  
 Name : Azaria Cindy Sahasika

Number Id : 2341760169 / 06

Class : 1G – Business Information System

Lesson : Algorithm and Data Structure

Material : Material 11 – Tree

Github Link : <https://github.com/azariacindy/algorithm-ds>

JOBSHEET

Tree

12.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. memahami model *Tree* khususnya *Binary Tree*
2. membuat dan mendeklarasikan struktur algoritma *Binary Tree*.
3. menerapkan dan mengimplementasikan algoritma *Binary Tree* dalam kasus *Binary Search Tree*

12.2 Kegiatan Praktikum 1

Implementasi Binary Search Tree menggunakan Linked List (45 Menit)

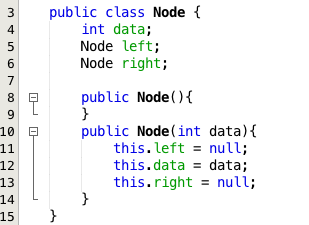
12.2.1 Percobaan 1

Pada percobaan ini akan diimplementasikan Binary Search Tree dengan operasi dasar, dengan menggunakan array (praktikum 2) dan linked list (praktikum 1). Sebelumnya, akan dibuat class Node, dan Class BinaryTree

|  |
| --- |
| Node |
| data: int  left: Node  right: Node |
| Node(left: Node, data:int, right:Node) |

|  |
| --- |
| BinaryTree |
| root: Node  size : int |
| DoubleLinkedLists()  add(data: int): void  find(data: int) : boolean  traversePreOrder (node : Node) : void  traversePostOrder (node : Node) void  traverseInOrder (node : Node): void  getSuccessor (del: Node)  add(item: int, index:int): void  delete(data: int): void |

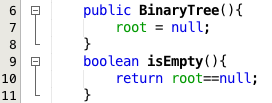
1. Buatlah class **Node**, **BinaryTree** dan **BinaryTreeMain**
2. Di dalam class **Node**, tambahkan atribut **data**, **left** dan **right**, serta konstruktor default dan berparameter.

****

1. Di dalam class **BinaryTree**, tambahkan atribut **root**.

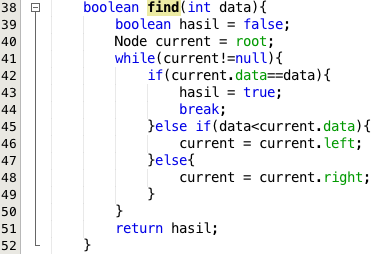
****

1. Tambahkan konstruktor default dan method **isEmpty()** di dalam class **BinaryTree**

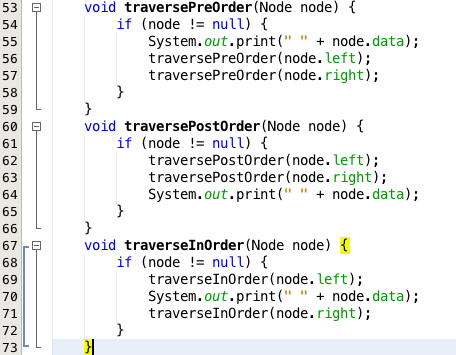
****

* 1. Tambahkan method **add()** di dalam class **BinaryTree**. Di bawah ini proses penambahan node **tidak dilakukan secara rekursif**, agar lebih mudah dilihat alur proses penambahan node dalam tree. Sebenarnya, jika dilakukan dengan proses rekursif, penulisan kode akan lebih efisien.sus no****

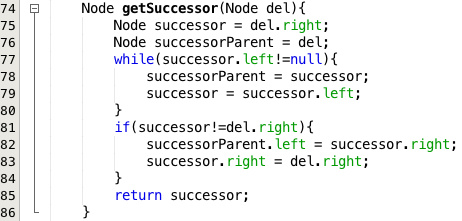
1. Tambahkan method find()

****

1. Tambahkan method **traversePreOrder()**, **traverseInOrder()** dan **traversePostOrder()**. Method traverse digunakan untuk mengunjungi dan menampilkan node-node dalam tree, baik dalam mode pre-order, in-order maupun post-order.

****

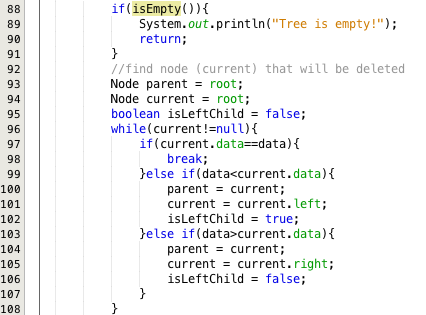
1. Tambahkan method **getSuccessor()**. Method ini akan digunakan ketika proses penghapusan node yang memiliki 2 child.

****

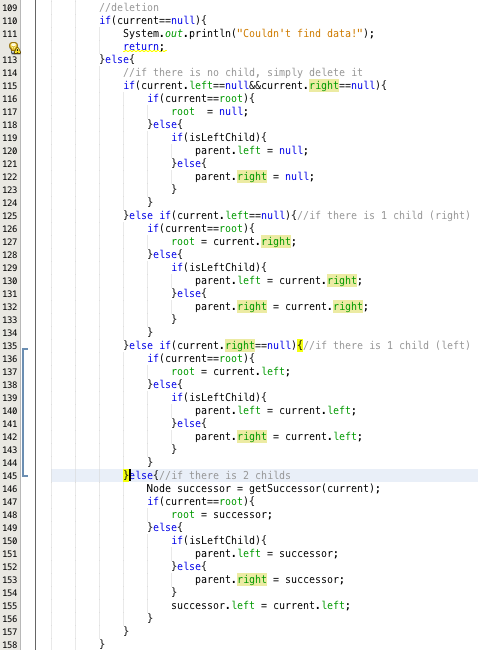
1. Tambahkan method **delete()**.

****

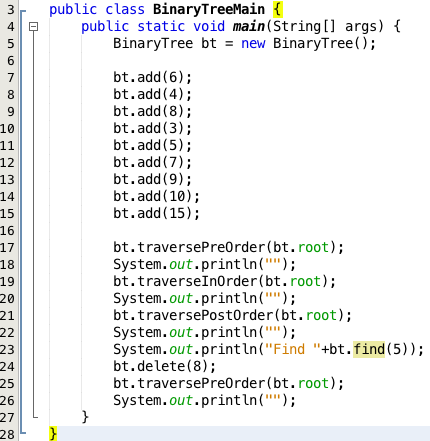
Di dalam method delete tambahkan pengecekan apakah tree kosong, dan jika tidak cari posisi node yang akan di hapus.



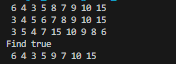
Kemudian tambahkan proses penghapusan terhadap node current yang telah ditemukan.



1. Buka class BinaryTreeMain dan tambahkan method main().

****

1. Compile dan jalankan class BinaryTreeMain untuk mendapatkan simulasi jalannya program tree yang telah dibuat.
2. Amati hasil running tersebut.



12.2.2 Pertanyaan Percobaan

* 1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
* Karena binary search tree node ini memiliki properti dimana nilai semua node di subTree left lebih kecil daripada nilai node parent, dan nilai semua node di subtree right lebih besar daripada nilai node parent. Pencarian data dilakukan secara lebih efisien dengan membandingkan nilai yang dicari dengan nilai node saat ini dan memilih subtree yang sesuai.
  1. Untuk apakah di class **Node**, kegunaan dari atribut **left** dan **right**?
* Left : menunjuk ke node child yang lebih kecil dalam binary search tree saat ini.
* Right : menunjuk ke node child yang lebih besar dalam binary search tree saat ini.
  1. a. Untuk apakah kegunaan dari atribut **root** di dalam class **BinaryTree**?

- untuk menyimpan referensi ke node akar dari binary tree

b. Ketika objek tree pertama kali dibuat, apakah nilai dari **root**?

- unutk menunjukkan bahwa tree tersebut kosong dan belum memiliki node apa pun.

* 1. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
* Node baru akan ditambahkan menjadi root dari tree tersebut. Kemudian dilakukan pengalokasian node baru dan menetapkan atribut root ke node baru.
  1. Perhatikan method **add()**, di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?

**if(data<current.data){**

**if(current.left!=null){**

**current = current.left;**

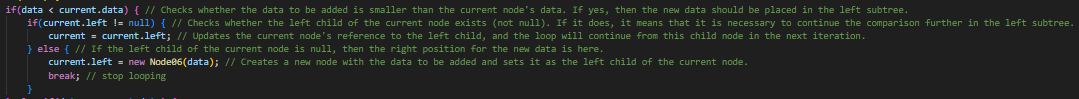
**}else{**

**current.left = new Node(data);**

**break;**

**}**

**}**

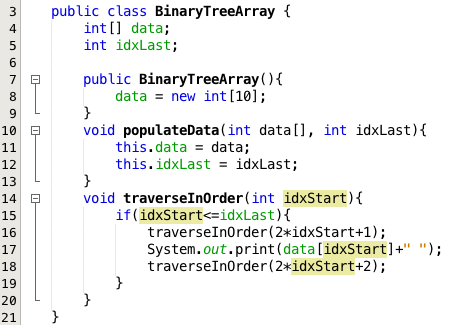


12.3 Kegiatan Praktikum 2

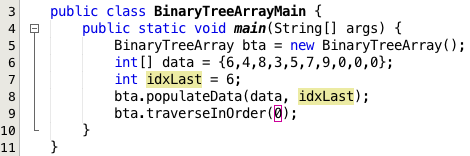
Implementasi binary tree dengan array (45 Menit)

13.3.1 Tahapan Percobaan

1. Di dalam percobaan implementasi binary tree dengan array ini, data tree disimpan dalam array dan langsung dimasukan dari method main(), dan selanjutnya akan disimulasikan proses traversal secara inOrder.
2. Buatlah class **BinaryTreeArray** dan **BinaryTreeArrayMain**
3. Buat atribut **data** dan **idxLast** di dalam class **BinaryTreeArray**. Buat juga method **populateData()** dan **traverseInOrder().**

****

1. Kemudian dalam class **BinaryTreeArrayMain** buat method main() seperti gambar berikut ini.

****

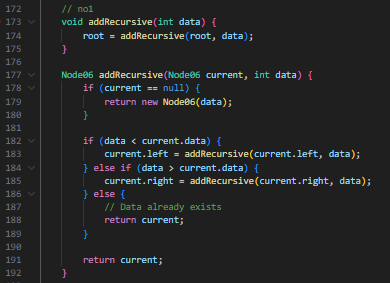
1. Jalankan class **BinaryTreeArrayMain** dan amati hasilnya! 

12.3.2 Pertanyaan Percobaan

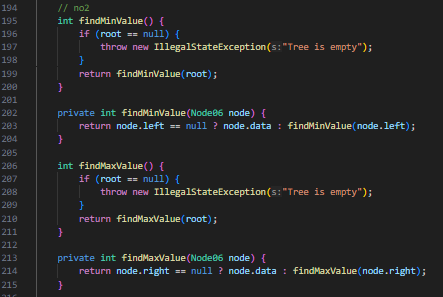
* 1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray**?
* Fungsi atribut data: menyimpan data array
* Fungsi idxLast: menyimpan batas index array
  1. Apakah kegunaan dari method **populateData()**?
* Unutk menunjukkan data pada idxLast
  1. Apakah kegunaan dari method **traverseInOrder()**?
* Untuk menelusuri tree untuk menggunakan metode order dan prinsip left visit right.
  1. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masin-masing?
* Jika array dari 0, maka left child index ke-5 dan right child index ke-6.
  1. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
* Untuk menunjukkan idxLast / max index arraynya merupakan 6.
  1. **Tugas Praktikum**

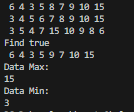
**Waktu pengerjaan: 90 menit**

* 1. Buat method di dalam class **BinaryTree** yang akan menambahkan node dengan cara rekursif.

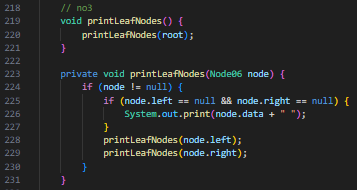


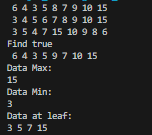
* 1. Buat method di dalam class **BinaryTree** untuk menampilkan nilai paling kecil dan yang paling besar yang ada di dalam tree.



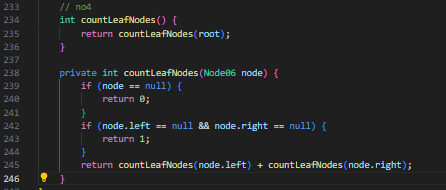


* 1. Buat method di dalam class **BinaryTree** untuk menampilkan data yang ada di leaf.





* 1. Buat method di dalam class **BinaryTree** untuk menampilkan berapa jumlah leaf yang ada di dalam tree.



1. Modifikasi class **BinaryTreeArray**, dan tambahkan :

* method **add(int data)** untuk memasukan data ke dalam tree
* method **traversePreOrder()** dan **traversePostOrder()**

