Nama : Muhammad Afrizal
Kelas : 47-02
NIM : 607062330011

A. Penjelasan Koding

Penjelasan BinaryTree2.java :

Kelas `BinaryTree2` adalah struktur data pohon biner yang memiliki fungsi-fungsi untuk menambahkan elemen, mencari elemen, dan menghitung jumlah elemen. Ketika menambahkan elemen baru, metode yang digunakan akan mencari secara rekursif tempat yang tepat berdasarkan ketersediaan posisi anak kiri atau kanan. Untuk mencari elemen, proses serupa dilakukan di mana setiap elemen simpul dibandingkan dengan elemen yang dicari. Kelas ini juga dilengkapi dengan fungsi untuk menghitung total elemen dan melakukan penelusuran atau traversal (yaitu, inorder, preorder, dan postorder), yang masing-masing memberikan tampilan berbeda dari struktur data.

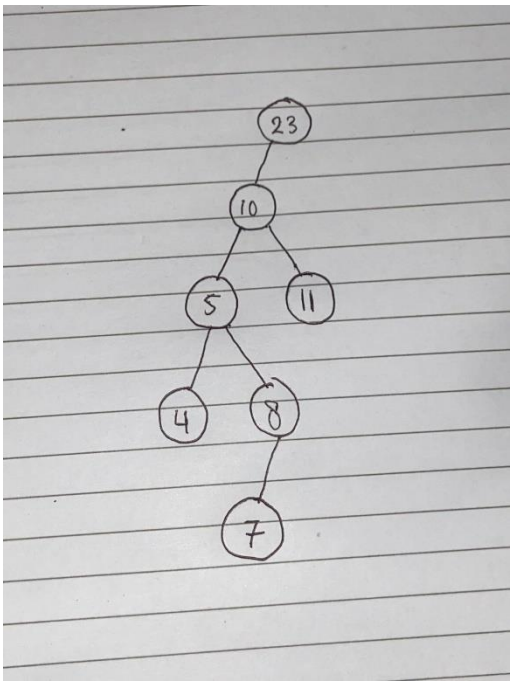
Penjelasan Node2.java :

`Node2` mendefinisikan simpul dalam pohon biner, menyimpan data dan referensi ke simpul anak kiri dan kanan. Kelas ini menyediakan konstruktor untuk inisialisasi dengan atau tanpa data awal dan termasuk metode untuk mengatur dan mendapatkan informasi dari simpul—baik itu data atau referensi ke simpul lain. Ini memungkinkan setiap simpul berfungsi sebagai blok bangunan untuk struktur pohon, membantu dalam menyusun data secara terstruktur dan hierarkis.

Penjelasan Jurnal11.java :

Kelas `Jurnal11` berfungsi sebagai titik interaksi pengguna dengan pohon biner. Di sini, pengguna bisa memilih berbagai operasi seperti memasukkan banyak elemen sekaligus, mencari elemen tertentu, atau menghitung jumlah total elemen dalam pohon. Sistem input memungkinkan pengguna memasukkan beberapa angka dalam satu baris, yang kemudian dipisahkan dan dimasukkan satu per satu ke dalam pohon. Aplikasi ini menjalankan perulangan yang memungkinkan pengguna terus memilih tindakan hingga mereka memutuskan untuk berhenti. Dengan struktur switch-case, aplikasi menangani pilihan pengguna dan memberikan umpan balik yang jelas dan informatif setelah setiap operasi dilakukan.

B. Gambar dari pohon yang terbentuk sesuai dengan masukan pada program



C. Penjelasan mengenai mengapa pohon yang terbentuk seperti itu, juga penjelasan mengenai traversal preorder, inorder, dan postorder.

Penjelasan pohon :

- 23 dijadikan sebagai root karena pohon awalnya kosong.
- 10 lebih kecil dari 23, jadi disisipkan ke kiri dari 23.
- 5 lebih kecil dari 23 dan lebih kecil dari 10, jadi disisipkan ke kiri dari 10.
- 8 lebih kecil dari 23, lebih kecil dari 10, tetapi lebih besar dari 5, jadi disisipkan ke kanan dari 5.
- 11 lebih kecil dari 23 tetapi lebih besar dari 10, jadi disisipkan ke kanan dari 10.
- 4 lebih kecil dari semua simpul sebelumnya (23, 10, 5), jadi disisipkan ke kiri dari 5.
- 7 lebih kecil dari 23 dan 10, tetapi lebih besar dari 5. Oleh karena 5 sudah memiliki anak di kanan (8), maka 7 akan disisipkan ke kiri dari 8.

Penjelasan mengenai traversal preorder, inorder, dan postorder :

Preorder

Traversal preorder mengikuti prinsip "akar dulu", yang artinya kita pertama-tama mengunjungi dan memproses simpul akar, kemudian secara rekursif melakukan traversal preorder pada anak

kiri, dan setelah itu pada anak kanan. Dengan kata lain, setiap simpul diproses sebelum kedua anaknya diproses. Urutan proses untuk pohon di atas adalah: 23, 10, 5, 4, 8, 7, 11.

Inorder

Traversal inorder pada pohon biner khususnya berguna karena memberikan elemen-elemen dalam urutan yang terurut jika pohon biner tersebut adalah pohon biner pencarian. Dalam traversal ini, kita pertama-tama mengunjungi anak kiri secara rekursif, kemudian memproses simpul akar, dan terakhir mengunjungi anak kanan secara rekursif. Hasilnya adalah elemen-elemen akan dikunjungi dalam urutan menaik. Urutan proses untuk pohon di atas adalah: 4, 5, 7, 8, 10, 11, 23.

Postorder

Dalam traversal postorder, simpul akar diproses setelah kedua anaknya telah diproses. Ini berarti kita pertama kali mengunjungi secara rekursif anak kiri, kemudian anak kanan, dan baru setelah itu simpul akar. Traversal ini sering digunakan untuk operasi yang perlu memastikan bahwa hasil dari anak-anak telah tersedia sebelum simpul induk diproses, seperti dalam menghitung total atau menghapus seluruh pohon. Urutan proses untuk pohon di atas adalah: 4, 7, 8, 5, 11, 10, 23.