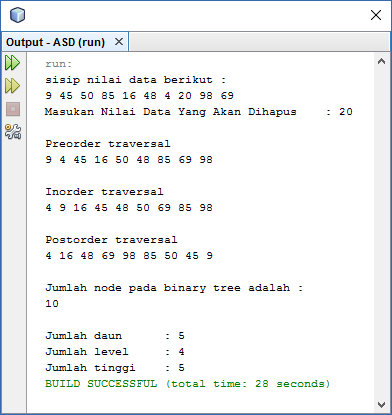
|  |  |
| --- | --- |
|  | Array\_Binary\_Tree.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84 | package Praktikum9;  public class Array\_Binary\_Tree {  int[] aTree = new int[100];  public Array\_Binary\_Tree() {  }  public void insert(int key) {  if (aTree[0] == 0) {  aTree[0] = key;  return;  }  int index = 0;  boolean add = false;  while (!add) {  if (key < aTree[index]) {  if (aTree[index \* 2 + 1] == 0) {  aTree[index \* 2 + 1] = key;  add = true;  } else {  index = index \* 2 + 1;  }  } else {  if (aTree[index \* 2 + 2] == 0) {  aTree[index \* 2 + 2] = key;  add = true;  } else {  index = index \* 2 + 2;  }  }  }  }  public void preOrder(int idx, String i) {  if (aTree[idx] != 0) {  System.out.print(i + aTree[idx] + ", ");  if (idx >= aTree.length) {  return;  } else {  preOrder(((2 \* idx) + 1), i);  preOrder(((2 \* idx) + 2), i);  }  }  }  public void postOrder(int idx, String i) {  if (aTree[idx] != 0) {  if (idx >= aTree.length) {  return;  } else {  postOrder(((2 \* idx) + 1), i);  postOrder(((2 \* idx) + 2), i);  }  System.out.print(i + aTree[idx] + ", ");  }  }  public void inOrder(int idx, String i) {  if (aTree[idx] != 0) {  if (idx >= aTree.length) {  return;  } else {  inOrder(((2 \* idx) + 1), i);  System.out.print(i + aTree[idx] + ", ");  inOrder(((2 \* idx) + 2), i);  }  }  }  public static void main(String[] args) {  int Tree[] = {23, 13, 10, 17, 6, 9, 21, 25, 22, 24, 29};  Array\_Binary\_Tree a = new Array\_Binary\_Tree();  System.out.println("sisip nilai data berikut : ");  for (int i = 0; i < Tree.length; i++) {  a.insert(Tree[i]);  System.out.print(Tree[i]+", ");  }  System.out.println("\n\n======= Binary Tree dengan Array =======");  System.out.println("Preorder:");  a.preOrder(0, "");  System.out.println("");  System.out.println("Postorder:");  a.postOrder(0, "");  System.out.println("");  System.out.println("Inorder:");  a.inOrder(0, "");  System.out.println("");  System.out.println("======================================");  }  } |

1. **PEMBAHASAN**

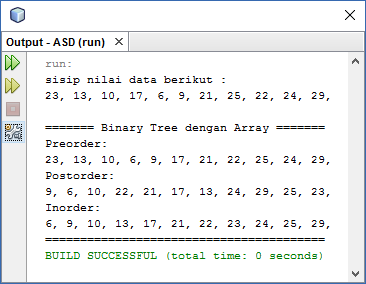
|  |  |
| --- | --- |
|  | Tree.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  50  51  52  53  54  55  56  57  58  59  60  61  62  63  65  66  67  68  69  71  72  73  75  76  77  78  79  80  81  82  83  85  86  88  89  90  92  93  95  96  98  99  101  102  103  105  106  107  109  110  112  113  114  116  117  118  120  121  122  123  125  126  128  129  130  131  132  133  134  135  136  137  138  139  141  142  145  146  147  148  149  150  151  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  169  170  171  172  173  174  175  176  178  180  182  183  184  185  186  187  188  189  191  192  193  195  196  197  198  199  200  201  202  203  204  205  206  207  208  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227 | Package di mana file java disimpan.  Perintah bagi java untuk mengambil data dari library random gunanya untuk menentukan perintah selanjutnya bagi program agar user dapat memasukkan sendiri nilai variabel yang dibutuhkan  Perintah import dari package Scanner  Deklarasi kelas dengan nama Node  Deklarasi variabel data dengan tipe data integer  Deklarasi variabel nodeKiri dengan tipe data Node  Deklarasi variabel nodeKanan dengan tipe data Node  Konstruktor node dengan parameter variabel dt dengan tipe data integer  Pemberian nilai pada variabel data dengan nilai dt  Pemberian nilai pada variabel nodeKiri yang sama dengan nilai variabel nodeKanan dengan nilai null  Akhir deklarasi konstruktor.  Method void dengan nama sisipDt dengan parameter variabel dtSisip yang bertipe data integer  Pemilihan kondisi dengan if else jika dtSisip lebih kecil dari data  Jika syarat sebelumnya terpenuhi maka masuk kembali pada pemilihan kondisi dengan syarat Jika nilai nodeKiri sama dengan null  Membuat objek baru dengan Nilai nodeKiri sama dengan nilai dtSisip  Pemilihan kondisi jika syarat sebelumnya tidak terpenuhi  Variabel nodeKiri memanggil variabel sisipDt dengan nilai dtSisip  Akhir kondisi if.  Pemilihan kondisi dengan if else dengan syarat jika dtSisip lebih besar dari data  Bila syarat sebelumnya sudah terpenuhi maka akan masuk pada proses pemilihan kondisi kembali dengan syarat nilai nodeKanan sama dengan null  Memanggil objek baru dengan Nilai nodeKanan sama dengan nilai dtSisip  Pemilihan kondisi jika syarat sebelumnya tidak terpenuhi  Variabel nodeKanan memanggil variabel sisipDt dengan nilai dtSisip  Akhir kondisi else kedua.  Akhir kondisi else pertama.  Akhir deklarasi method.  Akhir deklarsi class Node.  Deklarasi kelas dengan nama Tree  Deklarasi variabel root dengan tipe data Node yang memiliki modifier tipe private  Deklarasi variabel jumlah dan leave dengan tipe data integer  Deklarasi variabel panjang dengan tipe data integer yang memiliki modifier tipe private  Deklarasi method Konstruktor dengan nama Tree  Pemberian nilai root dengan nilai null  Akhir deklarasi konstruktor.  Method void dengan nama sisipDtNode dengan parameter dtSisip yang memiliki tipe data integer  Pemilihan kondisi dengan if jika root sama dengan null  Variabel root sama dengan nilai dari variable dtSisip  Pemilihan kondisi jika syarat sebelumnya tidak terpenuhi  Variabel root memanggil variabel sisipDt dengan nilai dtSisip  Akhir kondisi else.  Maka variabel jumlah bertambah 1  Akhir deklarsi method  Deklarasi Method dengan nama getJumlah  Mengembalikan nilai jumlah  Akhir deklarasi method  Deklarasi Method void dengan nama preorderTraversal  Variabel preorder yang bernilai root  Akhir deklarsi method  Deklarasi Method void dengan nama preorder berparameter node dengan tipe data Node  Pemilihan kondisi dengan if dengan syarat jika node bernilai null  Maka mengembalikan nilai node  Akhir kondisi if  Perintah akan menampilkan nilai data dengan format tertentu  Preorder dengan nilai nodeKiri  Preorder dengan nilai nodeKanan  Akhir deklarsi method.  Deklarasi Method void dengan nama getLeave berparameter node dengan tipe data node  Pemilihan kondisi dengan if yang memiliki syarat jika node yang memanggil nodeKanan bernilai null  Pemilihan kondisi dengan if yang memiliki syarat jika node yang memanggil nodeKiri bernilai null  Maka leave akan bertambah 1  Pemilihan kondisi dengan else yang jika syarat nya tidak memenuhi syarat diata  Method get leave akan bernilai node yang memanggil nilai nodeKiri  Pemilihan kondisi dengan if yang memiliki syarat jika node yang memanggil nodeKiri bernilai null  Pemilihan kondisi dengan if yang memiliki syarat jika node yang memanggil nodeKanan bernilai null  Maka leave akan bertambah 1  Pemilihan kondisi dengan else yang jika syarat nya tidak memenuhi syarat diatas  Method get leave akan bernilai node yang memanggil nilai nodeKiri  Pemilihan kondisi dengan else if dengan syarat jika nodeKanan dan nodeKiri tidak sama dengan null  Method getLeave bernilai node yang memanggil nodeKiri  Method getLeave bernilai node yang memanggil nodeKanan  Akhir deklarsi method  Deklarasi Method dengan nama cekleave dengan type integer berparameter node yang memilki tipe data Node  Pemilihan kondisi dengan if syarat node tidak sama dengan null  Deklarasi variabel a dengan tipe data integer yang memiliki nilai nodeKiri  Deklarasi variabel b dengan tipe data integer yang memiliki nilai nodeKanan  Pemilihan kondisi dengan if yang syaratnya a lebih besar dari b  Mengembalikan nilai variabel a yang bernilai a bertambah 1  Pemilihan kondisi dengan else  Mengembalikan nilai varibel b yang bertambah 1  Pemilihan kondisi dengan else  Mengembalikan nilai -1  Akhir deklarsi method  Deklarasi Method dengan nama level yang bertipe integer  Mengembalikan nilai cekLevel yang bernilai root  Deklarasi Method dengan nama Tinggi yang bertipe integer  Mengembalikan nilai method Level yang bertambah 1  Deklarasi Method void dengan nama cekLeave  Memanggil method getLevel yang bernilai root  Deklarasi Method void dengan nama inorderTraversal  Inorder bernilai root  Deklarasi Method void dengan nama inorder berparameter node yang memiliki tipe data Node  Pemilihan kondisi menggunakan if dengan syarat jika node bernilai null  Mengemblikan nilai pada variable node  Method inorder bernilai nodeKiri  Menampilkan data dengan perintah printf  Method inorder bernilai node yang memanggil nilai nodeKanan  Deklarasi Method void dengan nama postorderTraversal  Postorder benilai root  Deklarasi Method void dengan nama postorder berparamter node yang memiliki tipe data Node  Pemilihan kondisi menggunakan if dengan syarat jika node sama dengan null  Mengembalikan nilai pada variable node  Method postorder benilai node yang memanggil nilai nodeKiri  Method postorder benilai node yang memanggil nilai nodeKanan  Menampilkan data dengan perintah dari printf  Deklarasi Method dengan nama setPanjang berparameter p dengan tipe data integer  Pemberian nilai panjang dengan nilai p menggunakan keyword this  Deklarasi nilai a dengan nilai method tinggi – method level  Mengembalikan nilai a  Deklarasi Method dengan nama getPanjang dengan tipe integer  Mengembalikan nilai panjang  Deklarasi Method dengan nama delete dengan tipe Boolean berparameter data memiliki tipe data integer  Deklarasi current dengan tipe Node yang bernilai root  Deklarasi parent dengan tipe Node yang bernilai root  Deklarasi isLeftChild dengan tipe data Boolean yang bernilai true  Perulangan dengan while jika nilai current yang memanggil data tidak sama dengan nilai variable data  Nilai parent sama dengan current  Pemilihan jika data lebih kecil dari nilai current  Maka nilai isLeftChilt bernilai true  Nilai current bernilai current yang memanggil nilai nodeKiri  Pemilihan kondisi dengan else  isLeftChild bernilai false  Nilai current bernilai current yang memanggil nilai nodeKanan  Jika current bernilai null  Maka mengembalikan nilai false  Pemilihan kondisi menggunakan if dengan syarat jika current yang memanggil nodeKiri dan current yang memanggil nodeKanan bernilai null  Pemilihan kondisi dengan if yang bersyarat jika current bernilai root  Maka root bernilai null  Pemilihan kondisi dengan else if jika isLeftChild  Parent yang memanggil nodeKiri bernilai null  Pemilihan kondisi dengan else  Parent yang memanggil nodeKanan bernilai null  Pemilihan kondisi dengan else if syarat jika current yang memanggil nodeKanan bernilai null  Pemilihan kondisi dengan if syarat jika current bernilai root  Maka root bernilai current yang memanggil nodeKiri  Pemilihan kondisi dengan else if jika isLeftChild  Parent yang memanggil nodeKiri bernilai current yang memanggil nodeKiri  Pemilihan kondisi dengan else  Parent yang memanggil nodeKanan bernilai current yang memanggil nodeKiri  Pemilihan kondisi dengan else if syarat jika current yang memanggil nodeKiri bernilai null  Pemilihan kondisi dengan if syarat jika current bernilai root  Maka root bernilai current yang memanggil nodeKanan  Pemilihan kondisi dengan else if syarat jika isLeftChild  Parent yang memanggil nodeKiri bernilai current yang memanggil nodeKanan  Pemilihan kondisi dengan else  Parent yang memanggil nodeKanan bernilai current yang memanggil nodeKanan  Pemilihan kondisi menggunakan else  Deklarasi successor bertipe Node bernilai method getSuccessor dengan nilai current  Pemilihan kondisi menggunakan if syarat jika current bernilai root  Maka root bernilai successror  Pemilihan kondisi dengan else if jika isLeftChild  Parent yang memanggil nodeKiri bernilai successor  Pemilihan kondisi menggunakan else  Parent yang memanggil nodekanan bernilai successor  Successor yang memanggil node kiri sama dengan current yang memanggil nodeKiri  Mengembalikan nilai true  Method dengan nama getSuccessor dengan tipe Node berparameter delNode yang memiliki tipe data Node  Deklarasi objek baru dengan nama successorParent dengan tipe data Node bernilai delNode  Deklarasi objek baru dengan nama successor dengan nilai delNode  Deklarasi objek baru dengan nama current dengan nilai delNode yang memanggil nodeKanan  Perulangan dengan while syarat jika nilai current tidak sama dengan null  Inisialisasi nilai successorParent dengan nilai successor  Inisialisasi nilai successor dengan nilai current  Inisialisasi nilai dengan nilai current yang memanggil nodeKiri  Pemilihan kondisi dengan if jika successor tidak sama dengan delNode yang memanggil nodeKanan  Nilai successorParent yang memanggil nodeKiri bernilai successor yang memanggil nilai nodeKanan  Successor yang memanggil nodeKanan bernilai delNode yang memanggil nodeKanan  Mengembalikan nilai successor  Akhir deklari method  Deklarasi Main method, untuk menjalankan program.  Objek baru dengan nama Tree yang dipanggil dari kelas Tree  Deklarasi variable Scanner bernama input dari package Scanner  Objek array dengan nilai indeks 24  Deklarasi variable nilai dengan tipe data integer  Objek baru dengan nama randomNumber  Menampilkan kalimat “sisip data berikut”  Komentar  Perulangan dengan for jika nilai i sama dengan 1 dan nilai i lebih kecil atau sama dengan 10 maka nilai i bertambah 1  Nilai yang bernilai objek randomNumber memanggil nextInt dengan nilai 100  Menampilkan nilai dari variabel nilai  Objek Tree memanggil sisipDtNode bernilai variabel nilai  Mencetak baris kosong  Menampilkan kalimat “Masukkan nilai Data yang akan dihapus”  Perintah menunggu user memasukkan nilai pada variable nilai  Objek Tree memanggil delete yang bernilai variabel nilai  Akan menampilkan kalimat “Preorder traversal”  Objek Tree memanggil method preorderTraversal  Akan menampilkan kalimat “Inorder traversal”  Objek Tree memanggil method InorderTraversal  Akan menampilkan kalimat “Postorder traversal”  Objek Tree memanggil method postorderTraversal  Menampilkan kalimat “jumlah node pada binary tree adalah : “  Menampilkan nilai objek Tree yang memanggil method getJumlah  Objek tree memanggil method cekLeave  Menampilkan “jumlah leave :”  Menampilkan “jumlah level :”  Menampilkan “jumlah tinggi :”  Akhir deklarsi method  Akhir deklarasi class Tree. |

|  |  |
| --- | --- |
|  | Array\_Binary\_Tree.java |
| 1  2  3  4  6  7  8  9  11  12  13  14  15  16  17  18  19  21  22  23  24  25  30  31  32  33  34  35  36  38  41  42  43  44  45  46  47  48  50  52  53  54  55  56  57  58  59  60  63  64  65  66  67  68  69  70  72  73  74  75  76  77  78  79  80  81  82  83  84 | Package dimana file java di simpan.  Deklarasi class dengan nama Array\_Binary\_Tree  Inisialisasi variable array bertipe data integer dengan nama aTree berparameter 100  Konstraktor dengan nama Array\_Binary\_Tree yang nilainya adalah default.  Method dengan nama insert bertipe void dengan nama insert berparameter int key  Pemilihan kondisi dengan if yang memiliki syarat jika array dari aTree nilainya 0 akan sama dengan 0.  Jika array aTree bernilai 0 maka sama dengan key.  Return berfungsi mengembalikan nilai dari method insert.  Deklarasi variable index bertipe data integer yang bernilai 0  Deklarasi variable add bertipe Boolean yang memiliki nilai false.  Perulangan dengan while yang bersyarat jika nilainya tidak sama dengan add.  Pemiliha kondisi dengan syarat jika key kurang dari nilai aTree yang nilainya adalah index.  Pemilihan kondisi if didalam if dengan syarat jika aTree bernilai index \* 2+1 sama dengan 0.  aTree yang bernilai index \*2+1 samadengan key  Maka add samadengan true.  Jika syarat if tidak terpenuhi maka akan ke syarat else yang syaratnya  index sama dengan index \*2+1.  Pemilihan kondisi else jika if yang pertama syaratnya belum terpenuhi maka jika aTree nilainya index\*2+2 samadengan 0.  aTree bernilai index \*2+2 sama dengan key  maka add samadengan true.  Pemilihan kondisi dengan if else yang syaratnya  index sama dengan index \*2+2.  Akhir deklarasi method  Method dengan nama preorder bertipe void dengan parameter int idx, String i.  Pemilihan kondisi if dengan syarat jika aTree bernilai idx tidak sama dengan 0.  Maka akan mencetak nilai I + aTree dengan nilai idx dan tanda koma untuk pemisah.  Pemilihan kondisi dengan if di dalam if yang bersyarat jika nilai idx lebih dari sama dengan panjang aTree.  Return berfungsi mengembalikan nilai.  Pemilihan kondisi dengan if else jika syarat if belum terpenuhi maka akan ke else yang bersyarat preorder dengan nilai 2 \* nilai dari idx + 1, nilai dari i.  PreOrder dengan nilai 2 \* nilai dari idx + 2, nilai dari i.  Akhir deklarsi method  Method dengan nama postOrder bertipe void dengan parameter int idx dan String i.  Pemilihan kondisi if yang bersyarat jika aTree yang nilainya adalah idx tidak sama dengan nol  Pemilihan kondisi if jika nilai dari idx lebih dari sama dengan panjang aTree.  Return berfungsi mengembalikan nilai.  Jikaselain kondisi di atas.  PostOrder dengan nilai 2 \* nilai dari idx + 1, nilai dari i  PostOrder dengan nilai 2 \* nilai dari idx + 2, nilai dari i  Perintah untuk mencetak nilai I + aTree yang bernilai dari idx dan tanda koma untuk pemisah.  Akhir deklarasi method  Method dengan nama inOrder yang berparameter int idx String i.  Pemilihan kondisi dengan if yang bersyarat jika aTree bernilai idx negasi 0.  Jika nilai idx lebih dari sama dengan nilai dari panjang aTree.  Mengembalikan nilai  Jika selain kondisi diatas, maka  InOrder dengan nilai 2 \* nilai dari idx + 1, nilai dari i  Mencetak nilai I + aTree yang bernilai dari idx dan tanda koma untuk pemisah.  InOrder dengan nilai 2 \* nilai dari idx + 2, nilai dari i.  Akhir deklarasi method.  Deklarasi method main, untuk menjalankan program.  Deklarasi variable Tree yang nilainya sudah di tentukan.  Inisialisasi objek dengan nama a.  Mencetak “sisip nilai data berikut...”  Perulangan dengan for yang bersyarat int I sama dengan 0 dan I lebih besar dari panjang Tree maka nilai I akan bertambah.  Memanggil method insert dengan objek a.  Mencetak Tree index ke i  Perintah untuk mencetak “====== Binary Tree dengan Array====”.  Perintah ntuk mencetak kalimat “Preorder”.  Objek a memanggil method preorder.  Perintah untuk mencetak baris kosong  Perintah ntuk mencetak kalimat “Postorder”.  Objek a memanggil method postorder.  Perintah untuk mencetak baris kosong  Perintah ntuk mencetak kalimat “Inorder”.  Objek a memanggil method inorder.  Perintah untuk mencetak baris kosong.  Perintah untuk mencetak tanda ======.  Akhir deklarsi method main  Akhir deklarsi class. |

1. **SCREENSHOT PROGRAM**



Gambar. Studi kasus No. 1, 2, 3 dan 4.

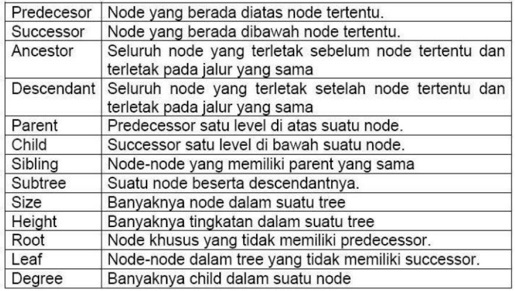


Gambar. Studi kasus No. 5

1. **KESIMPULAN**
2. Pengertian dan kegunaan struktur data binary tree?

Tree adalah kumpulan node yang saling terhubung satu sama lain dalam suatu  kesatuan yang membentuk layakya struktur sebuah pohon. Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (one-to-many) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut  hanya tampak sebagai kumpulan node-node  dari atas ke bawah.

Istilah – istilah dalam tree :



pohon biner *(binary tree)* adalah sebuah [pohon](https://id.wikipedia.org/wiki/Pohon_%28struktur_data%29) [struktur data](https://id.wikipedia.org/wiki/Struktur_data) dimana setiap [simpul](https://id.wikipedia.org/wiki/Pohon_%28struktur_data%29#Simpul_.28node.29) memiliki paling banyak dua [anak](https://id.wikipedia.org/wiki/Pohon_%28struktur_data%29). Secara khusus anaknya dinamakan anak kiri dan anak kanan.Tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal dua sub pohon dan kedua subpohon harus terpisah.

Kelebihan struktur Binary Tree :

1. Mudah dalam penyusunan algoritma sorting
2. Searching data relatif cepat
3. Fleksibel dalam penambahan dan penghapusan data

Sebuah pohon biner memiliki operasi traversal yaitu suatu kunjungan pada suatu simpul tepat satu kali. Dengan melakukan kunjungan lengkap kita akan memperoleh urutan informasi secara linier yang tersinpan di dalam pohon biner.

Terdapat tiga jenis kunjungan pada pohon biner, yaitu :

1. PREORDER

Kunjungan jenis ini mempunyai urutan kunjungan sebagai berikut :

* Cetak isi simpul yang dikunjungi.
* Kunjungi cabang kiri.
* Kunjungi cabang kanan.

1. INORDER

Kunjungan jenis ini mempunyai urutan kunjungan sebagai berikut :

* Kunjungi cabang kiri.
* Cetak isi simpul yang dikunjungi.
* Kunjungi cabang kanan.

1. POSTORDER

Kunjungan jenis ini mempunyai urutan kunjungan sebagai berikut :

* Kunjungi cabang kiri.
* Kunjungi cabang kanan.
* Cetak isi simpul yang dikunjungi

1. Contoh penggunaan / implementasi Binary tree?

Kegunaan dari tree :

* Memanipulasi data secara hierarki
* Membuat informasi mudah untuk dicari
* Memanipulasi data sorted lists

Binary Tree dapat direpresentasikan dengan menggunakan Array maupun linked list.

1. Binary Tree dengan linked list



Detail mengenai Binary tree dengan linked list :

* Terdapat setidaknya 8 operasi yakni, insert, findMin, findMax, find, remove, preOrder, inOrder, postOrder.
* Proses insert dilakukan dengan mengikuti aturan sebagai berikut:

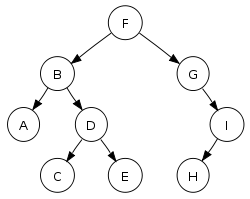
1. Setiap node baru akan menempati posisi sebagai daun (leaf).
2. Untuk insert ke dalam BST dimulai dari root, jika data lebih kecil dari root, node baru harus di masukkan ke dalam left sub tree. Jika data pada node baru lebih besar dari root, node baru harus di masukkan ke dalam right sub tree.

* Operasi findMin akan mencari nilai terkecil dari node yang ada di dalam BST dengan cara menelusuri left subtree sampai ke daun. (gunakan rekursif).
* Operasi findMax akan mencari nilai terbesar dari node yang ada di dalam BST dengan cara menelusuri right subtree sampai ke daun. (gunakan rekursif).
* Operasi find akan mencari nilai yang dicari dengan membandingkan dengan data pada root. Jika lebih kecil akan mencari di left sub tree, dan jika lebih besar cari di right sub tree. (gunakan rekursif).
* Operasi remove dilakukan dengan mengikuti aturan sebagai berikut:

1. Jika node yang dihapus berposisi sebagai daun, dengan sederhana bisa dihapus.
2. Jika node memiliki satu anak, maka anak tersebut akan menggantikan posisi node yang dihapus.
3. Jika node memiliki 2 anak [pilih salah satu]:

* Ganti node yang dihapus dengan node terkecil pada right sub tree.
* Ganti node yang dihapus dengan node terbesar pada left sub tree.

1. Gambarkan contoh struktur data binary tree?



Di Dalam Pohon Binary ini jika disusun secara :

1. Penyusunan Secara PreOrder ( Belum Disusun ) : F, B, A, D, C, E, G, I, H (root, left, right)
2. Penyusunan Secara InOrder : A, B, C, D, E, F, G, H, I (left, root, right)
3. Penyusunan Secara PostOrder: A, C, E, D, B, H, I, G, F (left, right, root)