Laporan Praktikum Struktur Data Pekan 7

DOSEN PENGAMPU:

Dr. Wahyudi S.T, M.T

OLEH:

Wahyu Khairi

2311531009

UNIVERSITAS ANDALAS

FAKULTAS TEKNOLOGI INFORMASI

JURUSAN INFORMATIKA

2023/2024

1. **TUJUAN**
2. Mengenal apa itu porting algorithm dalam bahasa pemrograman java
3. Memahami langkah langkah menjalankan program menggunakan porting algorithm
4. Memahami kapan porting algorithm dapat digunakan dalam sebuah program
5. **KAJIAN TEORI**
6. **Tree**

Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan hubungan yang bersifat hirarkis (hubungan one to many) antara elemen-elemen. Tree bisa didefinisikan sebagai kumpulan simpul/node dengan satu elemen khusus yang disebut Root dan node lainnya. Tree juga adalah suatu graph yang acyclic, simple, connected yang tidak mengandung loop.

Sebuah binary search tree (bst) adalah sebuah pohon biner yang boleh kosong, dan setiap nodenya harus memiliki identifier/value. Value pada semua node subpohon sebelah kiiri adalah selalu lebih kecil dari value dari root, sedangkan value subpohon di sebelah kanan adalah sama atau lebih besar dari value pada root, masing-masing subpohon tersebut (kiri dan kanan) itu sendiri adalah juga binary search tree.

Struktur data bst sangat penting dalam struktur pencarian, misalkan dalam kasus pencarian dalam sebuah list, jika list sudah dalam keadaan terurut maka proses pencarian akan semakin cepat, jika kita menggunakan list contigue dan melakukan pencarian biner,akan tetapi jika kita ingin melakukan perubahan isi list (insert atau delete), menggunakan list contigue akan sangat lambat, karena prose insert dan delete dalam list contigue butuh memindahkan linked-list, yang untuk operasi insert atau delete tinggal mengatur- atur pointer,akan tetapi pada n-linked list, kita tidak bisa melakukan pointer sembarangan setiap saat, kecuali hanya satu kali dengan kata lain hanya secara squential.

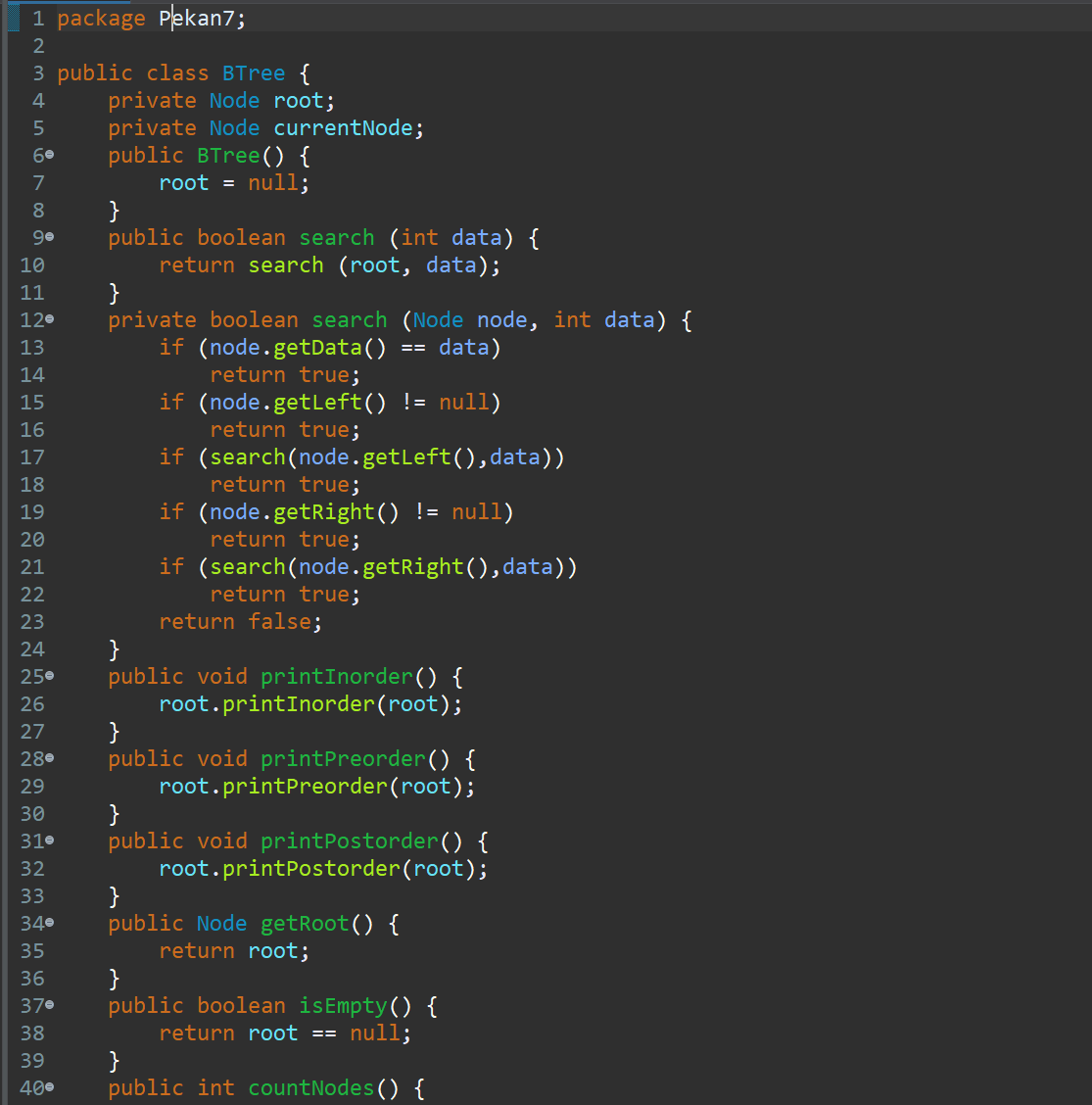
1. Btree

B-tree adalah [struktur data](https://builtin.com/data-science/data-structures" \t "_blank) yang menyediakan data terurut dan memungkinkan pencarian, akses berurutan, lampiran dan penghapusan dalam urutan terurut. B-tree sangat mampu menyimpan sistem yang menulis blok data dalam jumlah besar. B-tree menyederhanakan pohon pencarian biner dengan mengizinkan node dengan lebih dari dua anak.

1. Node

Node adalah struktur data dasar yang berisi data dan satu atau lebih tautan ke node lain. Node dapat digunakan untuk mewakili struktur pohon atau daftar tertaut. Dalam struktur yang menggunakan node, dimungkinkan untuk berpindah dari satu node ke node lainnya.

1. **LANGKAH PENGERJAAN**
2. **BTree**
3. Public Class BTree : Mendeklarasikan kelas publik bernama **BTree**.
4. Metode Public Boolean search(int data) : Metode ini mencari node dengan **data** nilai tertentu di pohon. Itu memanggil metode pribadi **search**untuk melakukan pencarian rekursif.
5. Metode private Boolean search(Node node, int data) :  Metode rekursif ini mencari data mulai dari yang diberikan **node**. Ia memeriksa data node saat ini, kemudian secara rekursif memanggil dirinya sendiri untuk subpohon kiri dan subpohon kanan jika diperlukan.
6. Metode (printInorder, printPostorder, printPreorder) : untuk memanggil metode kelas sesuai NODE
7. Metode public Node getRoot() :mengembalikan simpul akar
8. Metode public Boolean isEmpty() : memeriksa apakah pohonnya kosong (NULL)
9. Metode public int countNodes() : mengembalikan jumlah total node di pohon. Ini memmanggil metode private **countNodes** untuk menghitung node secara reekursif
10. Metode int countNodes (Node node) : Metode rekursif ini menghitung node mulai dari yang diberikan **node.** ia menambahkan 1 untuk node satt ini dan secara rekursif memanggil dirinya sendiri untuk subpohon kiri dan kanan.
11. Metode public void print() :mencetak pohon dalam beberapa format
12. Metode public Node getCurrent() ; mengemablikan **cureentNode**
13. Metode void setCurrent(Node node) : menyetel **currentNode** ke yang diberikan **node**
14. Metode void setRoot(Node root) ; menyetel **root** simpul pohon ke yang diberikan **node**

****

1. **Node**
2. Metode Node (int data) : menginisialisasi node baru dengan data yang diberikan
3. Metode setLeft (Node node), setRight (Node node) : mengatur masisng masing node anak kiri dan kanan
4. Metode getLeft(), getRight(), getData() : mengambalikan anak kiri, anak kanan, dan data node
5. SetData(int data) : memperbarui nilai data node
6. Metode Transversal : printPreorder(Node node), printPostorder(Node node), printInorder(Node node) : mengimplementasikan metode preorder, postorder, dan inorder transversal untuk pohon
7. Metode print(), print(String prefix, Boolean istail, String sb) : metode ini digunakan untuk mencetak struktur pohon dalam format yang menarik secara visual, dengan data dan koneksi setiap node ditampilkan, dan metode ini bersifat rekursif dan membangun representasi string dari pohon.

****

1. **TreeMain**
2. Membuat pohon biner

* Kode dimulai dengan membuat sebuah instance dari sebuah BTreekelas. Kelas ini mungkin mewakili struktur pohon biner.
* Kode dimulai dengan membuat sebuah instance dari sebuah BTreekelas. Kelas ini mungkin mewakili struktur pohon biner.

1. Menambahkan Node dan mengatur Root

* Kode membuat yang baru Nodedengan nilai data 1 dan menetapkannya ke rootvariabel.
* Panggilan tersebut tree.setRoot(root)menetapkan rootnode BTreeyang baru dibuat.
* Panggilan tersebut tree.setRoot(root)menetapkan rootnode BTreeyang baru dibuat.

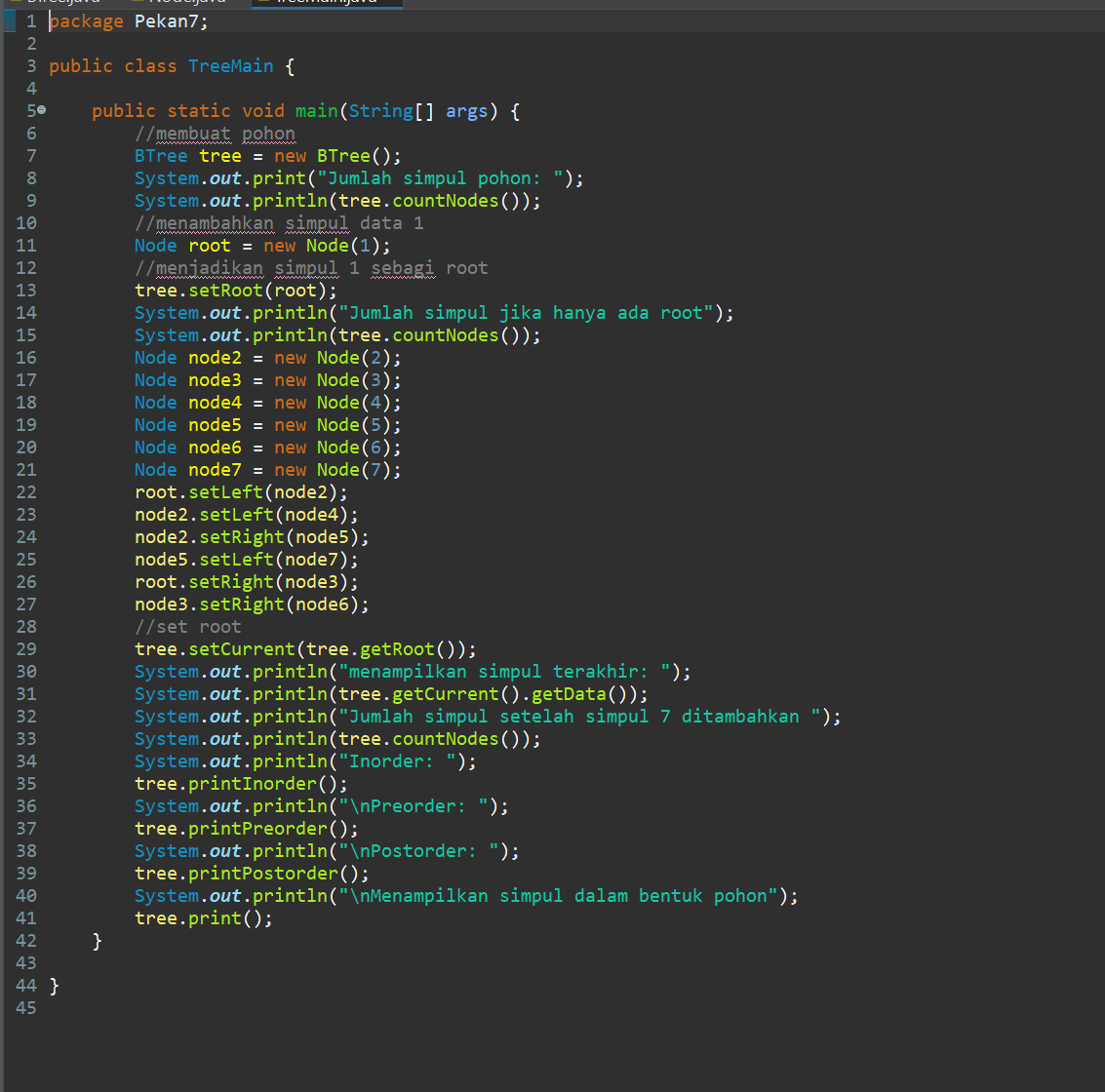
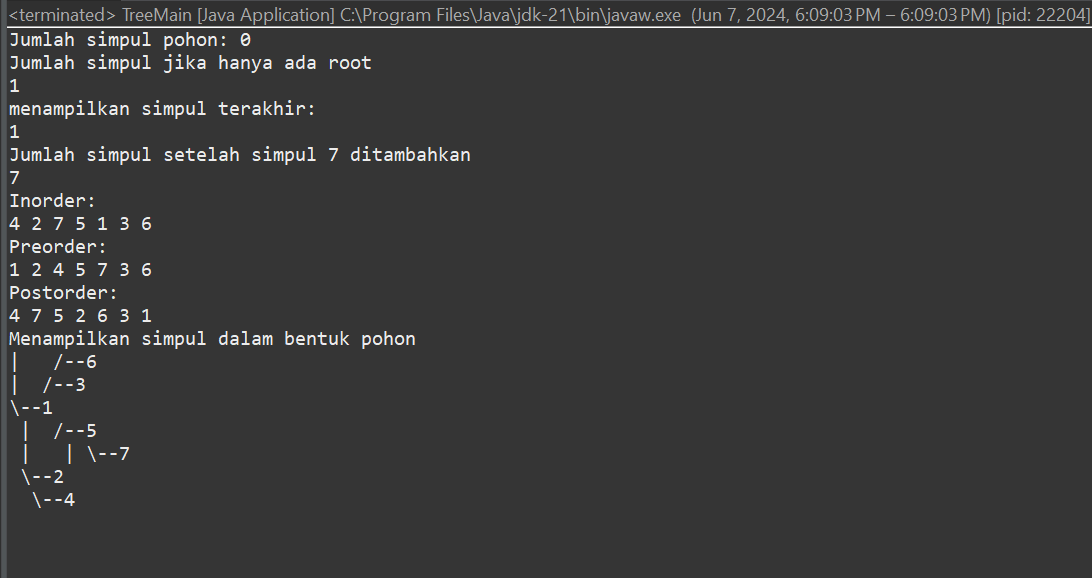
1. Menambahkan Lebih Banyak Node :

* Kode tersebut kemudian membuat Nodeobjek tambahan dengan nilai data dari 2 hingga 7.
* Kode tersebut kemudian membuat Nodeobjek tambahan dengan nilai data dari 2 hingga 7.
* ini menciptakan struktur pohon di mana :
* Ia rootmempunyai node2anak kiri dan node3anak kanan.
* node2memiliki node4sebagai anak kirinya dan node5sebagai anak kanannya.
* node5miliki node7sebagai anak kirinya.
* node3memiliki node6anak yang tepat.

1. Melintasi pohon

* Kode ini digunakan tree.setCurrent(tree.getRoot());untuk mengatur node saat ini untuk rootmelintasi pohon.
* Ia kemudian mencetak nilai data dari node saat ini, yang merupakan nilai data root (1 dalam kasus ini).
* Panggilan tersebut tree.countNodes()mencetak jumlah node setelah menambahkan semua node baru.
* Kode tersebut kemudian mencetak konten pohon dalam urutan traversal yang berbeda:
* norder: Ini mencetak node dalam urutan: subpohon kiri, akar, subpohon kanan.
* Preorder: Ini mencetak node dalam urutan: root, subpohon kiri, subpohon kanan.
* Postorder: Ini mencetak node dalam urutan: subpohon kiri, subpohon kanan, root.

1. Memvisualisasikan pohon :

* Terakhir, baris tersebut tree.print();kemungkinan memanggil metode untuk menampilkan struktur pohon secara visual, mungkin menggunakan string atau diagram.
* Output