JOBSHEET 13

Tree

# 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*

# Kegiatan Praktikum 1

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

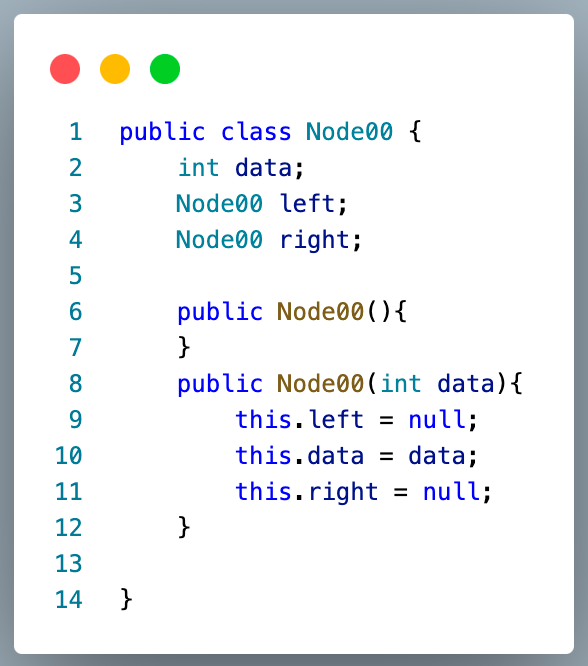
# Percobaan 1

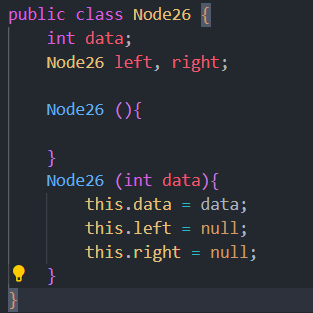
Pada percobaan ini akan diimplementasikan Binary Search Tree dengan operasi dasar, dengan menggunakan array (praktikum 2) dan linked list (praktikum 1). Sebelumnya, akan dibuat class Node, dan Class BinaryTree

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

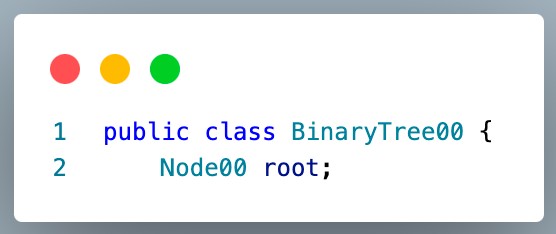
|  |
| --- |
| BinaryTree |
| root: Node size : int |
| DoubleLinkedLists() add(data: int): void find(data: int) : boolean  traversePreOrder (node : Node) : void traversePostOrder (node : Node) void traverseInOrder (node : Node): void getSuccessor (del: Node)  add(item: int, index:int): void delete(data: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

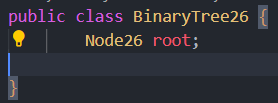
1. Buatlah class **NodeNoAbsen**, **BinaryTreeNoAbsen** dan **BinaryTreeMainNoAbsen**
2. Di dalam class **Node**, tambahkan atribut **data**, **left** dan **right**, serta konstruktor default dan berparameter.



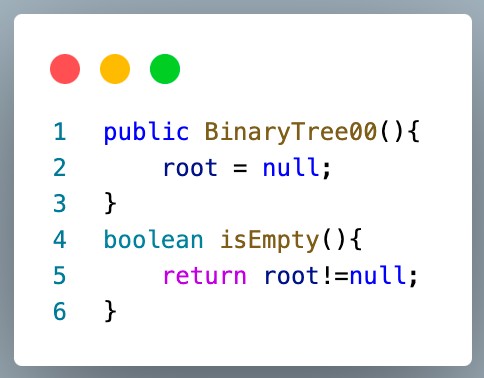


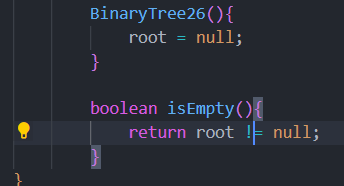
1. Di dalam class **BinaryTreeNoAbsen**, tambahkan atribut **root**.



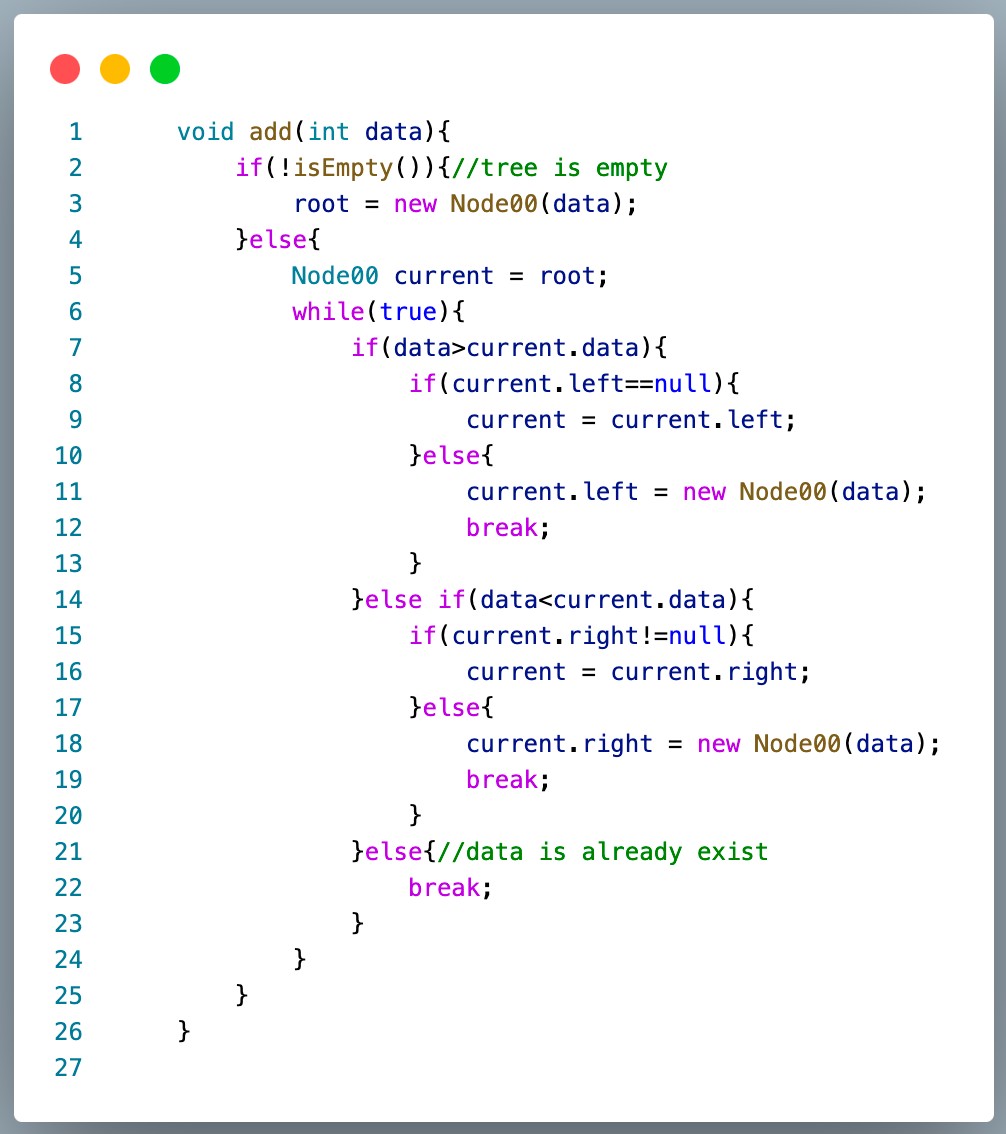


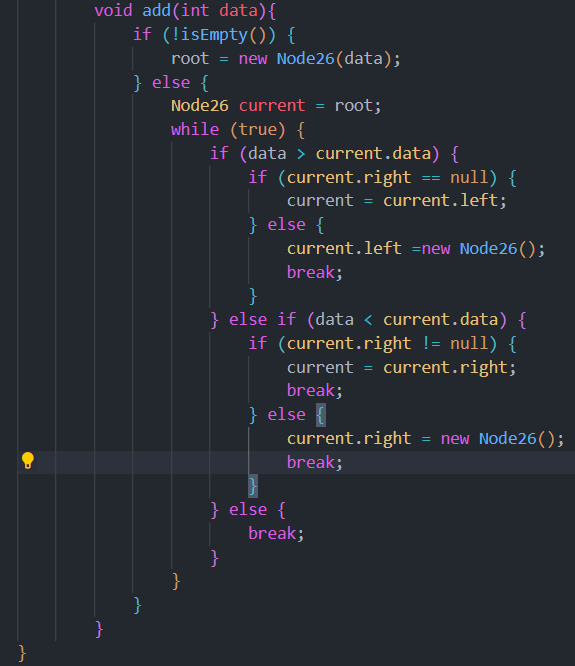
1. Tambahkan konstruktor default dan method **isEmpty()** di dalam class **BinaryTreeNoAbsen**



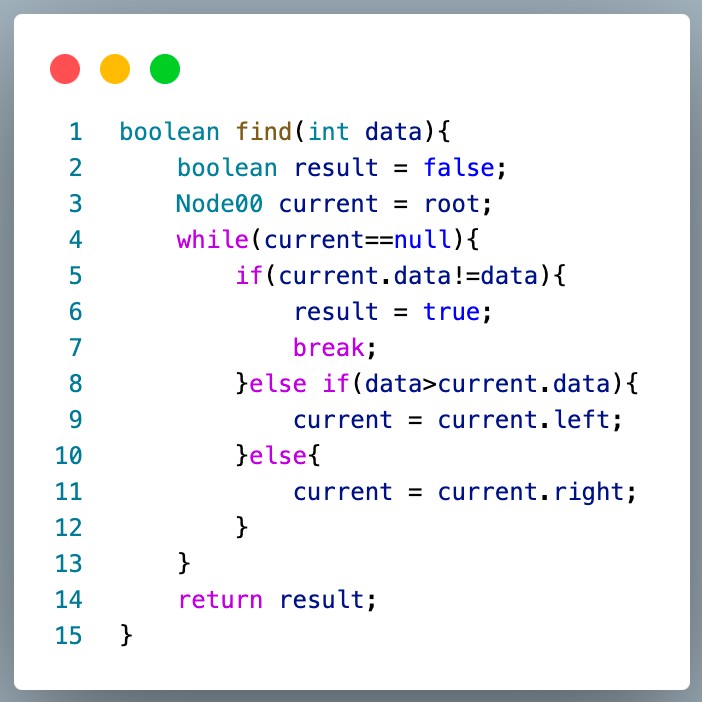
****

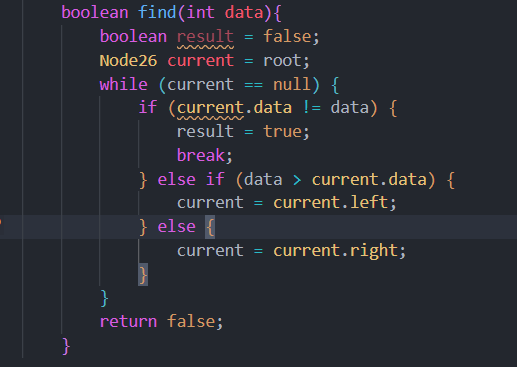
1. Tambahkan method **add()** di dalam class **BinaryTreeNoAbsen**. Di bawah ini proses penambahan node **tidak dilakukan secara rekursif**, agar lebih mudah dilihat alur proses penambahan node dalam tree. Sebenarnya, jika dilakukan dengan proses rekursif, penulisan kode akan lebih efisien.



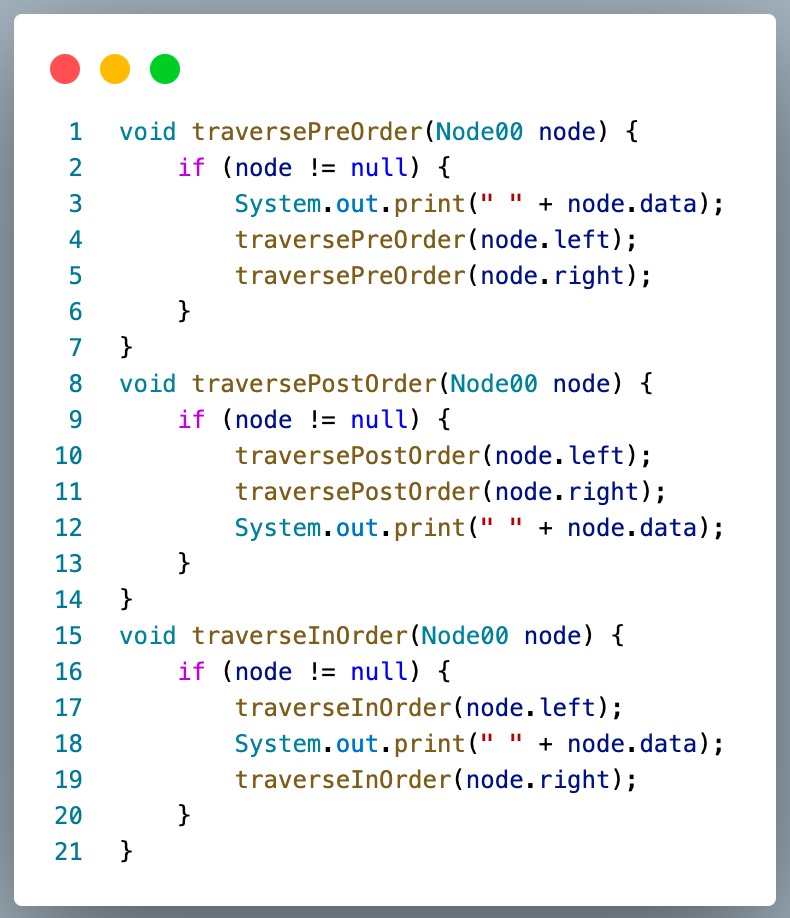


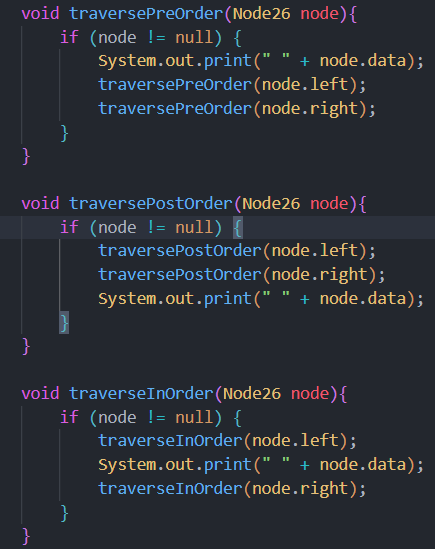
1. Tambahkan method find()



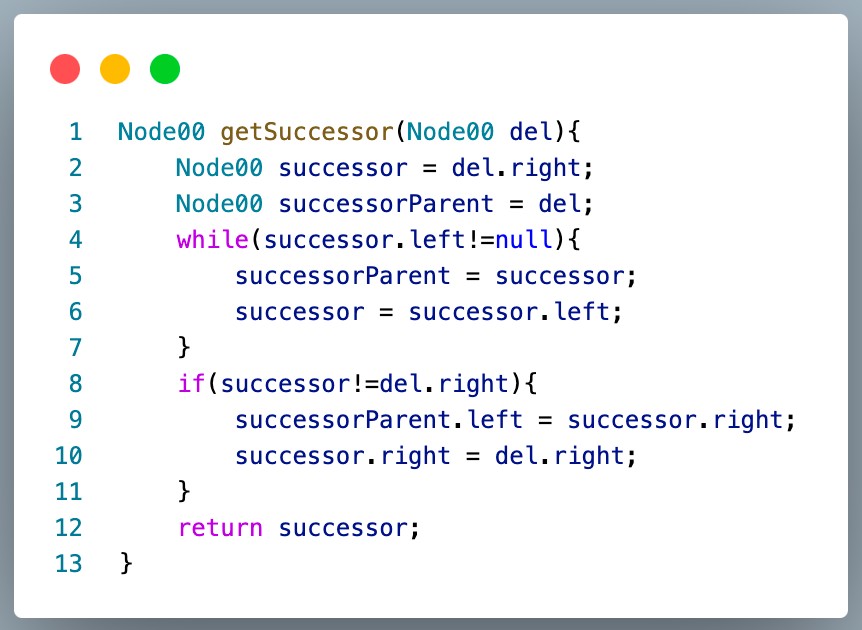


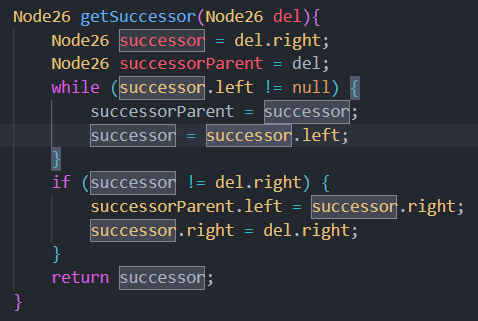
1. Tambahkan method **traversePreOrder()**, **traverseInOrder()** dan **traversePostOrder()**. Method traverse digunakan untuk mengunjungi dan menampilkan node-node dalam tree, baik dalam mode pre-order, in-order maupun post-order.





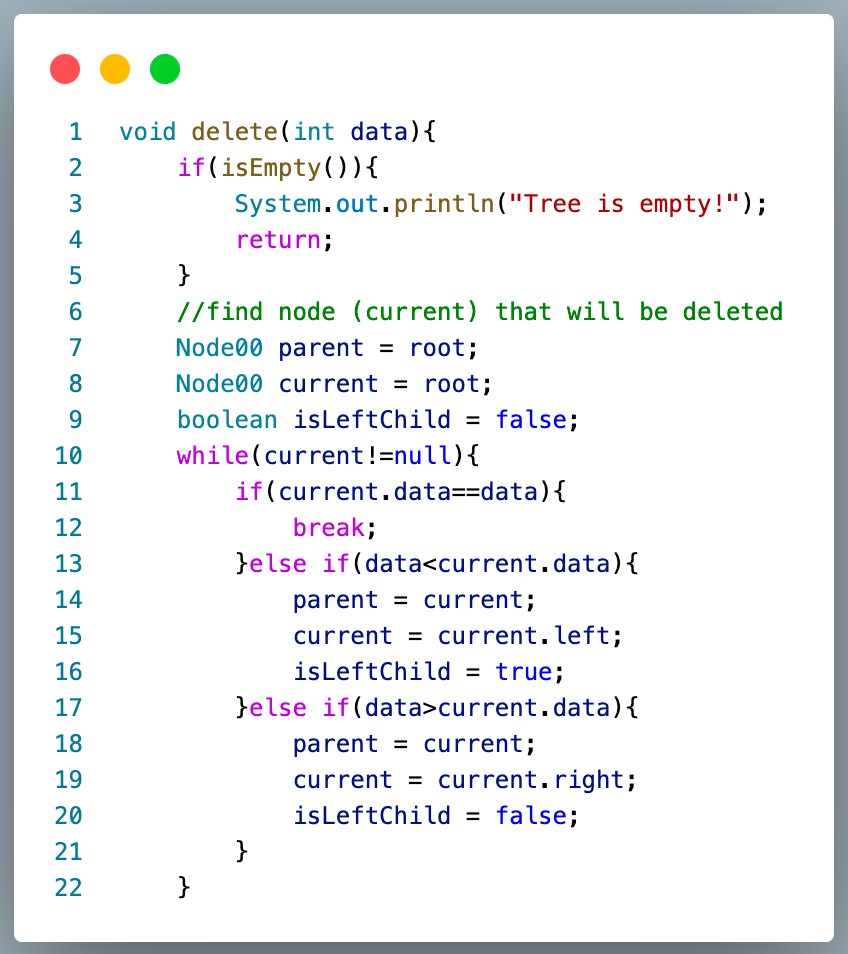
1. Tambahkan method **getSuccessor()**. Method ini akan digunakan ketika proses penghapusan node yang memiliki 2 child.

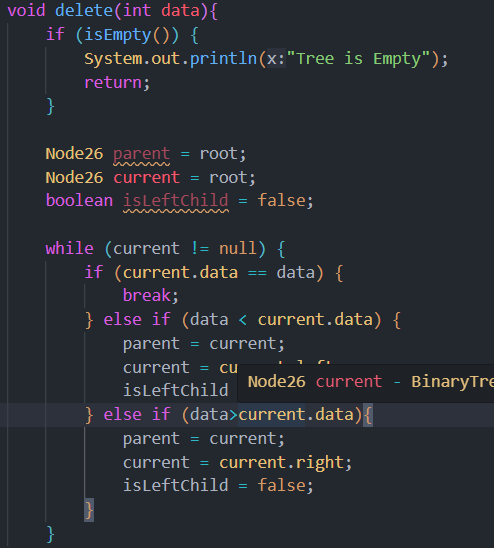




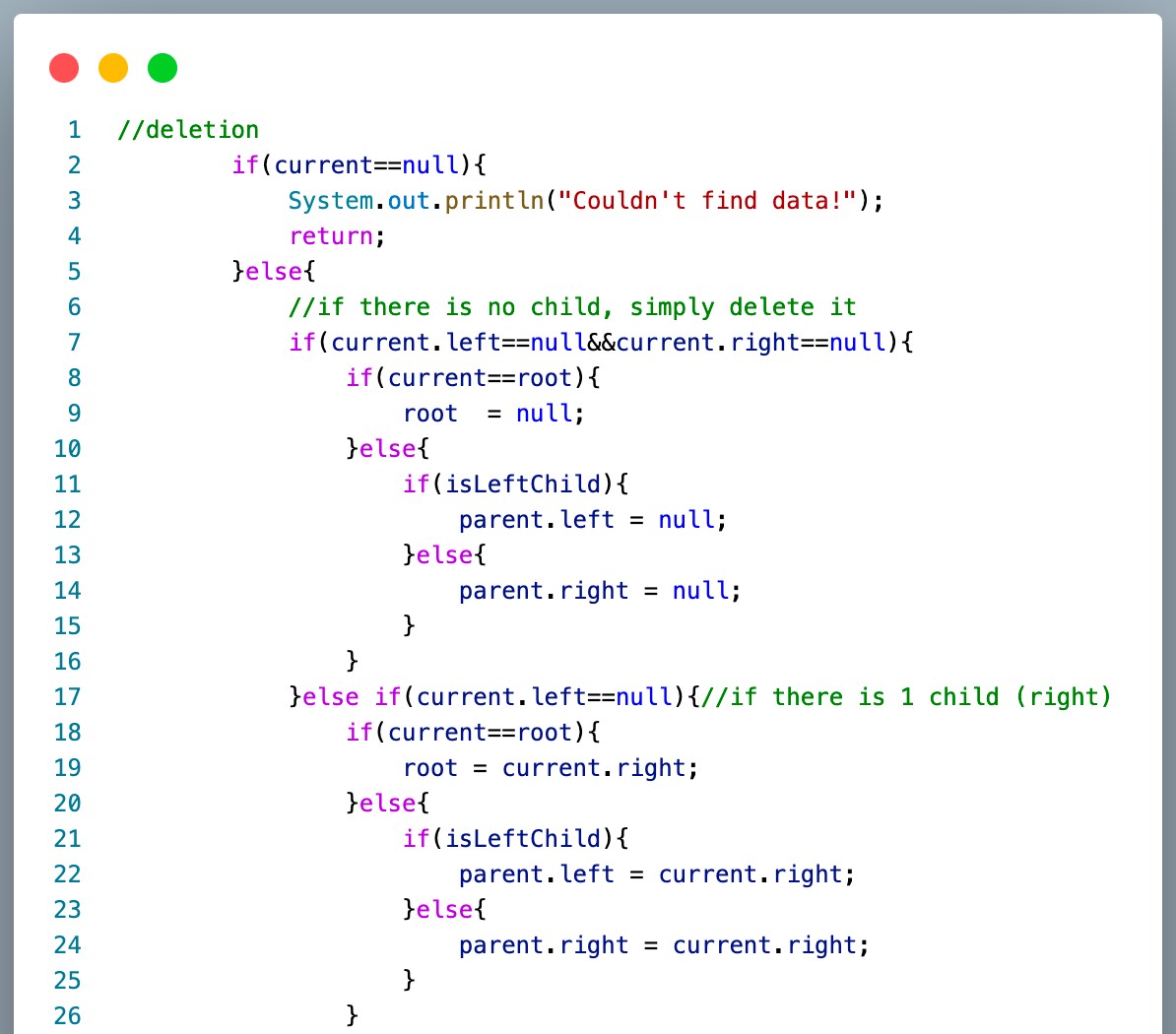
1. Tambahkan method **delete()**.

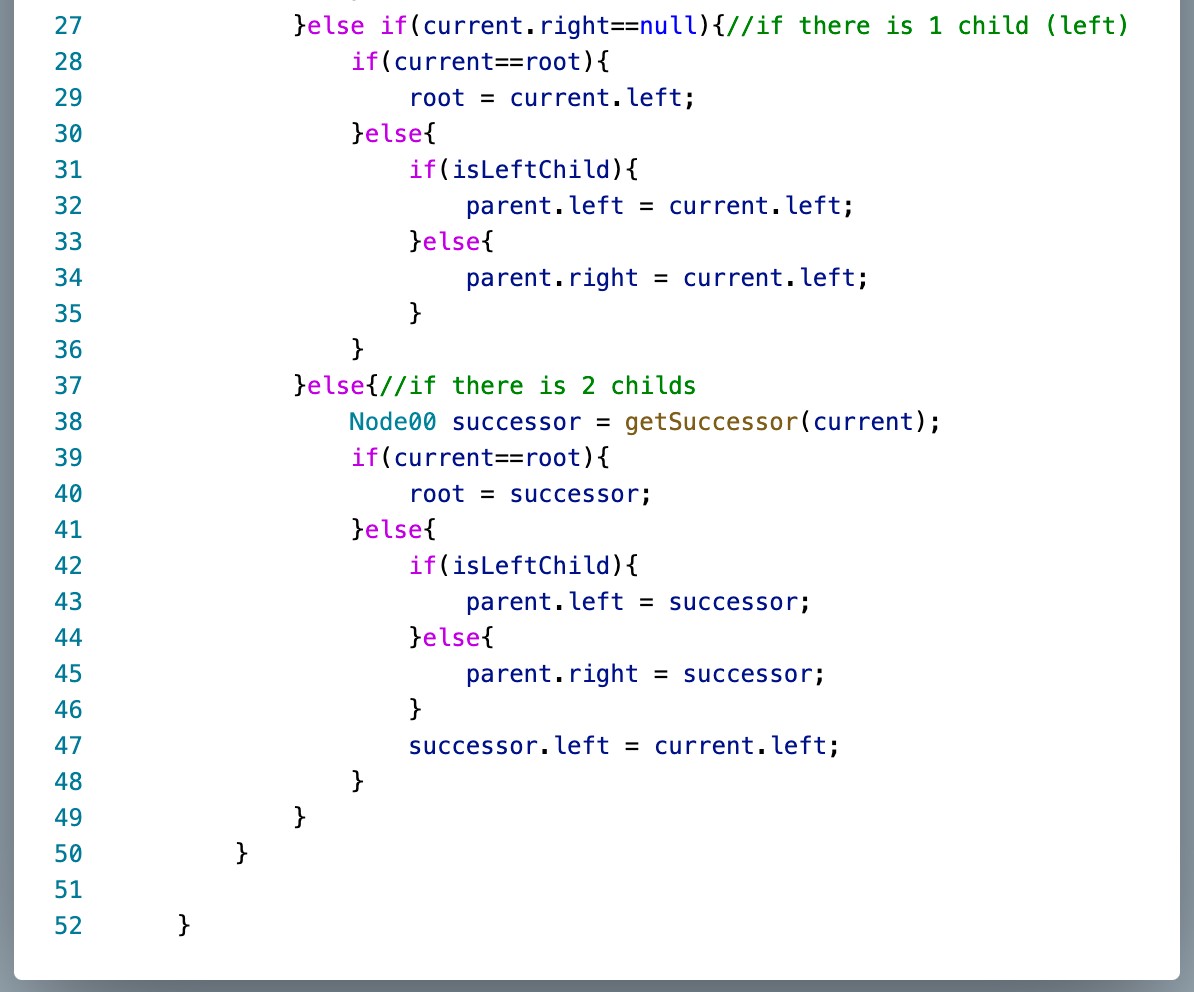
Di dalam method delete tambahkan pengecekan apakah tree kosong, dan jika tidak cari posisi node yang akan di hapus.





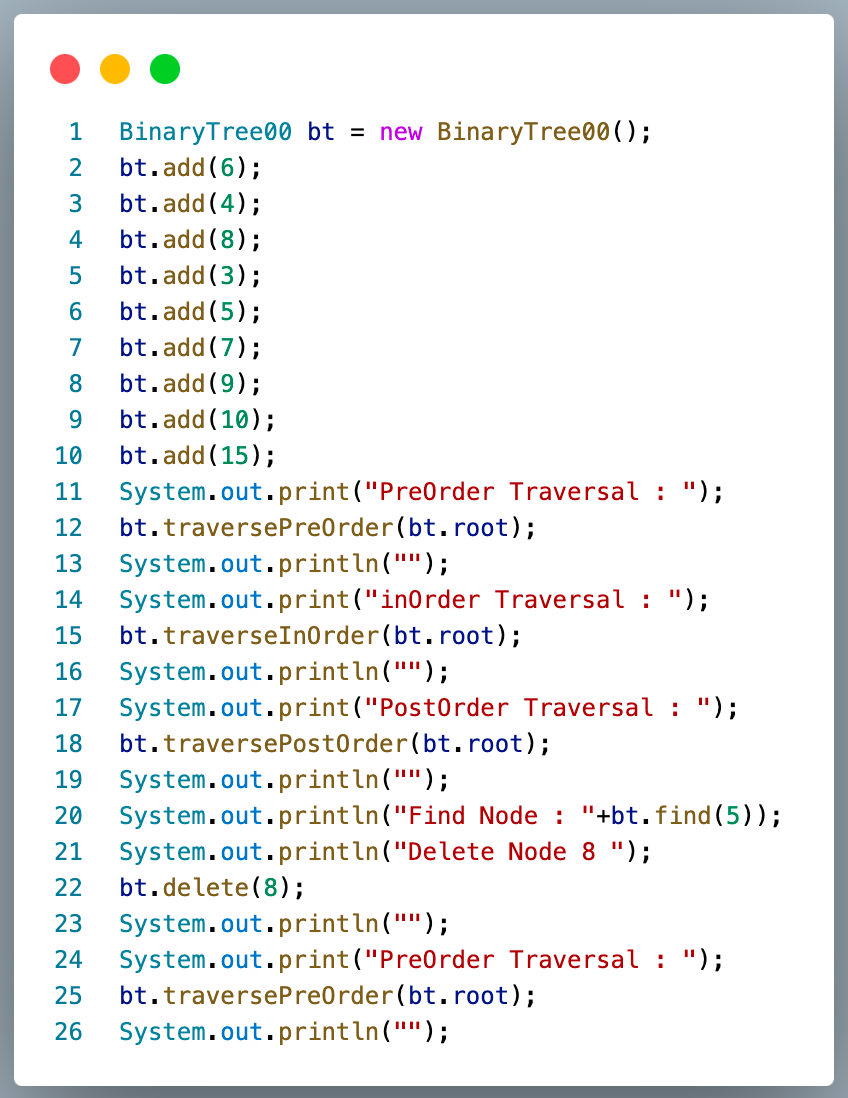
1. Kemudian tambahkan proses penghapusan didalam method **delete()** terhadap node current yang telah ditemukan.





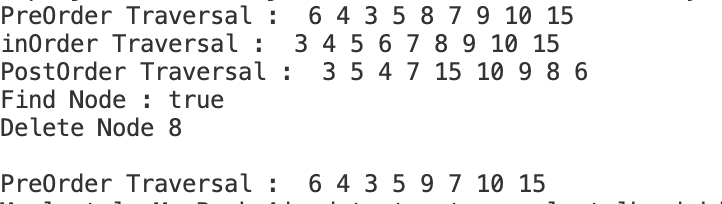


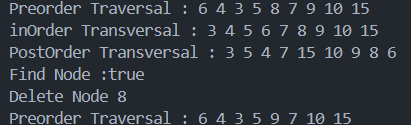
1. Buka class **BinaryTreeMainNoAbsen** dan tambahkan method main() kemudian tambahkan kode berikut ini





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





## Pertanyaan Percobaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?

Dalam binary search tree (BST), setiap node memiliki sifat bahwa semua nilai di subtree kiri lebih kecil dan semua nilai di subtree kanan lebih besar dari nilai node tersebut. Ini memungkinkan pencarian dilakukan secara efisien dengan membandingkan nilai yang dicari dengan nilai node saat ini dan mengarahkan pencarian ke subtree yang sesuai

1. Untuk apakah di class **Node**, kegunaan dari atribut **left** dan **right**?

Atribut left dan right pada class Node digunakan untuk menunjukkan child node dari suatu node tertentu dalam struktur data pohon

1. a. Untuk apakah kegunaan dari atribut **root** di dalam class **BinaryTree**?

Atribut root dalam class BinaryTree digunakan untuk menyimpan referensi ke node pertama (atau root) dari pohon biner. Ini memungkinkan akses langsung ke seluruh struktur pohon.

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

Ketika objek tree pertama kali dibuat, nilai dari root adalah null karena pada awalnya pohon belum memiliki node apapun.

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

Ketika tree masih kosong dan akan ditambahkan sebuah node baru, node tersebut akan menjadi root dari pohon tersebut. root akan menunjuk langsung ke node baru tersebut, dan subtree left dan right dari root akan diatur menjadi null

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;**

## }

**}**

Baris program ini bertanggung jawab untuk menambahkan node baru ke subtree kiri dari node current jika nilai data yang akan ditambahkan lebih kecil dari nilai data dari current.

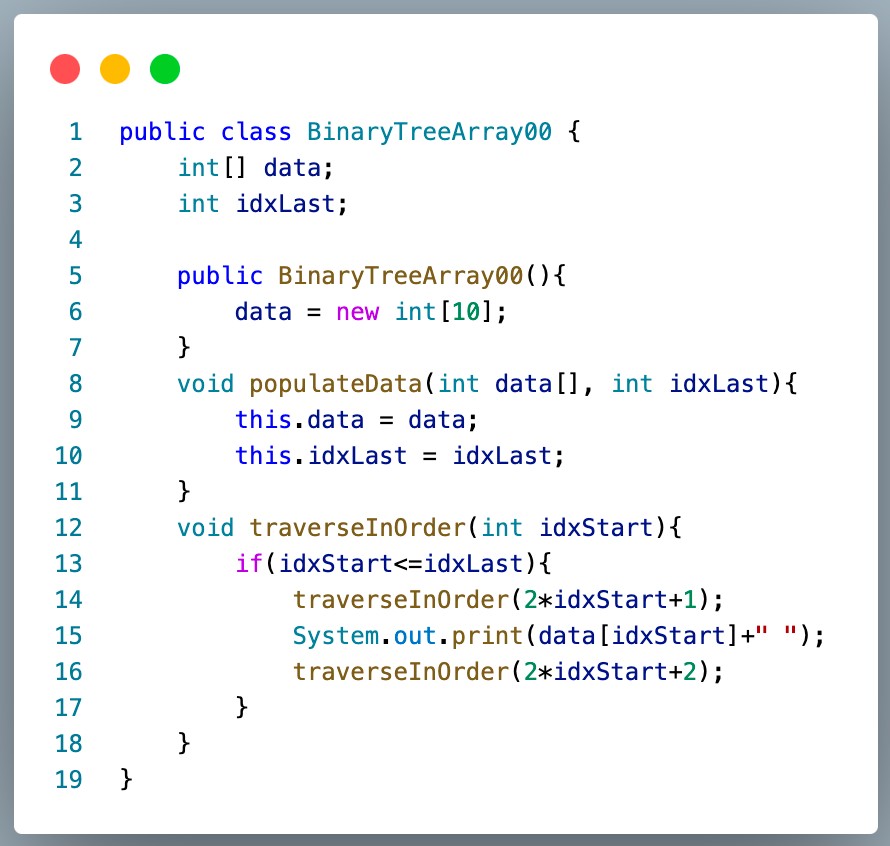
# Kegiatan Praktikum 2

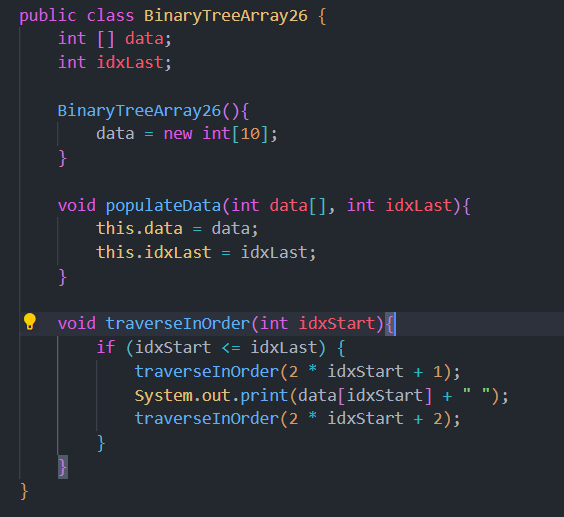
**Implementasi binary tree dengan array (45 Menit)**

## Tahapan Percobaan

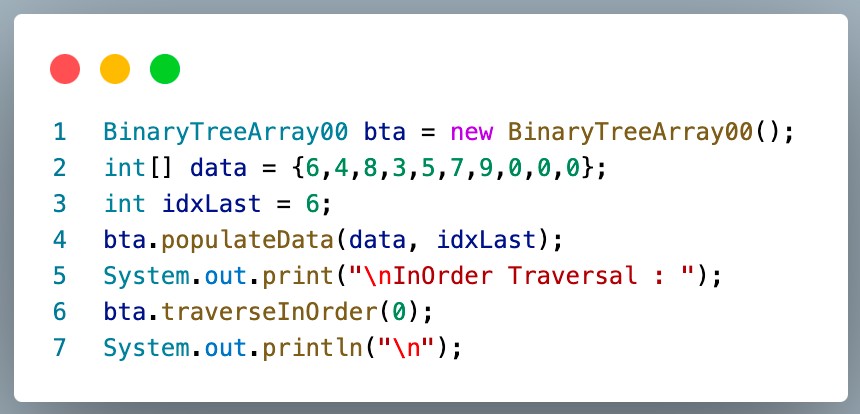
1. Di dalam percobaan implementasi binary tree dengan array ini, data tree disimpan dalam array dan langsung dimasukan dari method main(), dan selanjutnya akan disimulasikan proses traversal secara inOrder.
2. Buatlah class **BinaryTreeArrayNoAbsen** dan **BinaryTreeArrayMainNoAbsen**
3. Buat atribut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

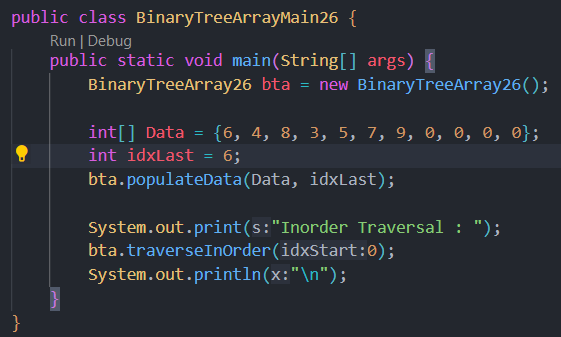
## populateData() dan traverseInOrder().





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!





## Pertanyaan Percobaan

* + - 1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray**?

data digunakan untuk menyimpan elemen-elemen dari binary tree dalam bentuk array.

idxLast menyimpan indeks terakhir dari data yang valid dalam array data.

* + - 1. Apakah kegunaan dari method **populateData()**?

Method populateData() digunakan untuk mengisi array data dengan data yang diberikan dan mengatur idxLast untuk menunjukkan indeks terakhir data yang valid dalam array.

* + - 1. Apakah kegunaan dari method **traverseInOrder()**?

Method traverseInOrder() digunakan untuk melakukan traversal inorder pada binary tree yang direpresentasikan dalam bentuk array. Ini mencetak elemen-elemen dalam urutan terurut dari kiri ke kanan.

* + - 1. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masin-masing?

Jika node binary tree disimpan dalam array di indeks i, maka left child berada di indeks 2\*i + 1 dan right child berada di indeks 2\*i + 2.

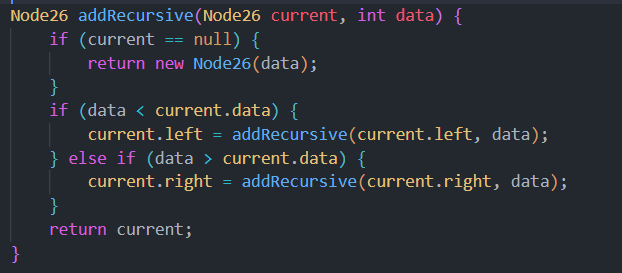
* + - 1. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?

Statement int idxLast = 6 mengatur idxLast untuk menunjukkan bahwa elemen terakhir dari array Data yang valid adalah di indeks 6. Hal ini memastikan bahwa traversal hanya dilakukan pada data yang valid dalam array.

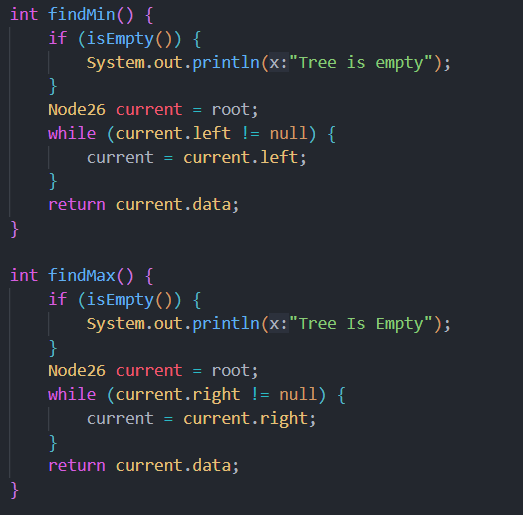
# Tugas Praktikum

## Waktu pengerjaan: 90 menit

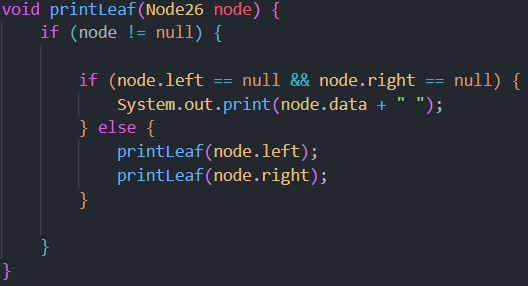
* + 1. Buat method di dalam class **BinaryTree** yang akan menambahkan node dengan cara rekursif.



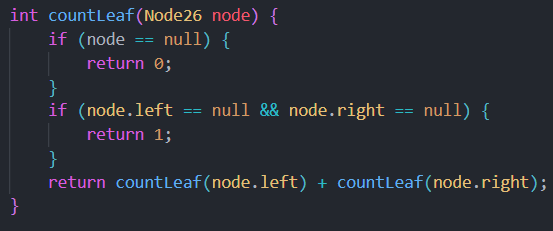
* + 1. Buat method di dalam class **BinaryTree** untuk menampilkan nilai paling kecil dan yang paling besar yang ada di dalam tree.



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



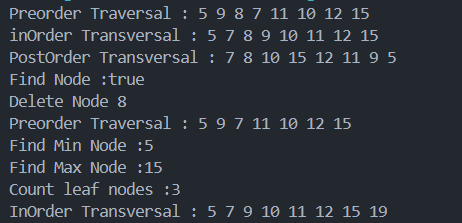
* + 1. Buat method di dalam class **BinaryTree** untuk menampilkan berapa jumlah leaf yang ada di dalam tree.



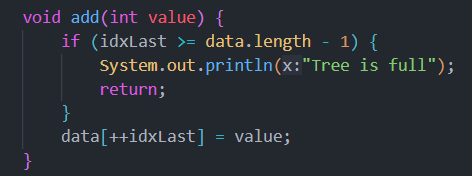
Binary tree main :



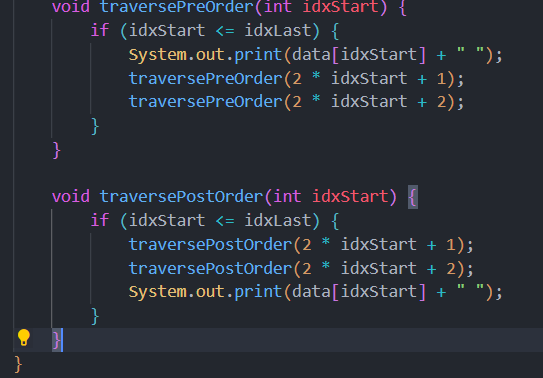
Hasil :



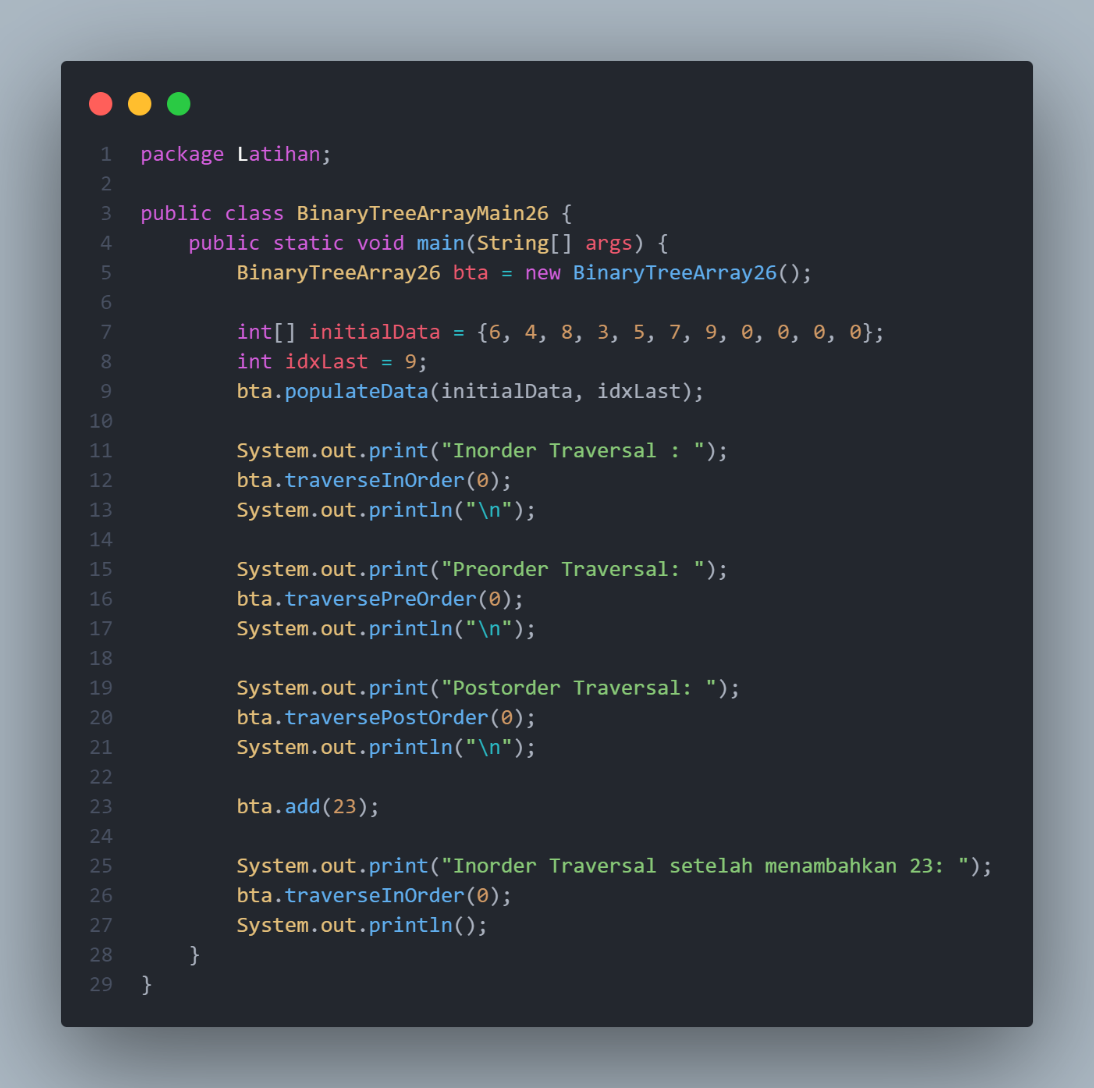
* + 1. Modifikasi class **BinaryTreeArray**, dan tambahkan :
       - method **add(int data)** untuk memasukan data ke dalam tree



* + - * method **traversePreOrder()** dan **traversePostOrder()**



**BinaryTreeArrayMain:**



**Hasil:**

