**LAPORAN PRAKTIKUM**

**Politeknik Negeri Malang**

**Praktikan**

**ALGORITMA DAN STRUKTUR DATA**

****

2022

2141720183

RIDWAN CAESAR RIZQI KARISMA BIWARNI

TI 1C

Daftar Isi

[Percobaan implementasi Binary Search Tree menggunakan Linked List 4](#_Toc105059761)

[Langkah 1 4](#_Toc105059762)

[Langkah 2 5](#_Toc105059763)

[Langkah 3 5](#_Toc105059764)

[Langkah 4 5](#_Toc105059765)

[Langkah 5 6](#_Toc105059766)

[Langkah 6 6](#_Toc105059767)

[Langkah 7 7](#_Toc105059768)

[Langkah 8 8](#_Toc105059769)

[Langkah 9 8](#_Toc105059770)

[Langkah 10 10](#_Toc105059771)

[Langkah 11 10](#_Toc105059772)

[Langkah 12 10](#_Toc105059773)

[Pertanyaan Percobaan 11](#_Toc105059774)

[Jawaban Percobaan 11](#_Toc105059775)

[Percobaan implementasi binary tree dengan array 12](#_Toc105059776)

[Langkah 1 12](#_Toc105059777)

[Langkah 2 12](#_Toc105059778)

[Langkah 3 12](#_Toc105059779)

[Langkah 4 13](#_Toc105059780)

[Langkah 5 13](#_Toc105059781)

[Pertanyaan Percobaan 13](#_Toc105059782)

[Jawaban Percobaan 13](#_Toc105059783)

[Tugas Praktikum 14](#_Toc105059784)

[Jawaban Nomor 1 14](#_Toc105059785)

[Menambahkan method add node dengan cara rekursif 14](#_Toc105059786)

[Modifikasi pada class BinaryTreeMain 15](#_Toc105059787)

[Output 15](#_Toc105059788)

[Jawaban Nomor 2 16](#_Toc105059789)

[Modifikasi pada class BinaryTree 16](#_Toc105059790)

[Modifikasi pada class BinaryTreeMain 16](#_Toc105059791)

[Output 17](#_Toc105059792)

[Jawaban Nomor 3 17](#_Toc105059793)

[Menambahkan method pada class BinaryTree 17](#_Toc105059794)

[Modifikasi pada class BinaryTreeMain 17](#_Toc105059795)

[Output 17](#_Toc105059796)

[Jawaban Nomor 4 18](#_Toc105059797)

[Menambahkan method untuk menghitung leaf pada class BinaryTree 18](#_Toc105059798)

[Modifikasi pada class BinaryTreeMain 18](#_Toc105059799)

[Output 18](#_Toc105059800)

[Jawaban Nomor 5 19](#_Toc105059801)

[Menambahkan method add 19](#_Toc105059802)

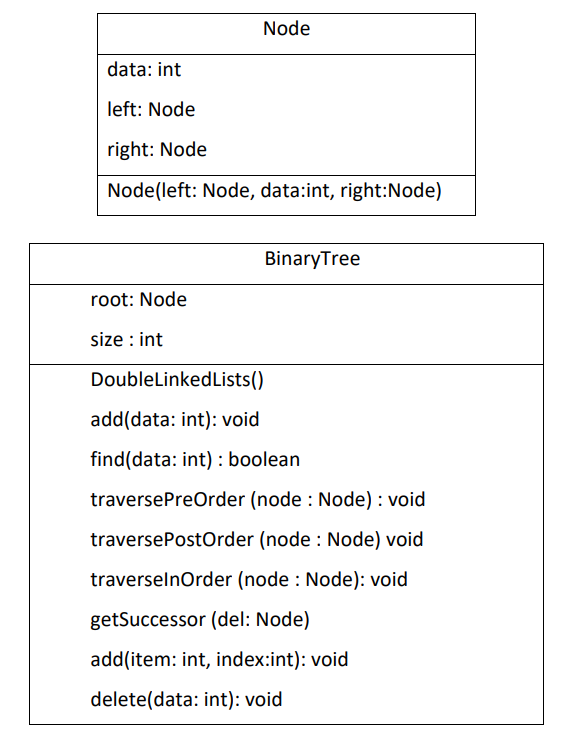
[Menambahkan method traversePreOrder() dan traversePostOrder() 19](#_Toc105059803)

[Modifikasi pada class BinaryTreeArrayMain 20](#_Toc105059804)

[Output 20](#_Toc105059805)

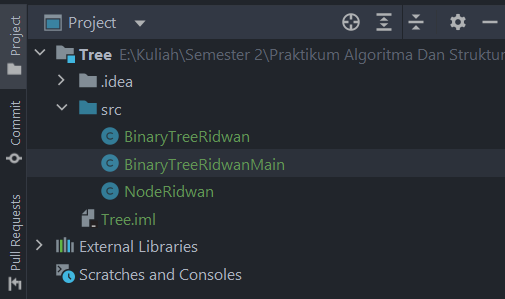
# Percobaan implementasi Binary Search Tree menggunakan Linked List

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



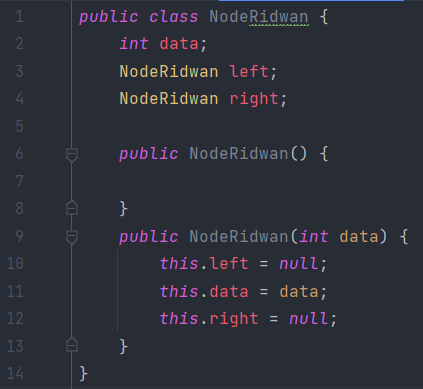
## Langkah 1

Buatlah class Node, BinaryTree dan BinaryTreeMain



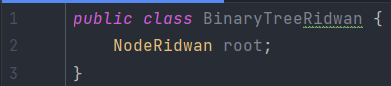
## Langkah 2

Di dalam class Node, tambahkan atribut data, left dan right, serta konstruktor default dan berparameter.



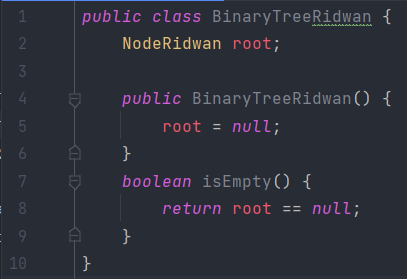
## Langkah 3

Di dalam class BinaryTree, tambahkan atribut root.



## Langkah 4

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



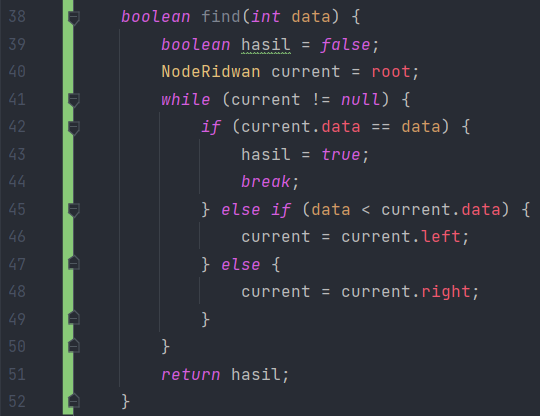
## Langkah 5

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.



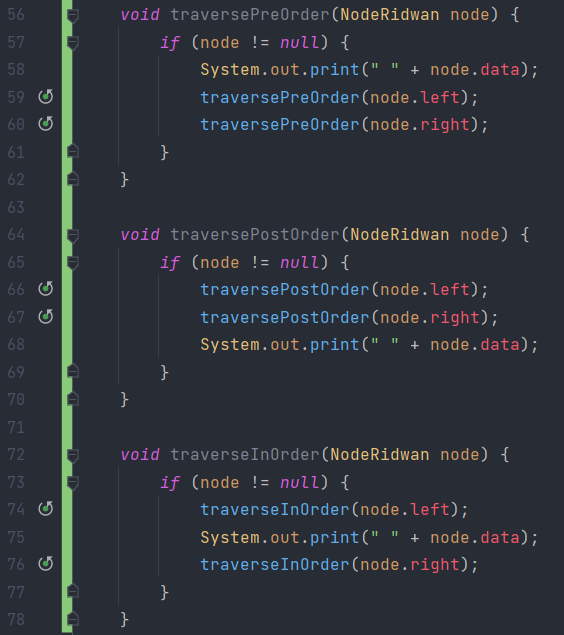
## Langkah 6

Tambahkan method find()



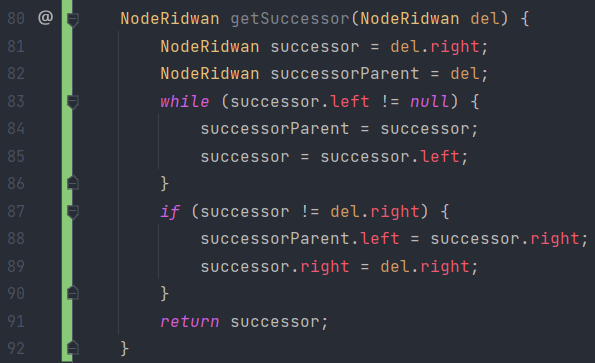
## Langkah 7

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.



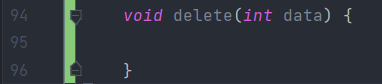
## Langkah 8

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



## Langkah 9

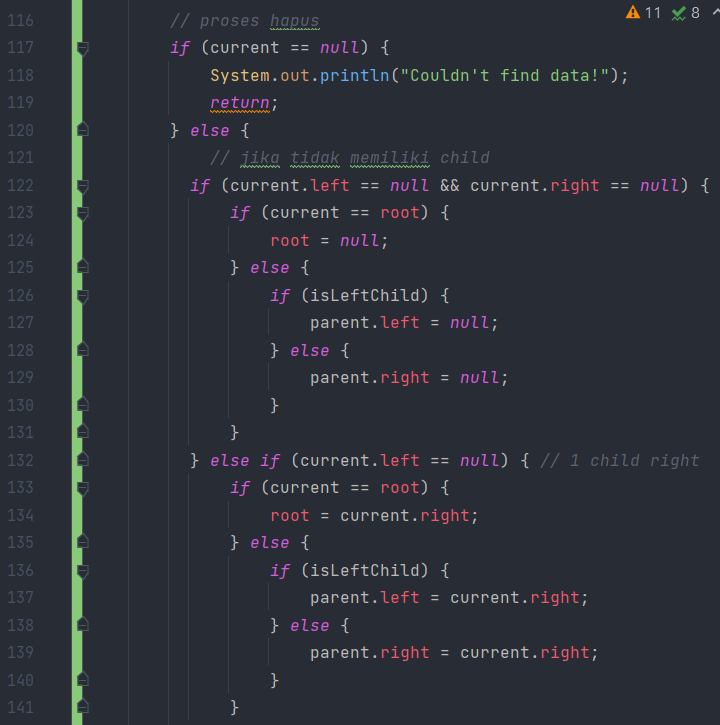
Tambahkan method delete().

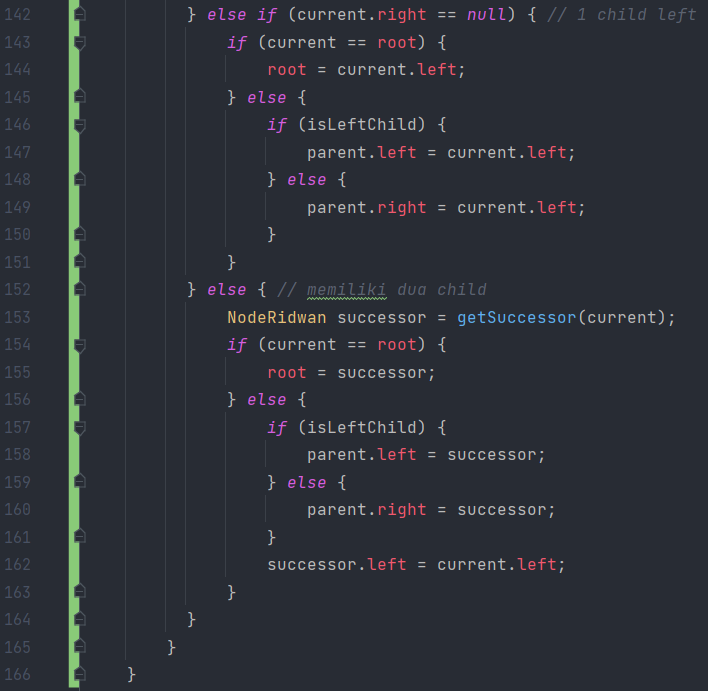


Di dalam method delete tambahkan pengecekan apakah tree kosong, dan jika tidak cari posisi node yang akan di hapus.



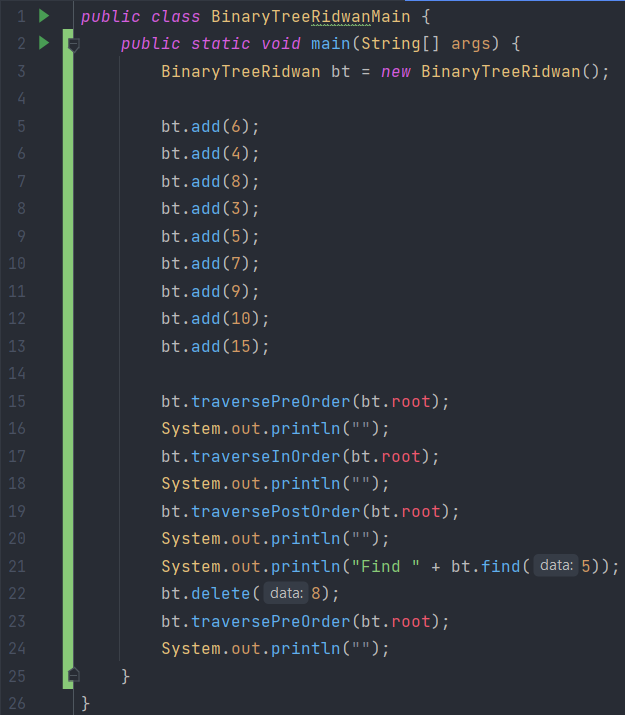
Kemudian tambahkan proses penghapusan terhadap node current yang telah ditemukan.





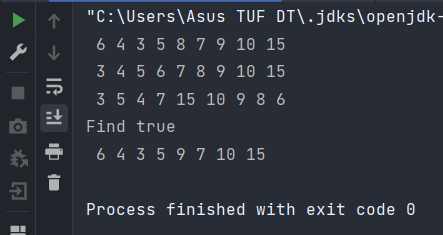
## Langkah 10

Buka class BinaryTreeMain dan tambahkan method main().



## Langkah 11

Compile dan jalankan class BinaryTreeMain untuk mendapatkan simulasi jalannya program tree yang telah dibuat.



## Langkah 12

Amati hasil running tersebut.

### 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 NodeRidwan(data);**

**break;**

**}**

**}**

### Jawaban Percobaan

1. Karena dalam binary search tree seluruh child dari tiap node sudah dalam keadaan terurut sehingga pencarian data lebih efektif.
2. Atribut left digunakan sebagai pointer yang akan menunjuk ke child disebelah kiri sedangkan atribut right digunakan sebagai pointer yang akan menunjuk ke child disebelah kanan.
3. a. Root digunakan sebagai data yang pertama kali dimasukkan ke dalam tree dan akan dimanfaatkan dalam operasi tree.

b. Nilai dari root ketika tree pertama kali dibuat adalah null.

1. Proses yang akan terjadi adalah root akan di isi dengan node baru.
2. Pada baris kode tersebut akan dilakukan pengecekan apakah data yang akan ditambahkan bernilai kurang dari nilai data current. Kemudian akan dilakukan pengecekan lagi jika nilai child kiri dari current bukan null maka current akan bernilai child sebelah kiri tadi, sedangkan jika tidak maka child di sebelah kiri akan di isi node baru dan akan keluar dari perulangan.

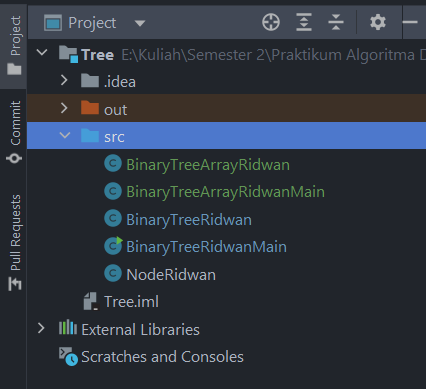
# Percobaan implementasi binary tree dengan array

## Langkah 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.

## Langkah 2

Buatlah class BinaryTreeArray dan BinaryTreeArrayMain



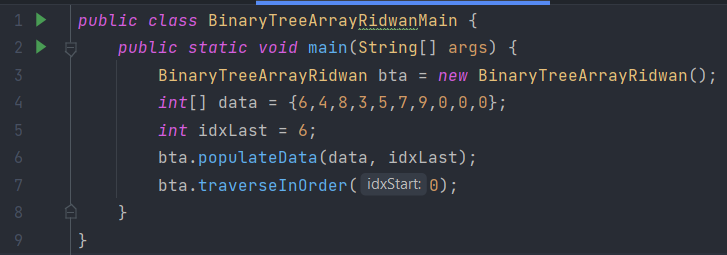
## Langkah 3

Buat atribut data dan idxLast di dalam class BinaryTreeArray. Buat juga method populateData() dan traverseInOrder().



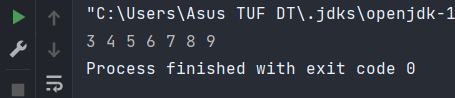
## Langkah 4

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



## Langkah 5

Jalankan class BinaryTreeArrayMain dan amati hasilnya!



### 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?

### Jawaban Percobaan

1. Atribut data akan digunakan untuk menyimpan data dalam array sedangkan idxlast adalah index terakhir dalam array yang sudah di isi pada kasus ini data yang bukan 0.
2. Method tersebut digunakan untuk mengisi variabel data dan idxLast pada class BinaryTreeArray sehingga dapat dilakukan operasi seperti traverse.
3. Method tersebut digunakan untuk mencetak isi dari Tree dengan aturan InOrder.
4. Jika sebuah data berada pada indeks ke 2 maka left child akan berada pada indeks ke 5 dan right child berada pada indeks ke 6.
5. idxLast berfungsi untuk menentukan indeks terakhir pada array yang sudah berisi data yang sesuai dalam kasus ini data yang bukan 0.

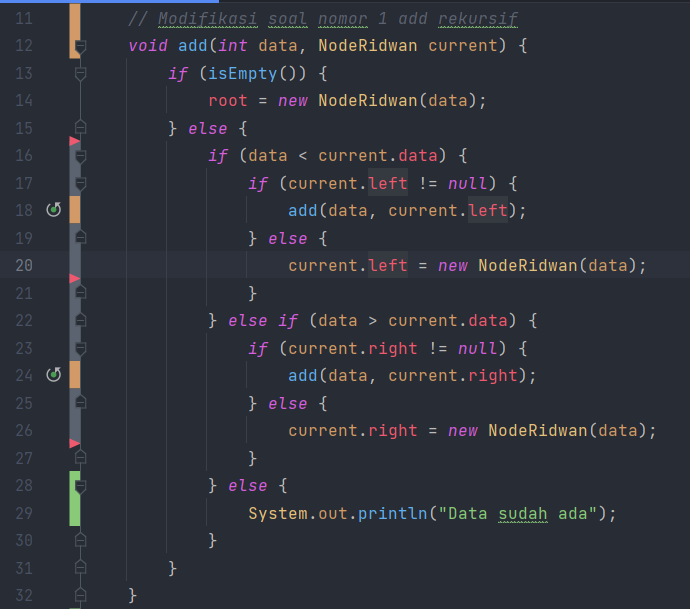
# Tugas Praktikum

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 :

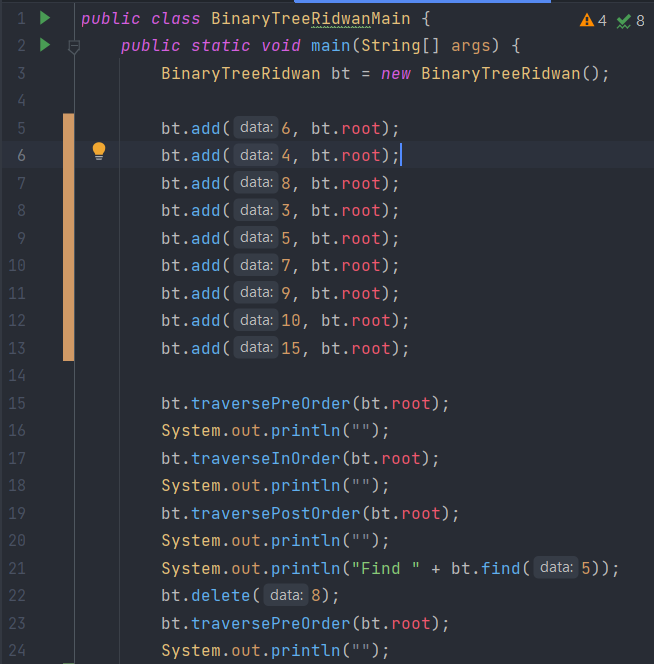
* method add(int data) untuk memasukan data ke dalam tree
* method traversePreOrder() dan traversePostOrder()

## Jawaban Nomor 1

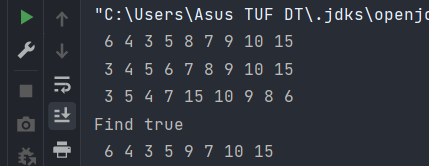
### Menambahkan method add node dengan cara rekursif



### Modifikasi pada class BinaryTreeMain

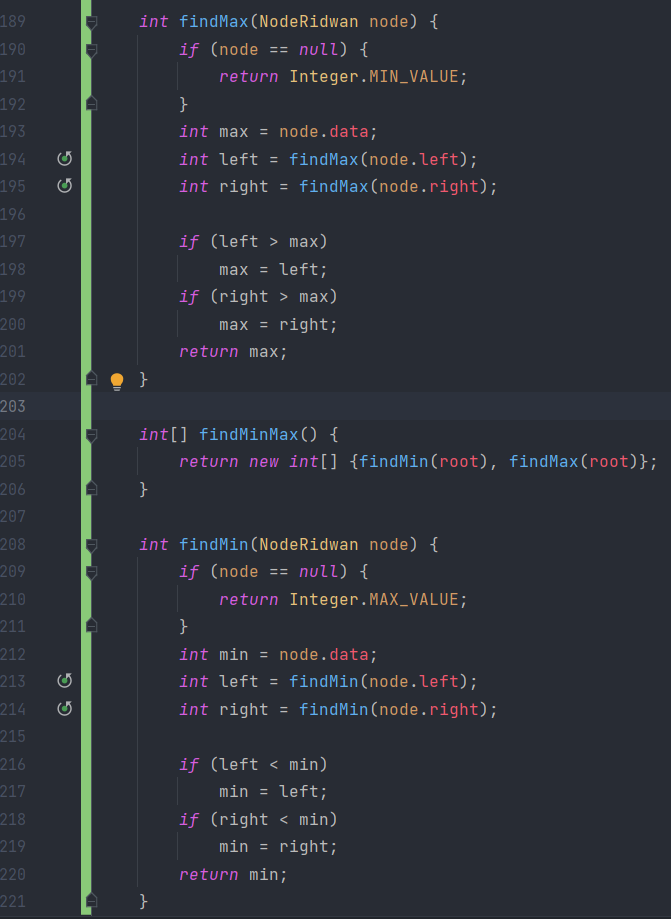


### Output

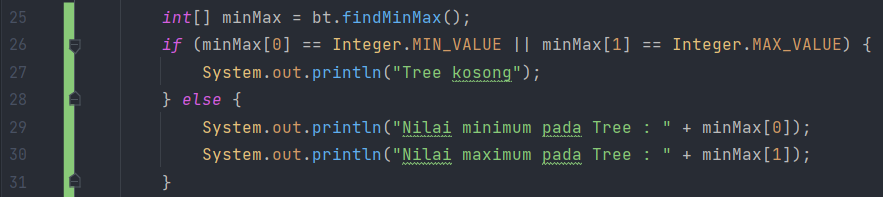


## Jawaban Nomor 2

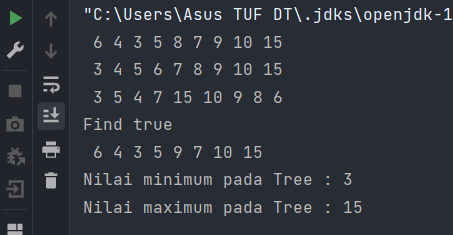
### Modifikasi pada class BinaryTree



### Modifikasi pada class BinaryTreeMain

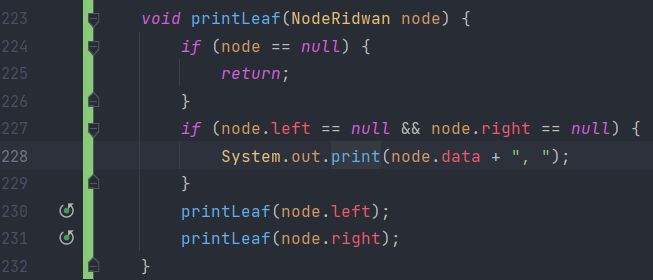


### Output



## Jawaban Nomor 3

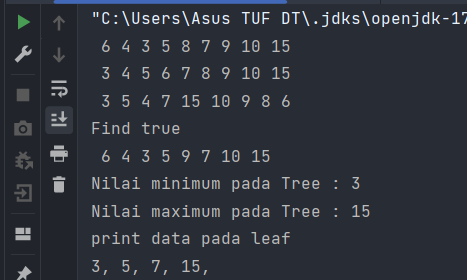
### Menambahkan method pada class BinaryTree



### Modifikasi pada class BinaryTreeMain

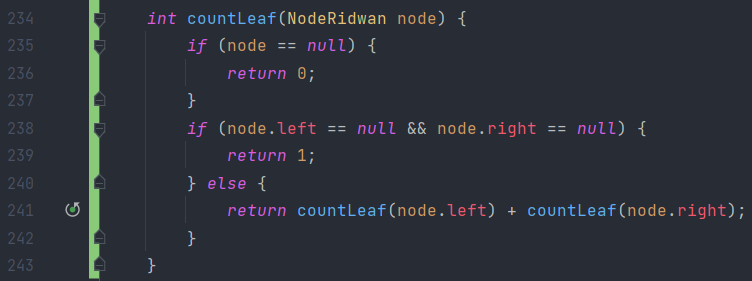


### Output



## Jawaban Nomor 4

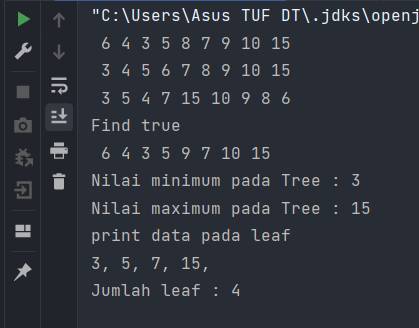
### Menambahkan method untuk menghitung leaf pada class BinaryTree



### Modifikasi pada class BinaryTreeMain



### Output

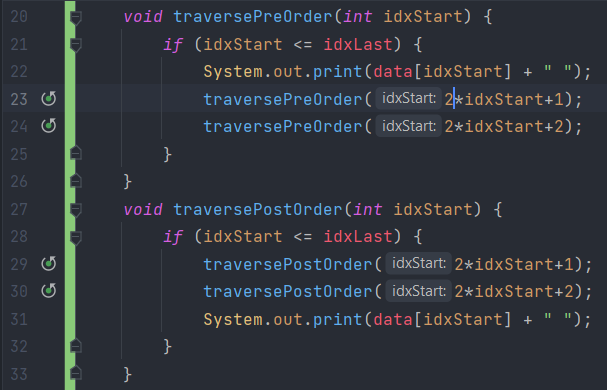


## Jawaban Nomor 5

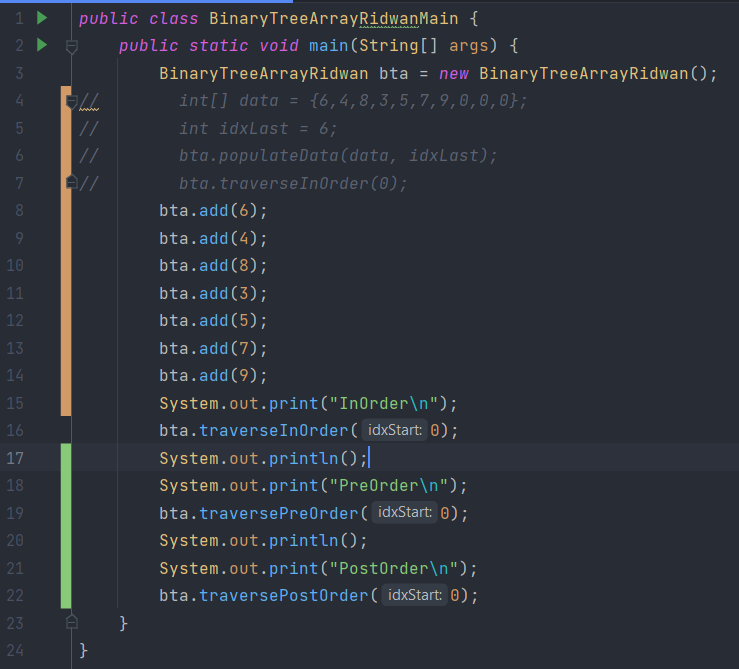
### Menambahkan method add



### Menambahkan method traversePreOrder() dan traversePostOrder()



### Modifikasi pada class BinaryTreeArrayMain



### Output

