LAPORAN PRAKTIKUM

BINARY TREE



Oleh :

MUHAMMAD GALID AVERO

NIM 2311532008

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T

ASISTEN LABORATORIUM : ZAKY ADIL HAKIM

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG, JUNI 2024

1. PENDAHULUAN

Struktur data berbentuk pohon berbentuk biner di mana setiap node memiliki paling banyak dua anak: anak kiri (left child) dan anak kanan (right child). Struktur ini digunakan untuk berbagai jenis pemrosesan, seperti pengurutan, pencarian, dan ekspresi aritmatika. Dalam representasi binar pohon, kelas node digunakan. Setiap node dalam pohon menyimpan data dan referensi ke anak kiri dan kanan.   
Binary tree dapat melakukan operasi dasar seperti penambahan node, pencarian, dan penelusuran(traversing).

1. Penambahan Node

Node baru dapat ditambahkan dengan menentukan apakah node tersebut akan menjadi anak kiri atau anak kanan dari node induk.

1. Traversing (Penelusuran)

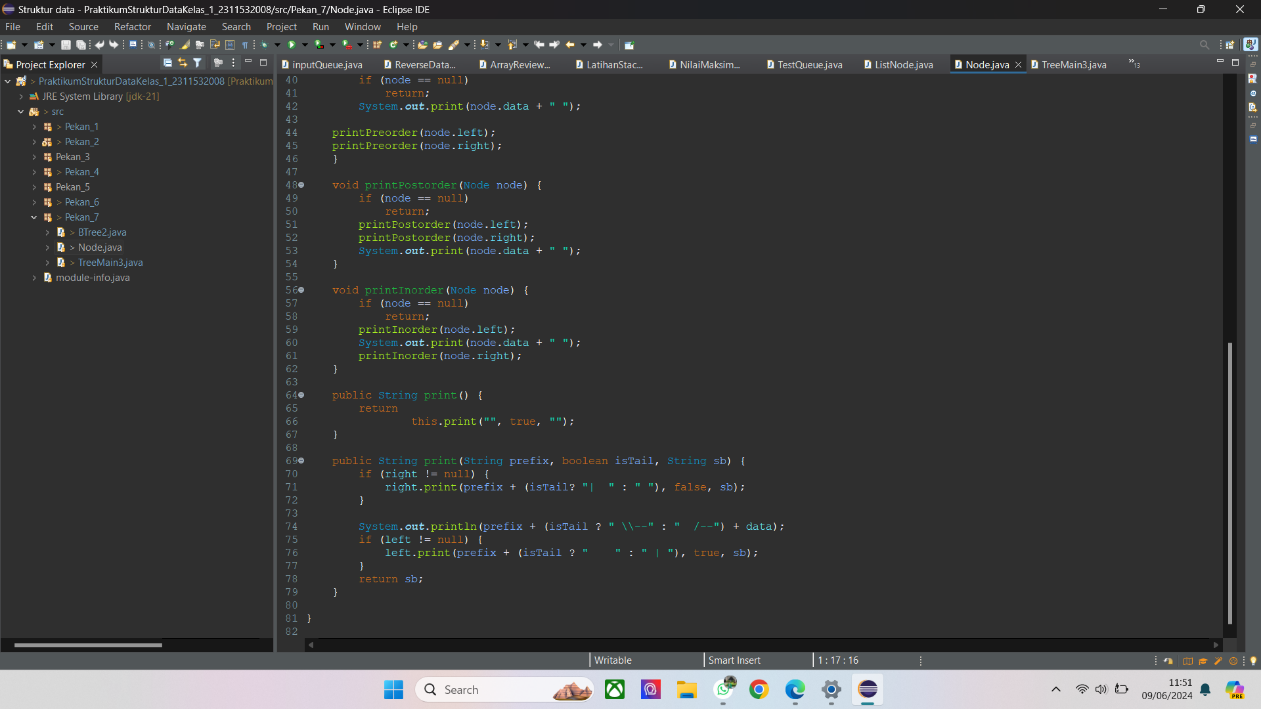
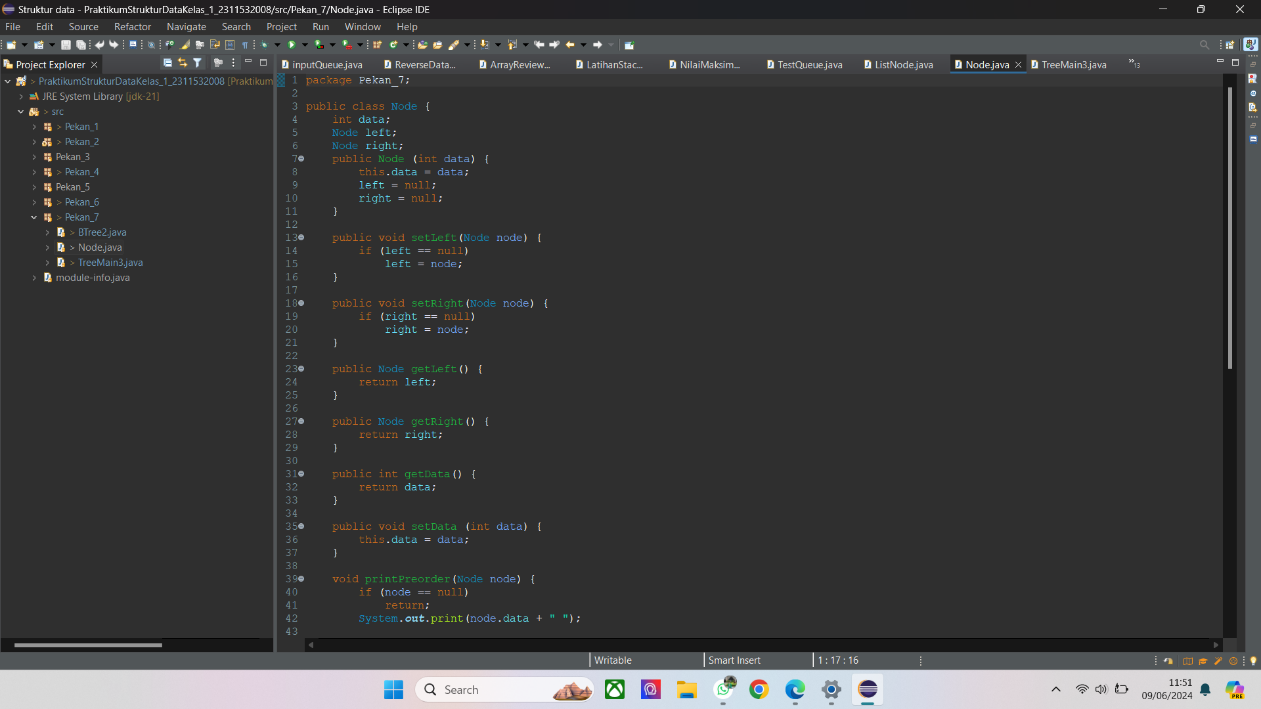
Penelusuran binary tree dapat dilakukan dengan tiga cara utama:

* Preorder Traversal: Mengunjungi node saat ini terlebih dahulu, kemudian anak kiri, dan terakhir anak kanan.
* Inorder Traversal: Mengunjungi anak kiri terlebih dahulu, kemudian node saat ini, dan terakhir anak kanan.
* Postorder Traversal: Mengunjungi anak kiri terlebih dahulu, kemudian anak kanan, dan terakhir node saat ini.

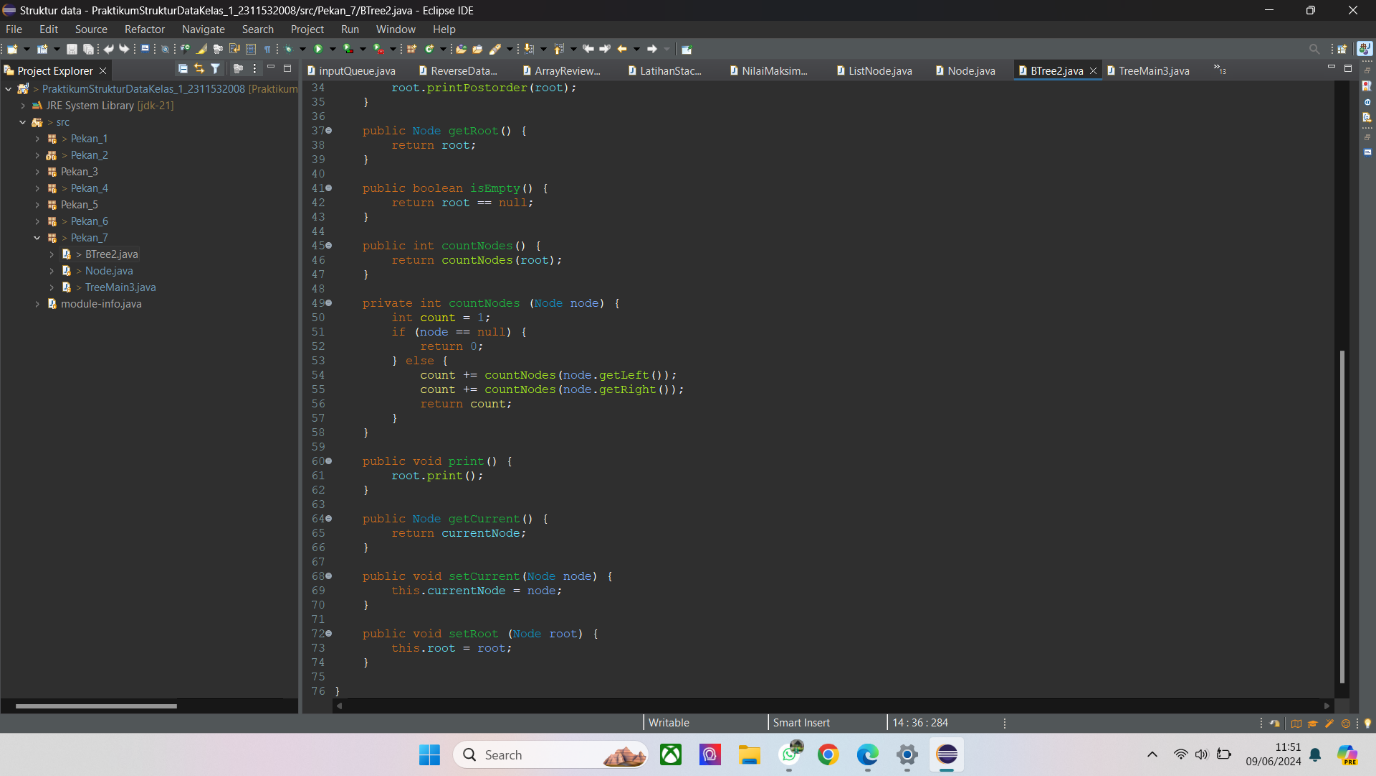
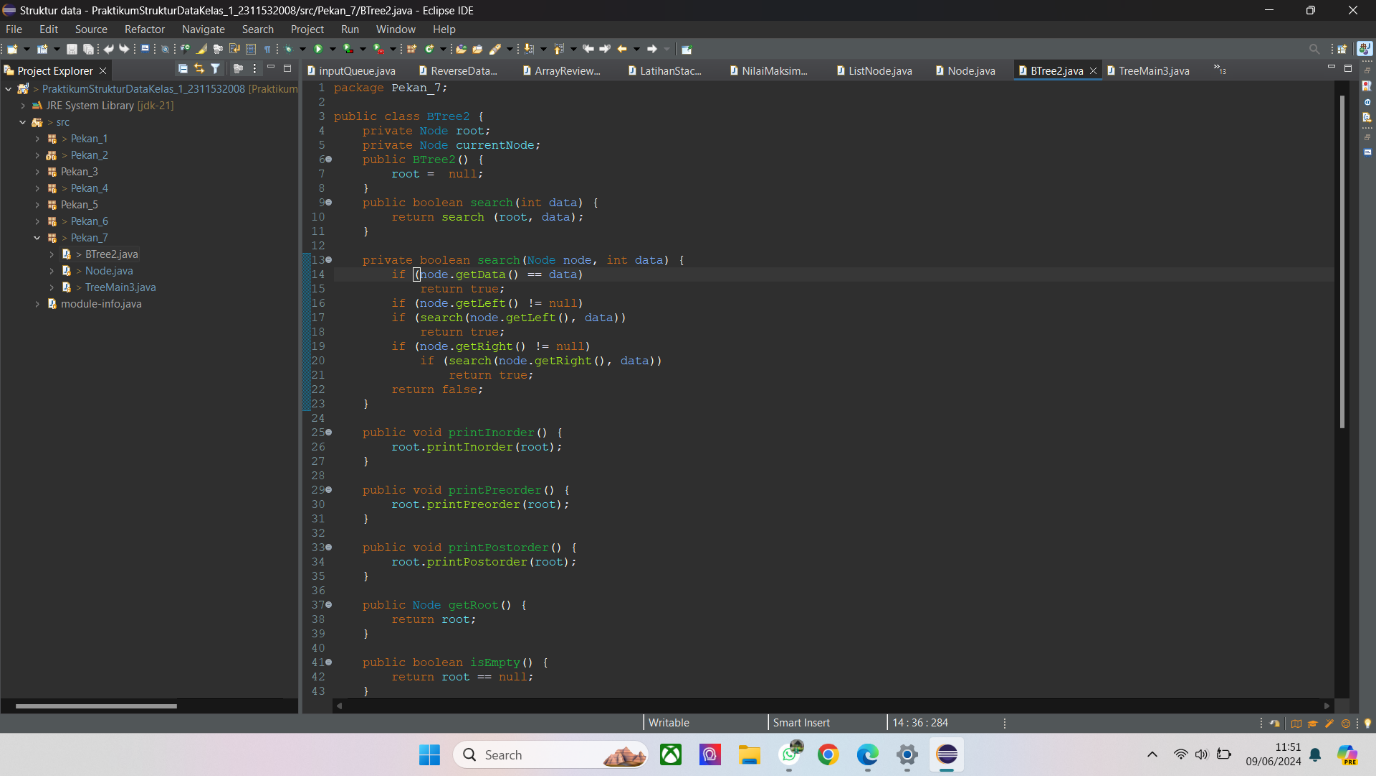
Kita dapat menggunakan teknik rekursif untuk menyusun string yang mewakili struktur pohon secara visual dan mencetak binary tree dalam format yang lebih mudah dibaca. Jenis Pohon Biner :

* Binary Search Tree (BST) : di mana setiap node memenuhi properti BST (anak kiri memiliki nilai yang lebih kecil dan anak kanan memiliki nilai yang lebih besar).
* Balanced Binary Tree : adalah pohon biner yang mempertahankan keseimbangan untuk memastikan bahwa operasi dasar dapat dilakukan dengan efisien dalam kompleksitas waktu.
* Complete Binary Tree : adalah pohon biner di mana semua level kecuali yang terakhir terisi penuh, dan semua node pada level terakhir berada di posisi paling kiri.

1. TUJUAN PRAKTIKUM
2. Memahami cara mendefiniskan kelas BTree untuk berinteraksi dengan pohon biner.
3. Memahami cara membuat kelas Node yang merepresentasi dari sebuah node dalam pohon biner.
4. Memahami cara membuat kelas main untuk menguji kelas yang sudah didefinisikan
5. LANGKAH PRAKTIKUM
6. Mendefinisikan kelas Node

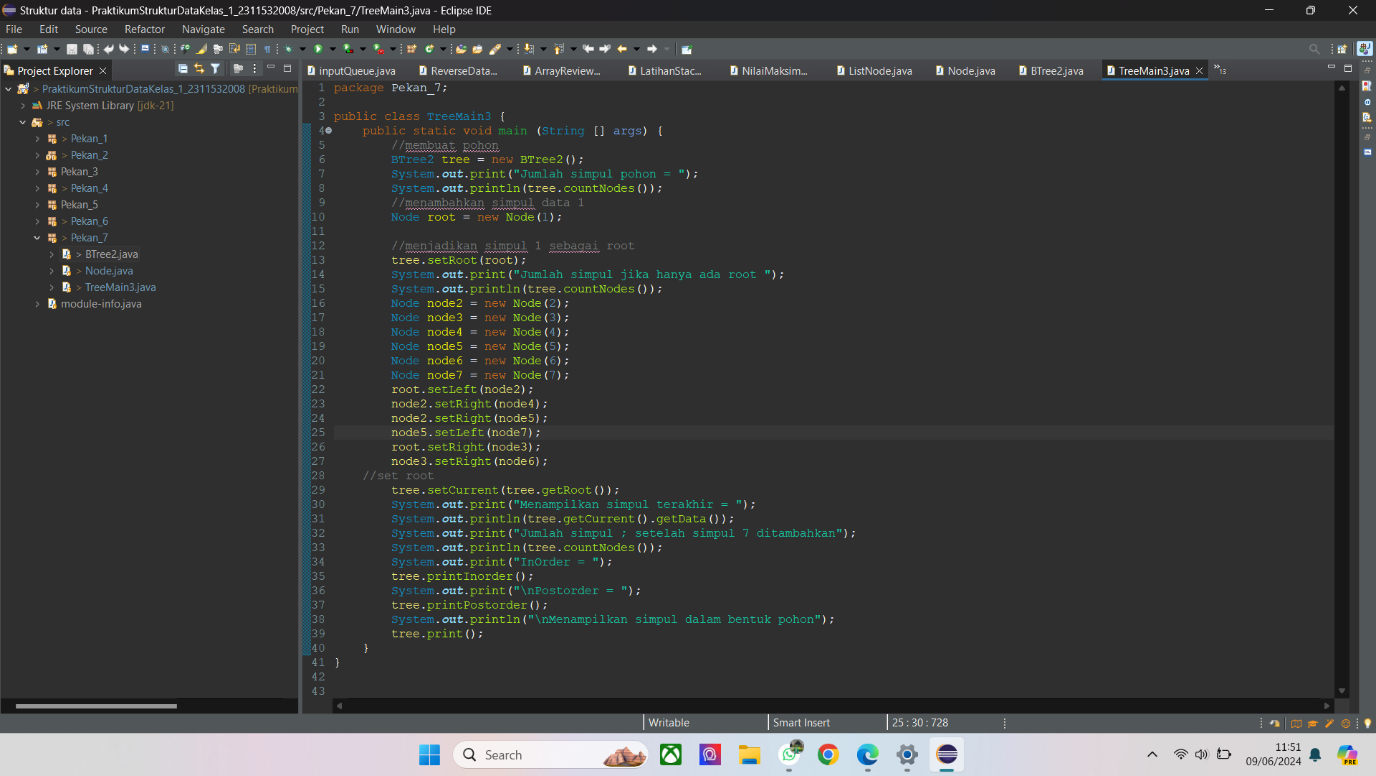


1. **Metode**print()**tanpa parameter**: Metode ini adalah metode publik yang digunakan untuk memulai proses pencetakan. Ia memanggil metode print(String prefix, boolean isTail, String sb) dengan parameter awal yang diperlukan untuk pencetakan.
2. **Metode**print(String prefix, boolean isTail, String sb): Metode ini adalah metode rekursif yang mencetak node pohon biner secara visual dengan format tertentu. Parameter yang digunakan adalah:
   * prefix: String yang digunakan untuk menambahkan spasi atau garis vertikal sebagai bagian dari format visualisasi.
   * isTail: Boolean yang menentukan apakah node saat ini adalah “ekor” dari cabangnya, yang mempengaruhi simbol yang digunakan dalam pencetakan.
   * sb: String yang seharusnya digunakan untuk mengumpulkan hasil cetak, tetapi dalam implementasi ini tidak digunakan.
3. Mendefinisikan kelas Btree



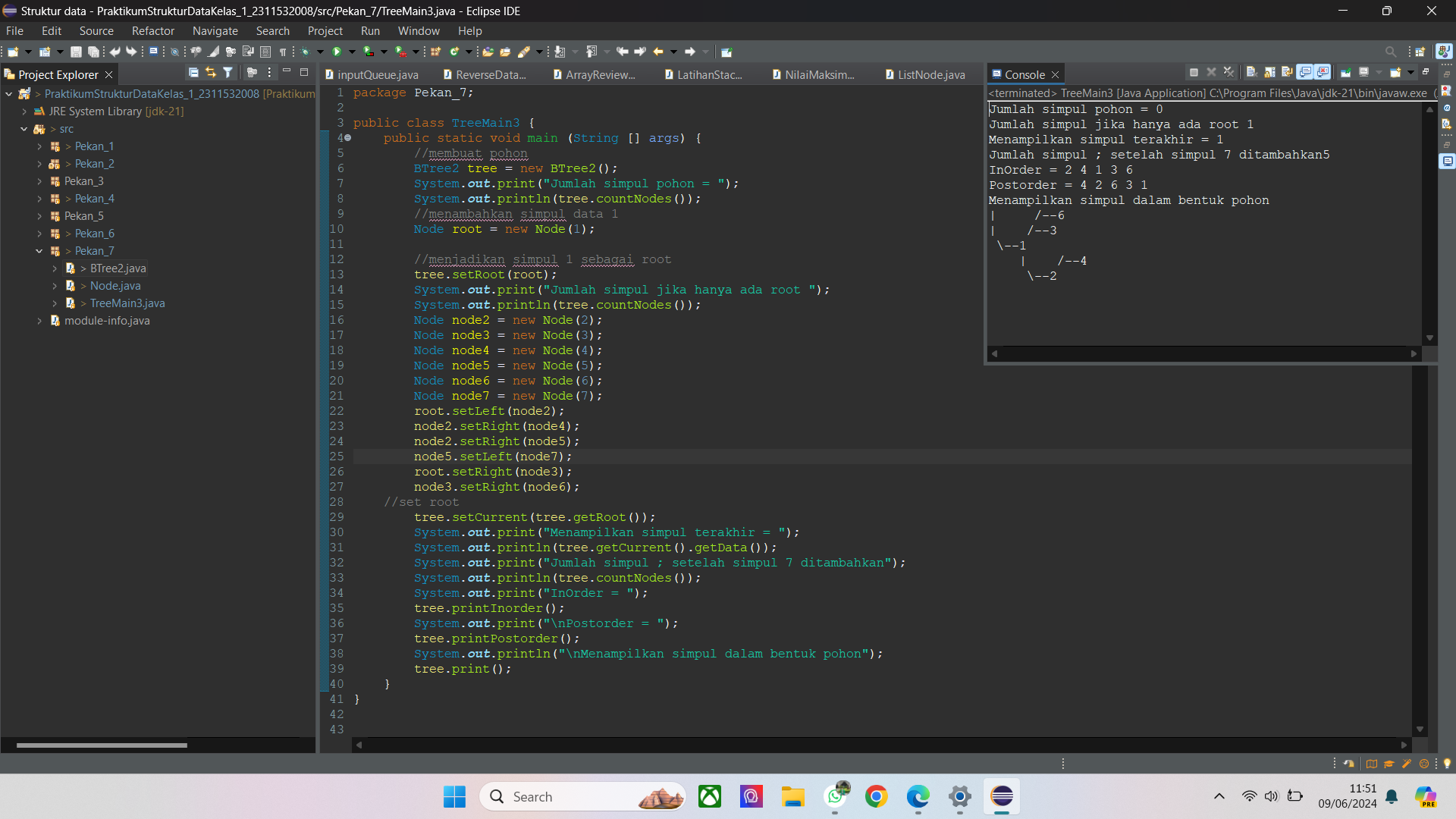
* Jika right (node kanan) tidak null, metode akan memanggil dirinya sendiri secara rekursif untuk mencetak subtree kanan terlebih dahulu, menggunakan prefix yang sesuai dan false untuk isTail karena subtree kanan bukanlah ekor.
* Kemudian, ia mencetak data dari node saat ini dengan format yang menunjukkan hubungan antara node dan anak-anaknya (kanan dan kiri) menggunakan simbol \\-- atau /-- tergantung pada nilai isTail.
* Jika left (node kiri) tidak null, metode akan memanggil dirinya sendiri secara rekursif untuk mencetak subtree kiri, menggunakan prefix yang sesuai dan true untuk isTail karena subtree kiri adalah ekor
* **Kelas**BTree2: Ini adalah kelas utama yang mendefinisikan pohon biner. Kelas ini memiliki dua variabel anggota, root dan currentNode. root adalah node akar dari pohon, sedangkan currentNode digunakan untuk menyimpan referensi ke node saat ini yang sedang diproses.
* **Konstruktor**BTree2(): Konstruktor ini menginisialisasi pohon dengan root yang bernilai null, yang menandakan bahwa pohon tersebut kosong pada awalnya.
* **Metode**search(int data): Metode publik ini digunakan untuk mencari nilai data di dalam pohon. Ia memanggil metode rekursif privat search(Node node, int data) dengan root sebagai titik awal pencarian.
* **Metode**search(Node node, int data): Ini adalah metode rekursif privat yang mencari nilai data di dalam pohon. Jika nilai data ditemukan di node saat ini, metode ini akan mengembalikan true. Jika tidak, metode ini akan mencari secara rekursif di subpohon kiri dan kanan.
* **Metode**printInorder()**,**printPreorder()**, dan**printPostorder(): Ketiga metode ini seharusnya mencetak elemen-elemen pohon dalam urutan inorder, preorder, dan postorder. Namun, tampaknya ada kesalahan karena metode printInorder, printPreorder, dan printPostorder tidak didefinisikan dalam kelas Node.
* **Metode**getRoot(): Metode ini mengembalikan node akar dari pohon.
* **Metode**isEmpty(): Metode ini memeriksa apakah pohon biner kosong (yaitu, root bernilai null).
* **Metode**countNodes(): Metode ini mengembalikan jumlah total node dalam pohon dengan memanggil metode rekursif privat countNodes(Node node).
* **Metode**countNodes(Node node): Ini adalah metode rekursif privat yang menghitung jumlah node dalam pohon. Jika node saat ini adalah null, metode ini mengembalikan 0. Jika tidak, metode ini menambahkan jumlah node di subpohon kiri dan kanan.
* **Metode**print(): Metode ini seharusnya mencetak pohon. Namun, tampaknya ada kesalahan karena root.print(); tidak akan bekerja kecuali metode print() didefinisikan dalam kelas Node.
* **Metode**getCurrent()**dan**setCurrent(Node node): Metode getCurrent() mengembalikan node saat ini, sedangkan setCurrent(Node node) mengatur node saat ini.
* **Metode**setRoot(Node root): Metode ini mengatur node akar dari pohon.

1. Mendefinisikan TreeMain, untuk mencetak pohon



* **Pembuatan Pohon**: Membuat objek BTree2 yang baru dan mencetak jumlah node saat pohon masih kosong (harusnya 0).
* **Penambahan dan Pengaturan Root**: Membuat node baru dengan data 1 dan mengatur node tersebut sebagai root dari pohon. Kemudian, mencetak jumlah node saat hanya ada root (harusnya 1).
* **Penambahan Node Lainnya**: Membuat node-node baru dengan data 2 hingga 7 dan mengatur hubungan antara node-node tersebut untuk membentuk struktur pohon.
* **Pengaturan Node Saat Ini**: Mengatur node saat ini (currentNode) ke root dan mencetak data dari node saat ini.
* **Mencetak Jumlah Node**: Mencetak jumlah total node setelah menambahkan semua node (harusnya 7).
* **Mencetak Pohon**: Mencetak elemen-elemen pohon dalam urutan InOrder dan PostOrder. Terdapat juga instruksi untuk mencetak pohon, tetapi metode print() belum didefinisikan dalam kelas Node.

Untuk hasil dari kode kode di atas adalah sebagai berikuit



1. KESIMPULAN

Pada kesimpulan bahwa untuk membuat Binary Tree, kita harus mendefinisikan beberapa kelas terlebih dahulu. Kelas BTree adalah pohon biner dan memiliki berbagai metode untuk melakukan operasi dasar pada pohon biner, seperti pencarian, penelusuran (traversal), dan pencetakan pohon. Berbagai metode untuk berinteraksi dengan pohon biner, seperti pencarian, penelusuran (traversal), menghitung jumlah node, dan pencetakan pohon, menjadikan metode ini berguna dalam banyak aplikasi pemrograman. Kita dapat membuat dan mengelola struktur pohon biner dengan kelas Node. Termasuk bentuk pohon biner yang akan dicetak, serta metode penanda dan penerima yang digunakan untuk mengatur nilai node dalam pohon biner. Selain itu, pohon biner dapat dicetak dalam tiga bentuk yaitu preorder, postorder, dan inorder. Kemudian dibuat kelas utama yang akan digunakan sehingga membentuk pohon biner.