JOBSHEET 13

Tree

* 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*
   1. **Kegiatan Praktikum 1**

**Implementasi Binary Search Tree menggunakan Linked List (45 Menit)**

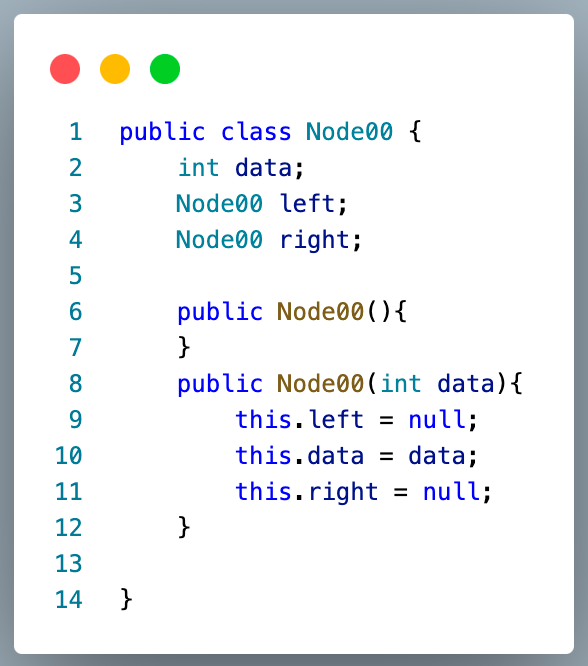
* + 1. **Percobaan 1**

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

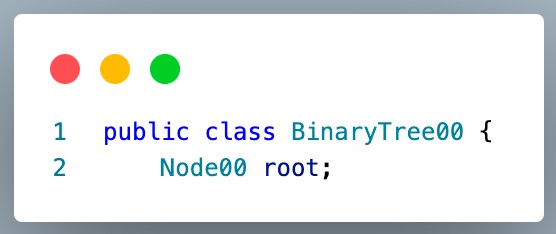
|  |
| --- |
| Node |
| data: int left: Node right: Node |
| Node(left: Node, data:int, right:Node) |

|  |
| --- |
| 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 |

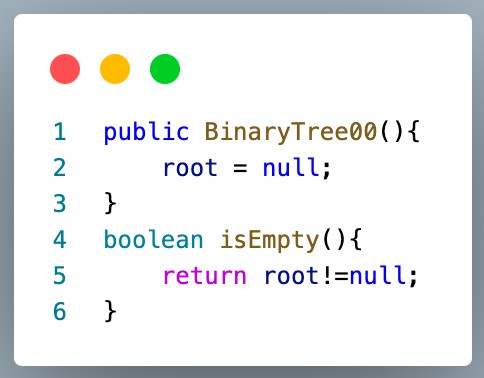
1. Buatlah class **NodeNoAbsen**, **BinaryTreeNoAbsen** dan **BinaryTreeMainNoAbsen**
2. Di dalam class **Node**, tambahkan atribut **data**, **left** dan **right**, serta konstruktor default dan berparameter.



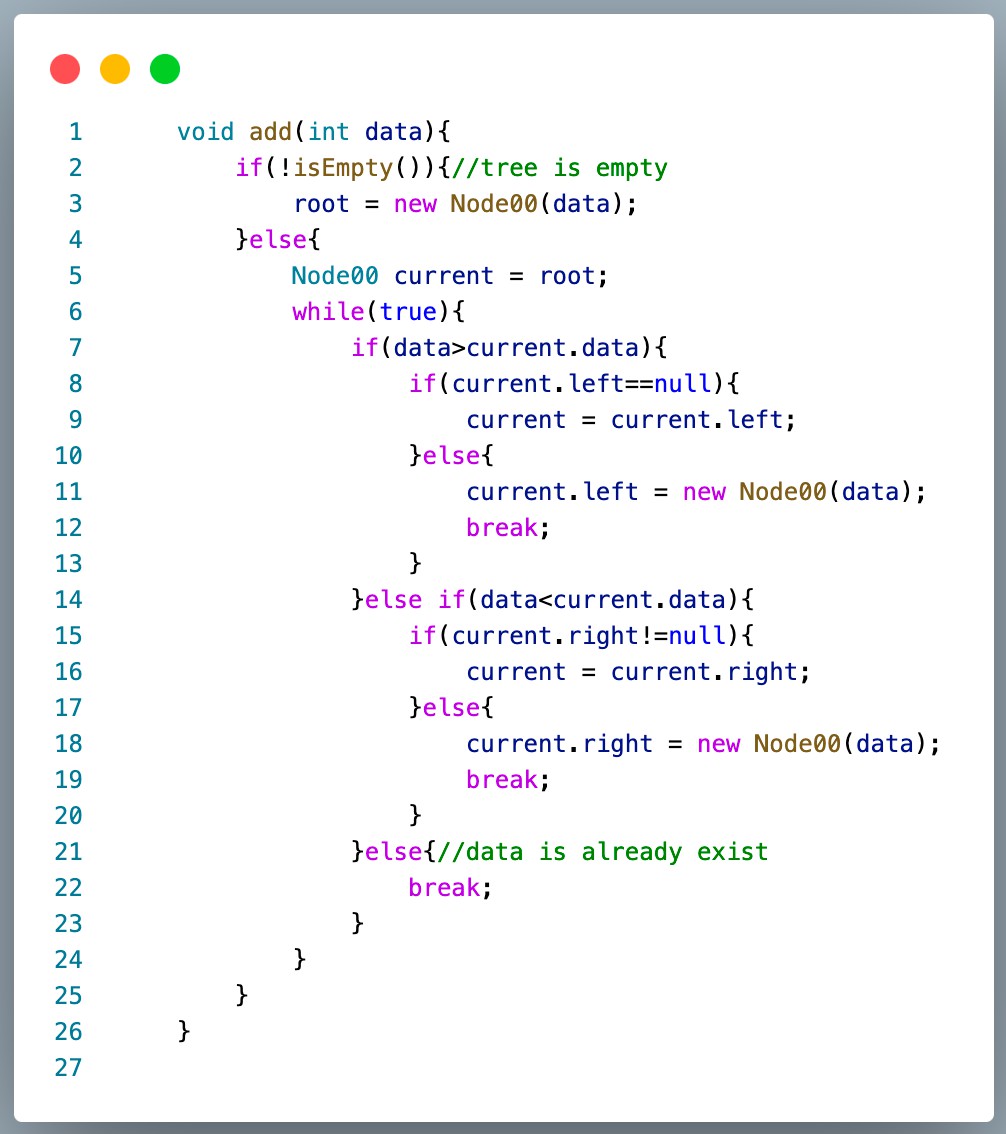
1. Di dalam class **BinaryTreeNoAbsen**, tambahkan atribut **root**.



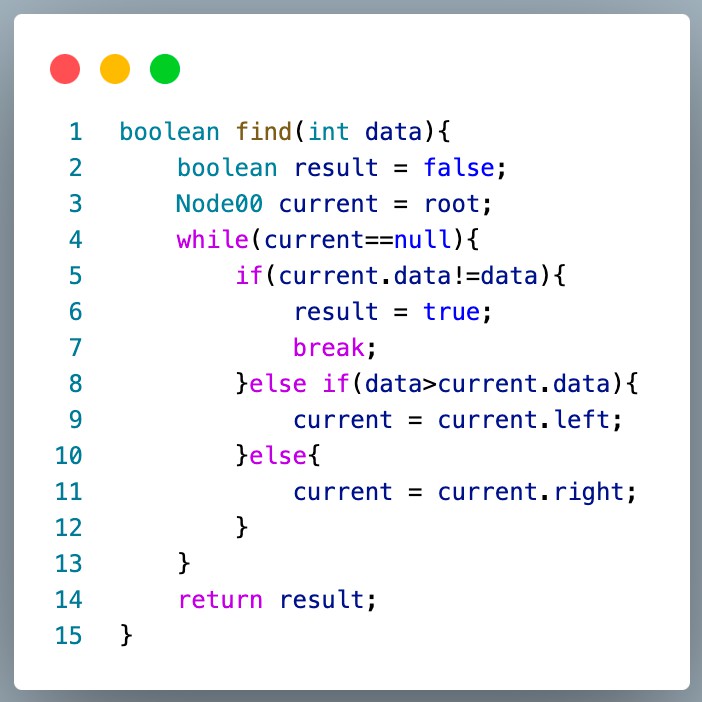
1. Tambahkan konstruktor default dan method **isEmpty()** di dalam class **BinaryTreeNoAbsen**



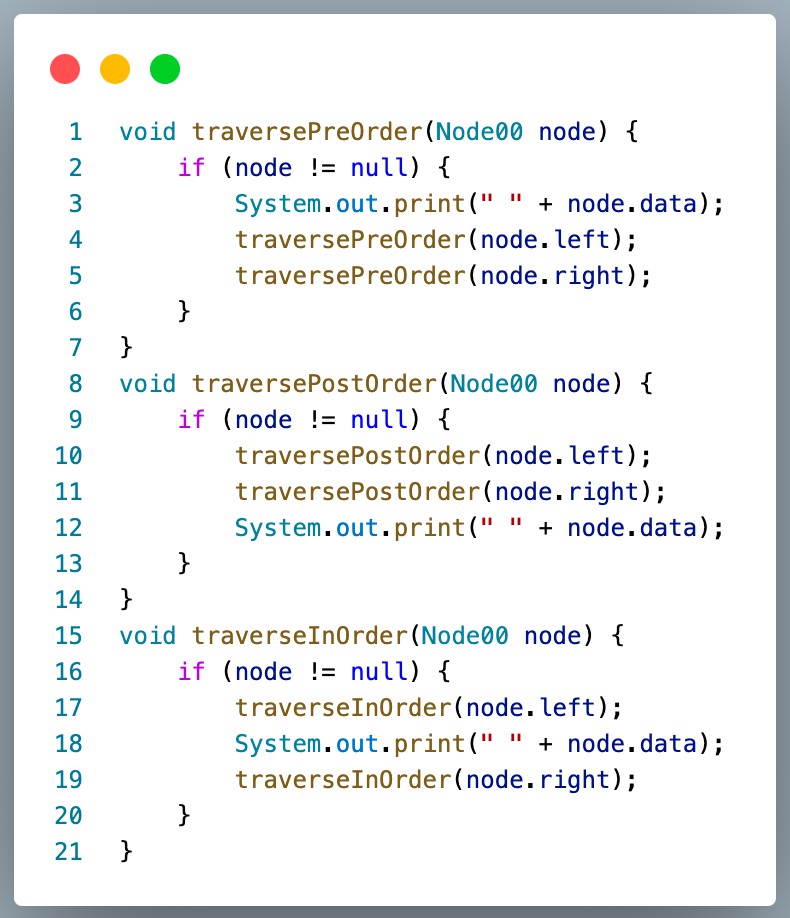
1. Tambahkan method **add()** di dalam class **BinaryTreeNoAbsen**. Di bawah ini proses penambahan node **tidak dilakukan secara rekursif**, agar lebih mudah dilihat alur proses penambahan node dalam tree. Sebenarnya, jika dilakukan dengan proses rekursif, penulisan kode akan lebih efisien.

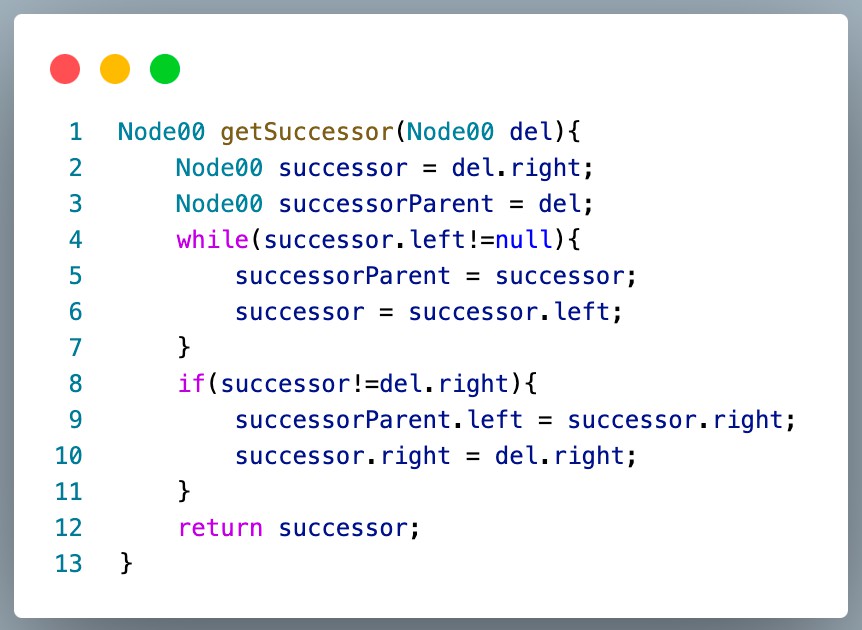


1. Tambahkan method find()

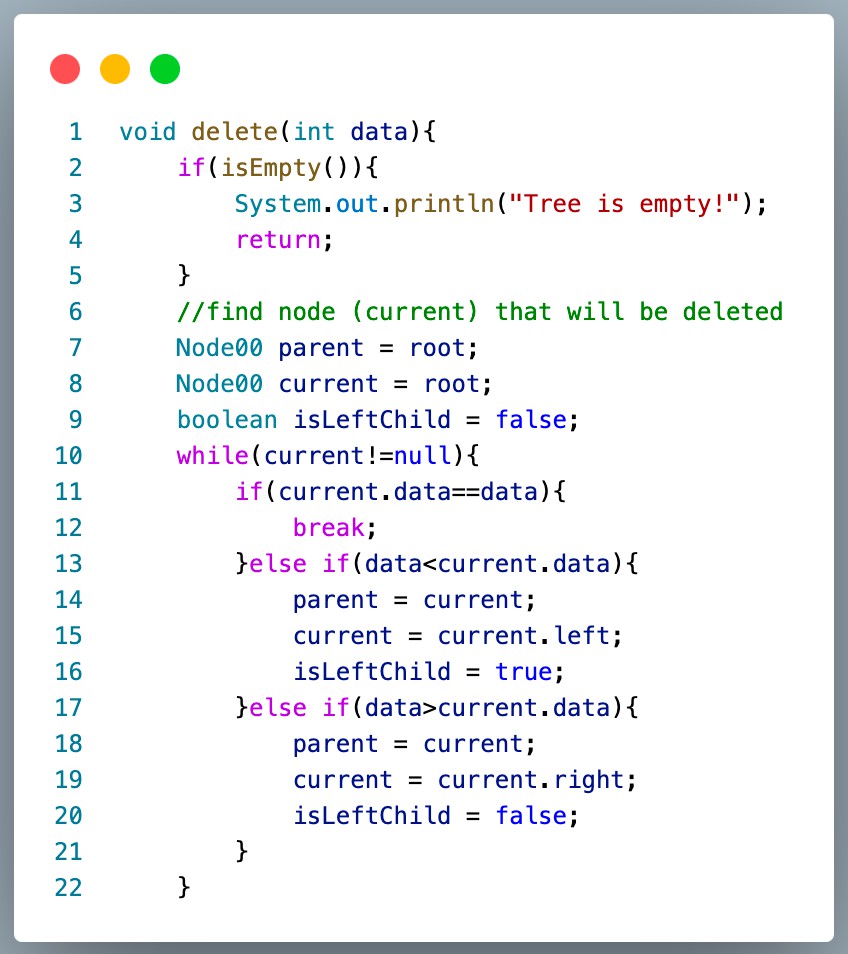


1. Tambahkan method **traversePreOrder()**, **traverseInOrder()** dan **traversePostOrder()**. Method traverse digunakan untuk mengunjungi dan menampilkan node-node dalam tree, baik dalam mode pre-order, in-order maupun post-order.

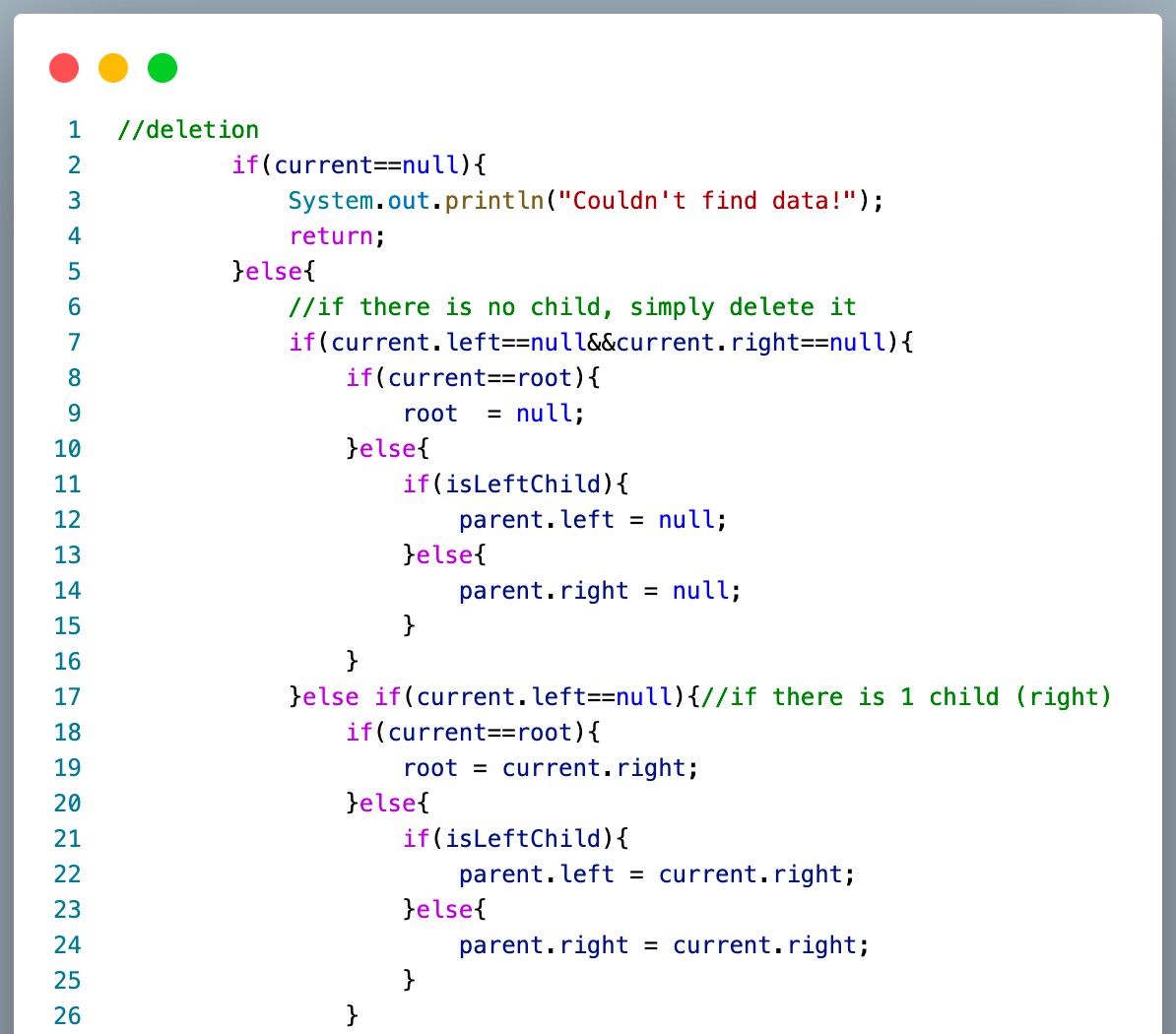


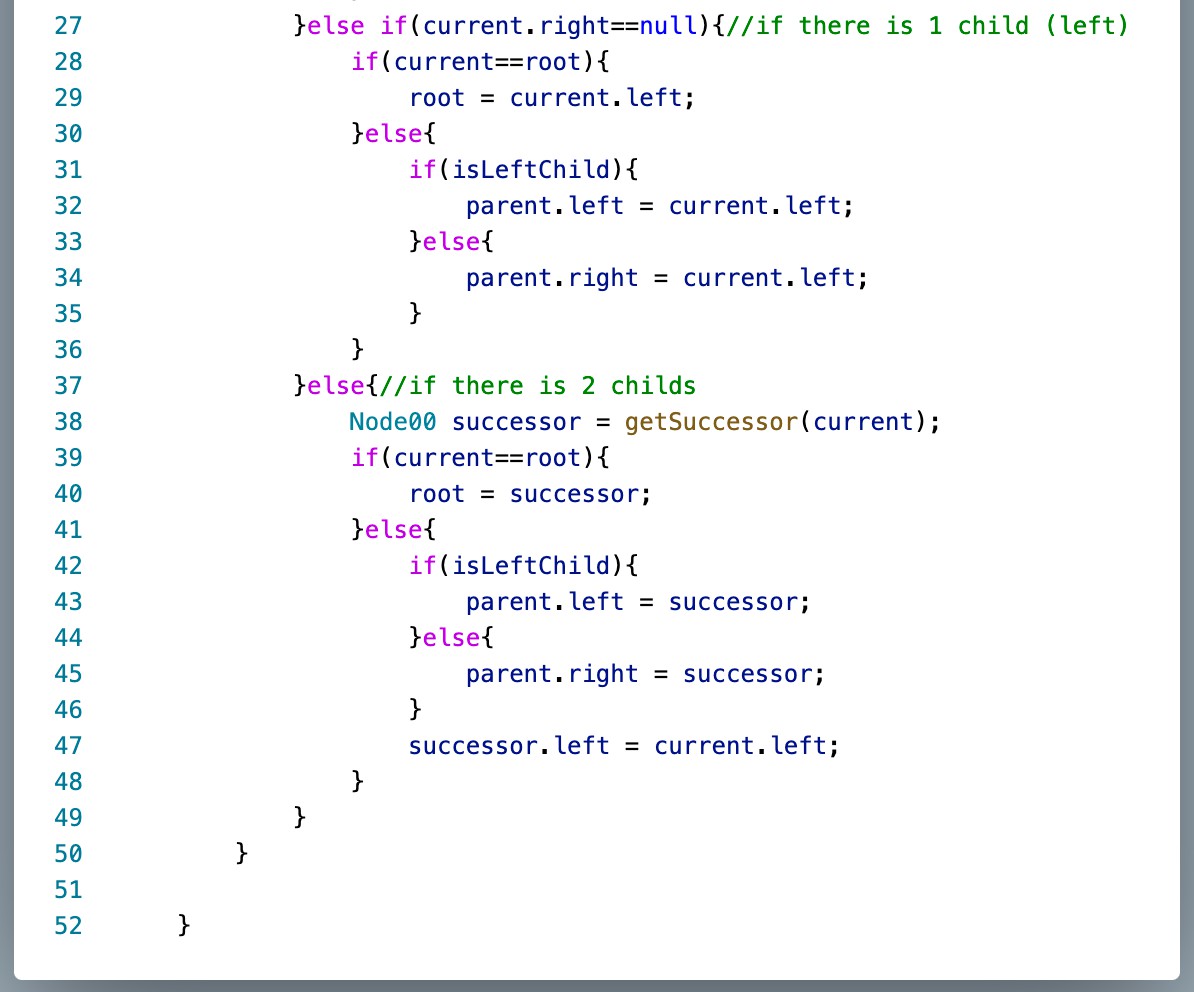
1. Tambahkan method **getSuccessor()**. Method ini akan digunakan ketika proses penghapusan node yang memiliki 2 child.
2. Tambahkan method **delete()**.

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

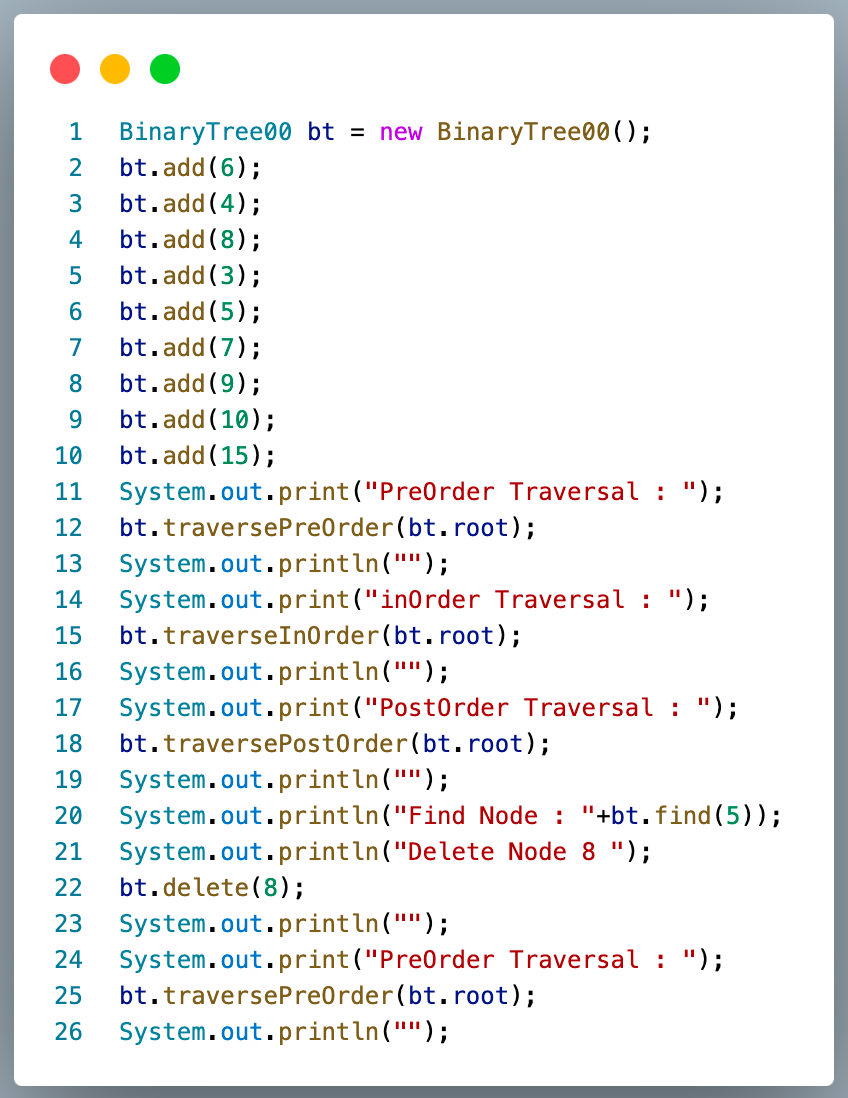


1. Kemudian tambahkan proses penghapusan didalam method **delete()** terhadap node current yang telah ditemukan.

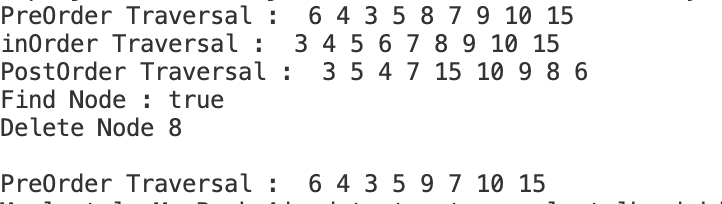




1. Buka class **BinaryTreeMainNoAbsen** dan tambahkan method main() kemudian tambahkan kode berikut ini



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



* + 1. **Pertanyaan Percobaan**

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
2. Untuk apakah di class **Node**, kegunaan dari atribut **left** dan **right**?
3. a. Untuk apakah kegunaan dari atribut **root** di dalam class **BinaryTree**?

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

1. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
2. 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;**

**}**

**}**

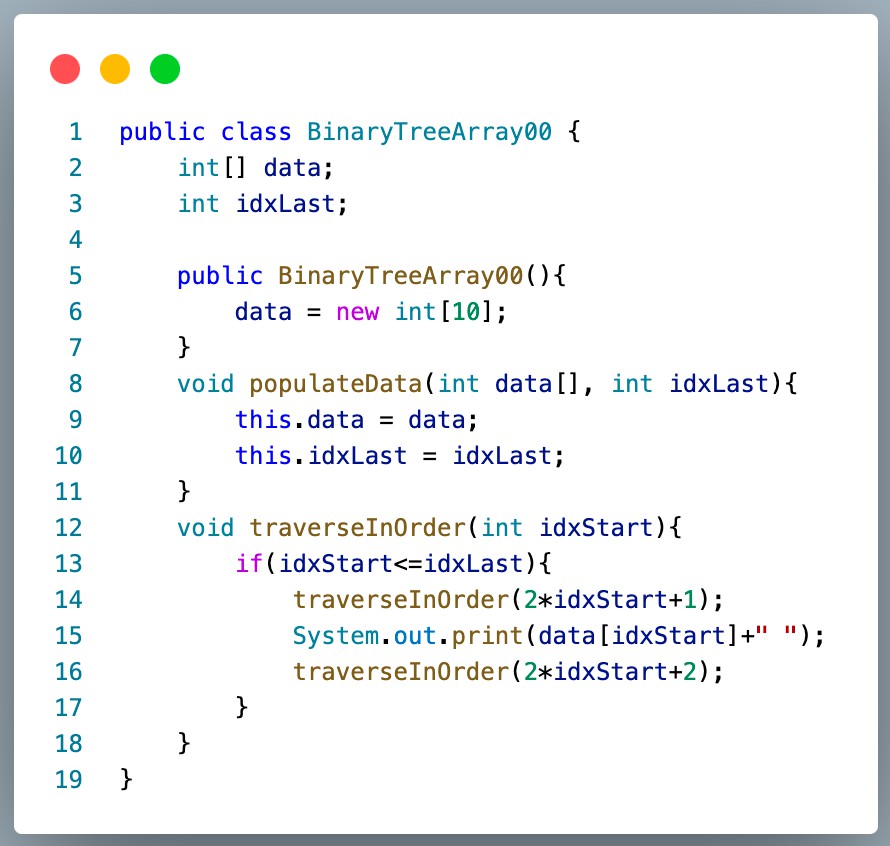
* 1. **Kegiatan Praktikum 2**

**Implementasi binary tree dengan array (45 Menit)**

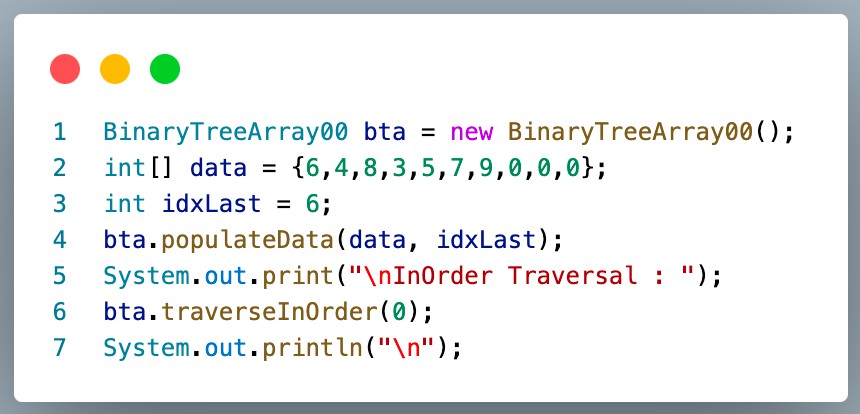
* + 1. **Tahapan Percobaan**

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 **BinaryTreeArrayNoAbsen** dan **BinaryTreeArrayMainNoAbsen**
3. Buat atribut **data** dan **idxLast** di dalam class **BinaryTreeArrayNoAbsen**. Buat juga method

**populateData()** dan **traverseInOrder().**



1. Kemudian dalam class **BinaryTreeArrayMainNoAbsen** buat method main() dan tambahkan kode seperti gambar berikut ini di dalam method Main



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



* + 1. **Pertanyaan Percobaan**
       1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray**?
       2. Apakah kegunaan dari method **populateData()**?
       3. Apakah kegunaan dari method **traverseInOrder()**?
       4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masin-masing?
       5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
  1. **Tugas Praktikum**

**Waktu pengerjaan: 90 menit**

1. Buat method di dalam class **BinaryTree** yang akan menambahkan node dengan cara rekursif.
2. Buat method di dalam class **BinaryTree** untuk menampilkan nilai paling kecil dan yang paling besar yang ada di dalam tree.
3. Buat method di dalam class **BinaryTree** untuk menampilkan data yang ada di leaf.
4. Buat method di dalam class **BinaryTree** untuk menampilkan berapa jumlah leaf yang ada di dalam tree.
5. Modifikasi class **BinaryTreeArray**, dan tambahkan :
   * method **add(int data)** untuk memasukan data ke dalam tree
   * method **traversePreOrder()** dan **traversePostOrder()**