Nama : Farid Aziz Wicaksono

Kelas : TI-1C

Absen : 14

1. Tugas
2. Tugas 1

Pada class BinarySearchTree, fungsi find() mengembalikan nilai bertipe boolean. Tambahkan fungsi baru yaitu findNode(), yang mengembalikan nilai bertipe Node.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | |  | | package jobsheet14;  public class BinarySearchTree {  Node root;  public BinarySearchTree() {  root = null;  }  boolean isEmpty() {  return root == null;  }  void add(int data) {  if (isEmpty()) {  root = new Node(data);  } else {  Node current = root;  while (true) {  if (data < current.data) {  if (current.left != null) {  current = current.left;  } else {  current.left = new Node(data);  break;  }  } else if (data > current.data) {  if (current.right != null) {  current = current.right;  } else {  current.right = new Node(data);  break;  }  } else {  break;  }  }  }  }  Node find(int data) {  Node temp;  Node current = root;  while (true) {  if (current.data == data) {  temp = current;  break;  } else if (data < current.data) {  current = current.left;  } else {  current = current.right;  }  }  return temp;  }  void traversepreorder(Node node) {  if (node != null) {  System.out.println(" " + node.data);  traversepreorder(node.left);  traversepreorder(node.right);  }  }  void traversepostorder(Node node) {  if (node != null) {  traversepostorder(node.left);  traversepostorder(node.right);  System.out.println(" " + node.data);  }  }  void traverseinorder(Node node) {  if (node != null) {  traverseinorder(node.left);  System.out.print(" " + node.data);  traverseinorder(node.right);  }  }  Node getsuccessor(Node del) {  Node successor = del.right;  Node successorParent = del;  while (successor.left != null) {  successorParent = successor;  successor = successor.left;  }  if (successor != del.right) {  successorParent.left = successor.right;  successor.right = del.right;  }  return successor;  }  public boolean delete(int id) {  Node parent = root;  Node current = root;  boolean isLeftChild = false;  while (current.data != id) {  parent = current;  if (current.data > id) {  isLeftChild = true;  current = current.left;  } else {  isLeftChild = true;  current = current.right;  }  if (current == null) {  return false;  }  }  if (current.left == null && current.right == null) {  if (current == root) {  root = null;  }  if (isLeftChild == true) {  parent.left = null;  } else {  parent.right = null;  }  } else if (current.right == null) {  if (current == root) {  root = current.left;  } else if (isLeftChild) {  parent.left = current.left;  } else {  parent.right = current.left;  }  } else if (current.left == null) {  if (current == root) {  root = current.right;  } else if (isLeftChild) {  parent.left = current.right;  } else {  parent.right = current.right;  }  } else if (current.left != null && current.right != null) {  Node successor = getsuccessor(current);  if (current == root) {  root = successor;  } else if (isLeftChild) {  parent.right = successor;  }  successor.left = current.left;  }  return true;  }  } |

1. Tugas 2

Pada class BinarySearchTree, buat fungsi getMax() untuk mencari node terbesar, dan getMin() untuk mencari node terkecil.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | | package jobsheet14;  public class BinarySearchTree {  Node root;  public BinarySearchTree() {  root = null;  }  boolean isEmpty() {  return root == null;  }  void add(int data) {  if (isEmpty()) {  root = new Node(data);  } else {  Node current = root;  while (true) {  if (data < current.data) {  if (current.left != null) {  current = current.left;  } else {  current.left = new Node(data);  break;  }  } else if (data > current.data) {  if (current.right != null) {  current = current.right;  } else {  current.right = new Node(data);  break;  }  } else {  break;  }  }  }  }  Node find(int data) {  Node temp;  Node current = root;  while (true) {  if (current.data == data) {  temp = current;  break;  } else if (data < current.data) {  current = current.left;  } else {  current = current.right;  }  }  return temp;  }    int Min() {  Node current = root;  while (true) {  if (current.left != null) {  current = current.left;  } else {  break;  }  }  return current.data;  }    int Max() {  Node current = root;  while (true) {  if (current.right != null) {  current = current.right;  } else {  break;  }  }  return current.data;  }  void traversepreorder(Node node) {  if (node != null) {  System.out.println(" " + node.data);  traversepreorder(node.left);  traversepreorder(node.right);  }  }  void traversepostorder(Node node) {  if (node != null) {  traversepostorder(node.left);  traversepostorder(node.right);  System.out.println(" " + node.data);  }  }  void traverseinorder(Node node) {  if (node != null) {  traverseinorder(node.left);  System.out.print(" " + node.data);  traverseinorder(node.right);  }  }  Node getsuccessor(Node del) {  Node successor = del.right;  Node successorParent = del;  while (successor.left != null) {  successorParent = successor;  successor = successor.left;  }  if (successor != del.right) {  successorParent.left = successor.right;  successor.right = del.right;  }  return successor;  }  public boolean delete(int id) {  Node parent = root;  Node current = root;  boolean isLeftChild = false;  while (current.data != id) {  parent = current;  if (current.data > id) {  isLeftChild = true;  current = current.left;  } else {  isLeftChild = true;  current = current.right;  }  if (current == null) {  return false;  }  }  if (current.left == null && current.right == null) {  if (current == root) {  root = null;  }  if (isLeftChild == true) {  parent.left = null;  } else {  parent.right = null;  }  } else if (current.right == null) {  if (current == root) {  root = current.left;  } else if (isLeftChild) {  parent.left = current.left;  } else {  parent.right = current.left;  }  } else if (current.left == null) {  if (current == root) {  root = current.right;  } else if (isLeftChild) {  parent.left = current.right;  } else {  parent.right = current.right;  }  } else if (current.left != null && current.right != null) {  Node successor = getsuccessor(current);  if (current == root) {  root = successor;  } else if (isLeftChild) {  parent.right = successor;  }  successor.left = current.left;  }  return true;  }  } |

1. Tugas 3

Modifikasi program diatas agar memiliki menu sebagai berikut:

a. Insert

b. Delete

c. Find

d. Display

e. Keluar

|  |  |
| --- | --- |
| 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 | package jobsheet14;  import java.util.Scanner;  public class BinarySearchTreeMain {  static void menu() {  System.out.println("=============");  System.out.println("= menu =");  System.out.println("1. Insert");  System.out.println("2. Delete");  System.out.println("3. Find");  System.out.println("4. Display");  System.out.println("5. keluar");  System.out.println("=============");  }  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int pil = 0;  BinarySearchTree aa = new BinarySearchTree();  do {  menu();  System.out.print("Masukan Pilihan : ");  pil = sc.nextInt();  switch (pil) {  case 1:  System.out.println("Masukkan angka = ");  int add = sc.nextInt();  aa.add(add);  break;  case 2:  System.out.println("Hapus = ");  int id = sc.nextInt();  aa.delete(id);  break;  case 3:  System.out.print("Cari = ");  int temu = sc.nextInt();  Node cari = aa.find(temu);  if (cari.data == temu) {  System.out.println("Ada");  } else {  System.out.println("Tidak Ada");  }  break;  case 4:  aa.traverseinorder(aa.root);  System.out.println("");  aa.traversepreorder(aa.root);  System.out.println("");  aa.traversepostorder(aa.root);  System.out.println("");  break;  }  } while (pil != 5);  }  } |