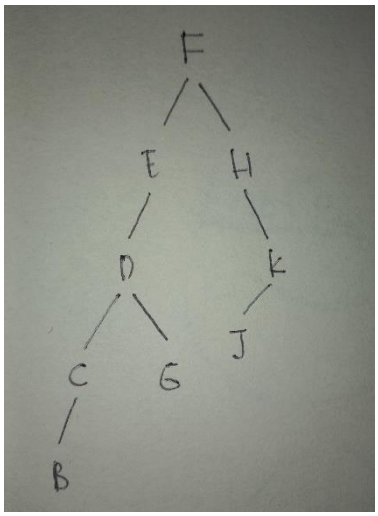
NAMA : FITRIYA NUR ALYSHIFA

NIM : 607062300097

KELAS : 47-02

LINK REPO : *https://github.com/fitriyanuralyashifa/Jurnal13_607062300097_fitriyanuralyashifa.git*

GAMBAR BST



Konsep BST : setiap node memiliki nilai lebih kecil dari node induknya berada di sisi kiri, dan untuk nilai yang lebih besar dari node induknya berada di sisi kanan.

Jadi :

E lebih kecil dari F, H lebih besar dari F, D lebih kecil dari E, C lebih kecil dari D, B lebih kecil dari C, sedangkan G lebih besar dari D. K lebih besar dari H, dan J lebih kecil dari K.

Buatlah program untuk membentuk Binary Search Tree (BST) untuk insertion dan searching. Masukkan data berikut ke dalam program tersebut, gambarkan BST yang terbentuk. F, E, H, D, G, C, B, H, K, J Dari BST yang terbentuk, carilah karakter K dan A.

Node :

```
public class Node {
    char data;
    Node left, right;

    public Node(char item) {
        data = item;
```

```
        left = right = null;
    }
}
```

BST :

```
import java.util.Scanner;

public class BST {
    Node root;

    // Konstruktor
    public BST() {
        root = null;
    }

    // Metode untuk menyisipkan data ke dalam BST
    public void insert(char data) {
        root = insertRec(root, data);
    }

    // Metode rekursif untuk menyisipkan data ke dalam BST
    private Node insertRec(Node root, char data) {
        // Jika pohon kosong, buat node baru
        if (root == null) {
            root = new Node(data);
            return root;
        }

        // Jika tidak, rekursi ke bawah pohon
        if (data < root.data) {
            root.left = insertRec(root.left, data);
        } else if (data > root.data) {
            root.right = insertRec(root.right, data);
        }

        // Mengembalikan pointer node (tidak berubah)
        return root;
    }

    // Metode untuk mencari data dalam BST
    public boolean search(char data) {
        return searchRec(root, data);
    }
```

```
    // Metode rekursif untuk mencari data dalam BST
    private boolean searchRec(Node root, char data) {
        // Kasus dasar: root null atau data ditemukan pada root
        if (root == null) {
            return false;
        }
        if (root.data == data) {
            return true;
        }

        // Jika data lebih besar dari data root
        if (root.data < data) {
            return searchRec(root.right, data);
        }

        // Jika data lebih kecil dari data root
        return searchRec(root.left, data);
    }

    public static void main(String[] args) {
        BST tree = new BST();

        // Data yang diberikan
        char[] data = {'F', 'E', 'H', 'D', 'G', 'C', 'B', 'H', 'K', 'J'};

        // Membentuk BST dengan data di atas
        for (char datum : data) {
            tree.insert(datum);
        }

        // Memungkinkan pengguna untuk memasukkan karakter yang ingin dicari
        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan karakter yang ingin dicari: ");
        char charToSearch = scanner.next().charAt(0);

        // Mencari karakter yang dimasukkan pengguna
        boolean found = tree.search(charToSearch);
        if (found) {
            System.out.println("Karakter '" + charToSearch + "' ditemukan di
BST.");
        } else {
            System.out.println("Karakter '" + charToSearch + "' tidak ditemukan
di BST.");
        }
```

```
        scanner.close();
    }
}
```