**LAPORAN**

**PRAKTIKUM STRUKTUR DATA**

**Pertemuan ke – 11**

****

Disusun Oleh :

Rendra Eka Herlambang

175410028

**LABORATORIUM TERPADU**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER AKAKOM YOGYAKARTA**

**2018**

**BinerySearchTree**

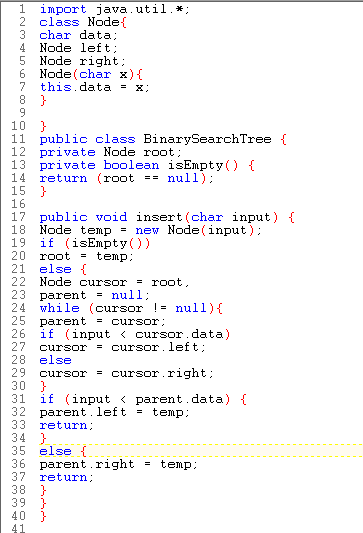
1. **TUJUAN**

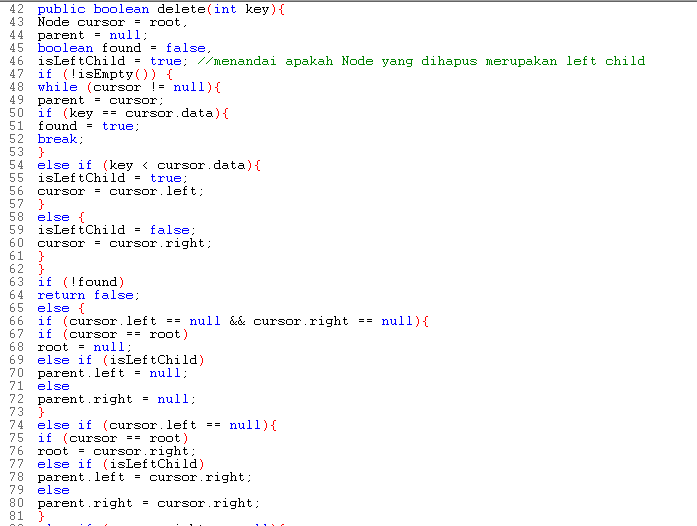
-Mahasiswa dapat membuat program double linked list pohon

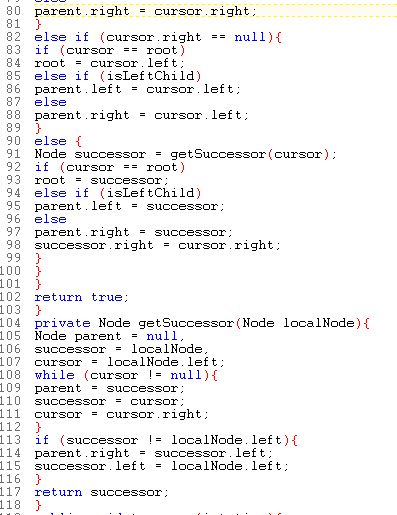
- Mahasiswa mampu mengimplementasikan pada Binery search dengan beda tipe data.

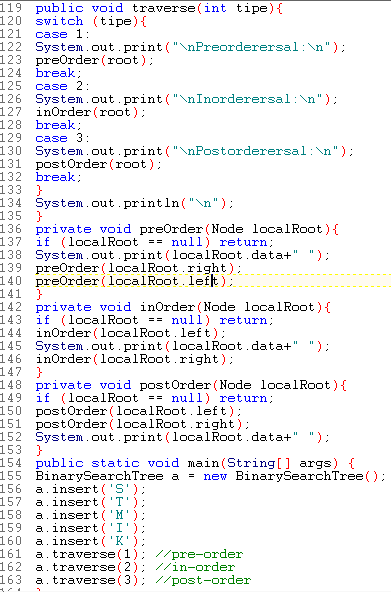
- Mahasiswa mampu mengimplemetasikan BinerySearch Tree dengan baik dan benar.

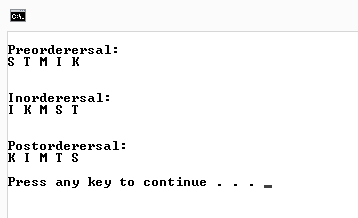
1. **PEMBAHASAN PRAKTIK**











Pembahasan

import java.util.\*;

class Node{

char data;

Node left;

Node right;

Node(char x){

this.data = x;

}

//yaitu untuk membuat class pada Node maka dengan tipe data char pada bagian Node left right maka data di dekrasikan menjadi x

}

public class BinarySearchTree {

private Node root;

private boolean isEmpty() {

return (root == null);

}

//Yaitu pada method ini maka root di privat dan root jika kosong maka null

public void insert(char input) {

Node temp = new Node(input);

if (isEmpty())

root = temp;

else {

Node cursor = root,

parent = null;

while (cursor != null){

parent = cursor;

if (input < cursor.data)

cursor = cursor.left;

else

cursor = cursor.right;

//yaitu method untuk masukan jika kosong maka root temp jika node cursor maka root ,maka melakukan peruangan jika cursor ! =null ika input < cursor data maka cursor kalau tidak di kanan ya kiri.

}

if (input < parent.data) {

parent.left = temp;

return;

}

else {

parent.right = temp;

return;

}

}

}

//jika input < parent data maka .left =temp maka return, jika tidak maka parent.right = temp maka return

public boolean delete(int key){

Node cursor = root,

parent = null;

boolean found = false,

isLeftChild = true; //menandai apakah Node yang dihapus merupakan left child

if (!isEmpty()) {

while (cursor != null){

parent = cursor;

if (key == cursor.data){

found = true;

break;

}

//yaitu untuk method bagian delete datanya,maka node cursor = root, parent =null jika tidak kosong maka akan melakukan perulangan,maka cursor =null jika == cursor.data maka data true maka berhenti.

else if (key < cursor.data){

isLeftChild = true;

cursor = cursor.left;

}

else {

isLeftChild = false;

cursor = cursor.right;

}

}

//yaitu jjik a key < cursor.data maka is LeftChild maka benar maka cursor pada bagian kiri maka jika tidak maka false cursor akan ke kanan.

if (!found)

return false;

else {

if (cursor.left == null && cursor.right == null){

if (cursor == root)

root = null;

else if (isLeftChild)

parent.left = null;

else

parent.right = null;

}

//jika not found maka bila salah maka jika cursor kiri dan kanan maka di bandingkan == null,jika cursor == root maka jika tidak, maka menghapus node paling kecil maka parent.left atau kanan maka null

else if (cursor.left == null){

if (cursor == root)

root = cursor.right;

else if (isLeftChild)

parent.left = cursor.right;

else

parent.right = cursor.right;

}

//jika cursor == root maka bagian kanan, maka jika node paling kecil maka pada bagian kanan paretnya

else if (cursor.right == null){

if (cursor == root)

root = cursor.left;

else if (isLeftChild)

parent.left = cursor.left;

else

parent.right = cursor.left;

}

//yaitu jika cursor kana == null maka root == cursor.left jika node paling kecil maka terdapat pada kiri cursor.

else {

Node successor = getSuccessor(cursor);

if (cursor == root)

root = successor;

else if (isLeftChild)

parent.left = successor;

else

parent.right = successor;

successor.right = cursor.right;

}

}

}

return true;

//maka node successor jjka cursor == root,jika tidak maka node paling kecil,jika tidak maka successor pada bagian kanan atau kiri

}

private Node getSuccessor(Node localNode){

Node parent = null,

successor = localNode,

cursor = localNode.left;

while (cursor != null){

parent = successor;

successor = cursor;

cursor = cursor.right;

}

//yaitu maka bersifat private Node get Succesor maka localNode , maka pada cursur = local Node =.left maka melakukan perulangam jika not cursor maka=null, maka cursor pada bagian kanan.

if (successor != localNode.left){

parent.right = successor.left;

successor.left = localNode.left;

}

return successor;

}

//yaitu jika not successor = localNode.left , pada parent .right maka successor pada sebelah kiri, pada successor kiri maka LocalNode di kiri.

public void traverse(int tipe){

switch (tipe){

case 1:

System.out.print("\nPreorderersal:\n");

preOrder(root);

break;

//yaitu outputnya pilihan 1 yaitu Preorderersal,maka brek berhenti

case 2:

System.out.print("\nInorderersal:\n");

inOrder(root);

break;

//yaitu outputnya pilihan 2 yaitu Inorderersal ,maka brek berhenti

case 3:

System.out.print("\nPostorderersal:\n");

postOrder(root);

break;

}

System.out.println("\n");

}

//yaitu outputnya pilihan 3 yaitu Postorderersal ,maka brek berhenti

private void preOrder(Node localRoot){

if (localRoot == null) return;

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

preOrder(localRoot.right);

preOrder(localRoot.left);

}

//yaitu maka private voidOrder maka jika localRoot ==null ,maka keluaran datanya pre order maka localRoot dari kanan.

private void inOrder(Node localRoot){

if (localRoot == null) return;

inOrder(localRoot.left);

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

inOrder(localRoot.right);

}

// yaitu in Order jiks localRoot == null maka pada bagian kiri. Maka hasil keluaran, maka inOrder bagian kanan.

private void postOrder(Node localRoot){

if (localRoot == null) return;

postOrder(localRoot.left);

postOrder(localRoot.right);

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

}

//yaitu pada postOrder jika localRoot == null maka postOrder, local Root maka di kiri dan kanan maka keluaran datanya.

public static void main(String[] args) {

BinarySearchTree a = new BinarySearchTree();

a.insert('S');

a.insert('T');

a.insert('M');

a.insert('I');

a.insert('K');

a.traverse(1); //pre-order

a.traverse(2); //in-order

a.traverse(3); //post-order

}

}

// yaitu hasil data yang di masukan, STMIK maka yang preOrder 1 , yang inOder 2, yang post order

TUGAS

KESIMPULAN

Kita dapat menyimpulkan bahwa pada pohon adalah sekumpulan akar cabang dan simpul yang terhubung secara heraki, maka pada pohon biner memiliki simpulnya hanya maksimal 2 anak dari cabang kanan dan kiri. Dan Binery Search dengan Tree(pohon ) yang terdapat pada node kiri dan kanan .