Laporan Hasil Praktikum
ALSD 13

NAMA : Arifah Zhafirah Wikananda

NIM : 244107020188

KELAS : 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

14.2 Kegiatan Praktikum 1

Implementasi Binary Search Tree menggunakan Linked List (100 Menit)

14.2.1 Percobaan 1

```
package jobsheet13;

public class Mahasiswa04 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa04() {
    }

    public Mahasiswa04(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampilInformasi() {
        System.out.println("NIM: " +this.nim+" "+"Nama: " +this.nama+"
"+"Kelas: "+this.kelas+" "+"IPK: "+this.ipk);
    }
}
```

```
package jobsheet13;

public class Node04 {
    Mahasiswa04 mahasiswa;
    Node04 left, right;

    public Node04() {
    }

    public Node04(Mahasiswa04 mahasiswa) {
        this.mahasiswa = mahasiswa;
        left = right = null;
    }
}
```

```
package jobsheet13;

public class BinaryTree04 {
    Node04 root;

    public BinaryTree04() {
        root = null;
    }

    public boolean isEmpty() {
        return root == null;
    }
```

```
    public void add(Mahasiswa04 mahasiswa) {
        Node04 newNode = new Node04(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node04 current = root;
            Node04 parent = null;
            while(true){
                parent = current;
                if(mahasiswa.ipk < current.mahasiswa.ipk) {
                    current = current.left;
                    if(current == null) {
                        parent.left = newNode;
                        return;
                    }
                } else {
                    current = current.right;
                    if(current == null) {
                        parent.right = newNode;
                        return;
                    }
                }
            }
        }
    }
    boolean find(double ipk){
            boolean result = false;
            Node04 current = root;
            while(current != null) {
                if(current.mahasiswa.ipk == ipk) {
                    result = true;
                    break;
                } else if(ipk > current.mahasiswa.ipk) {
                    current = current.right;
                } else {
                    current = current.left;
                }
            }
            return result;
        }
    void traversePreOrder(Node04 node) {
        if (node != null) {
            node.mahasiswa.tampilInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }
    void traverseInOrder(Node04 node) {
        if (node != null) {
            traverseInOrder(node.left);
            node.mahasiswa.tampilInformasi();
            traverseInOrder(node.right);
        }
    }
    void traversePostOrder(Node04 node) {
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.mahasiswa.tampilInformasi();
        }
    }
```

```
    Node04 getSuccessor(Node04 del) {
        Node04 successor= del.right;
        Node04 successorParent = del;
        while (successor.left != null){
            successorParent = successor;
            successor = successor.left;
        }if (successor != del.right){
            successorParent.left = successor.right;
            successor.right = del.right;
        }
        return successor;
    }

    void delete(double ipk) {
        if(isEmpty()) {
            System.out.println("Binary tree kosong");
            return;
        }
        // cari node (current) yang akan dihapus
        Node04 parent = root;
        Node04 current = root;
        boolean isLeftChild = false;
        while (current != null) {
            if(current.mahasiswa.ipk == ipk) {
                break;
            }else if (ipk < current.mahasiswa.ipk) {
                parent = current;
                current = current.left;
                isLeftChild = true;
            } else if(ipk > current.mahasiswa.ipk){
                parent = current;
                current = current.right;
                isLeftChild = false;
            }
        }
    // penghapusan
    if (current == null) {
        System.out.println("Data tidak ditemukan");
        return;
    } else {
        // jika tidak ada anak (leaf), maka node dihapus
        if(current.left == null && current.right == null) {
            if(current == root) {
                root = null;
            } else {
                if(isLeftChild) {
                    parent.left = null;
                } else {
                    parent.right = null;
                }
            }
        }else if (current.left == null){ //jika hanya punya 1 anak (kanan)
            if(current == root) {
                root = current.right;
            } else {
                if(isLeftChild) {
                    parent.left = current.right;
                } else {
                    parent.right = current.right;
                }
            }
```

```
        } else if (current.right == null) { //jika hanya punya 1 anak (kiri)
            if(current == root) {
                root = current.left;
            } else {
                if(isLeftChild) {
                    parent.left = current.left;
                } else {
                    parent.right = current.left;
                }
            }
        } else { //jika punya 2 anak
            Node04 successor = getSuccessor(current);
            System.out.println("Jika 2 anak, current = ");
            successor.mahasiswa.tampilInformasi();
            if(current == root) {
                root = successor;
            } else {
                if(isLeftChild) {
                    parent.left = successor;
                } else {
                    parent.right = successor;
                }
            }
            successor.left = current.left;
            }
        }
    }
}
```

Outputnya:

```
c:\Praktikum ASD>
c:\Praktikum ASD> c: && cd "c:\Praktikum ASD" && cmd /C ""C:\Prog
ram Files\Java\jdk-21\bin\java.exe" -XX:+ShowCodeDetailsInExcepti
onMessages -cp "C:\Users\WINDOWS 11\AppData\Roaming\Code\User\wor
kspaceStorage\9ab0f952d31733a7c7e6cca34608067d\redhat.java\jdt_ws
\Praktikum ASD_c51dcdc\bin" jobsheet13.BinaryTreeMain04 "

Daftar semua mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

Pencarian data mahasiswa:
Cari mahasiswa dengan ipk: 3.54 :Ditemukan
Cari mahasiswa dengan ipk: 3.22 :Tidak ditemukan
```

```
Daftar semua data mahasiswa setelah penambahan 3 mahasiswa:
InOrder Traversal:
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

PreOrder Traversal:
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

PostOrder Traversal:
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
```

```
Penghapusan data mahasiswa
Jika 2 anak, current =
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua data mahasiswa setelah penghapusan 1 mahasiswa (in order traversal)
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

14.2.2 Pertanyaan Percobaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
Jawab: Binary Search Tree memungkinkan pencarian data lebih cepat dan efisien karena memiliki struktur yang teratur dan terurut, sehingga kita bisa langsung menuju ke cabang yang kemungkinan besar mmenyimpan data, tanpa harus mengecek semua node.

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?
Jawab: Gunanya sama dengan kegunaan pointer dalam double linked list, left menunjuk pada child kiri dan right menunjuk pada child kanan.

3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?
Jawab: Sebagai node yang berada paling atas dan tidak memiliki parent.

b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?
Jawab: Kondisi awal root berisi nilai null

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa

yang akan terjadi?
Jawab: Ketika masih kosong maka node root akan diisi dengan data baru yang ingin ditambahkan.

5. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?

```
parent = current;
if (mahasiswa.ipk < current.mahasiswa.ipk) {
    current = current.left;
    if (current == null) {
        parent.left = newNode;
        return;
    }
} else {
    current = current.right;
    if (current == null) {
        parent.right = newNode;
        return;
    }
}
```

Jawab:

1. parent = current;

- Menyimpan **referensi node saat ini (current) sebagai parent**.
- Ini penting karena nanti jika current menjadi null, kita perlu tahu **siapa node sebelumnya** untuk menentukan di mana newNode harus ditambahkan.

2. if (mahasiswa.ipk < current.mahasiswa.ipk) {

- Melakukan **perbandingan IPK**:
  Jika IPK dari mahasiswa yang baru ingin ditambahkan lebih kecil dari IPK node saat ini:
  - Maka newNode harus ditempatkan di **subtree kiri** (karena lebih kecil → kiri).

3. current = current.left;

- Berpindah ke anak kiri (left child) untuk melanjutkan pencarian tempat yang cocok bagi newNode.

4. if (current == null) {

- Jika tidak ada node di kiri (null), maka:
  - **Tempat ini kosong → cocok untuk menempatkan newNode.**

5. parent.left = newNode;

- Menghubungkan node baru ke sisi kiri parent.

6. else { ... }

- Jika IPK lebih besar atau sama, maka newNode harus ditempatkan di **subtree kanan**.
- Langkah-langkahnya sama seperti di subtree kiri, hanya beda arah.

Proses Ini Akan Diulang dan Biasanya kode ini dibungkus dalam loop seperti while (true) atau while (current != null) yang akan terus berjalan hingga ditemukan posisi null (kosong) di kiri atau kanan — tempat yang tepat untuk menambahkan newNode.

6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?

Jawab: Langkah-langkah penghapusan node dengan 2 anak:

1. Cari **successor** dari node yang akan dihapus (yaitu node terkecil di subtree kanannya).

2. Gantikan node yang dihapus dengan successor.

3. Hubungkan ulang semua child node agar struktur BST valid.

peran **getSuccessor():**

- Mengidentifikasi node pengganti (successor) yang tepat.
- Menyusun ulang child nodes agar tidak ada node yang terlepas dari tree.

14.3 Kegiatan Praktikum 2
Implementasi Binary Tree dengan Array (45 Menit)
14.3.1 Tahapan Percobaan

```
package jobsheet13;

public class BinaryTreeArray04 {
    Mahasiswa04[] dataMahasiswa;
    int idxLast;

    public BinaryTreeArray04() {
        this.dataMahasiswa = new Mahasiswa04[10];
    }

    void populateData(Mahasiswa04 dataMhs[], int idxLast) {
        this.dataMahasiswa = dataMhs;
        this.idxLast = idxLast;
    }

    void traverseInOrder(int idxStart){
        if(idxStart <= idxLast) {
            if(dataMahasiswa[idxStart] != null) {
                traverseInOrder(2*idxStart + 1);
                dataMahasiswa[idxStart].tampilInformasi();
                traverseInOrder(2 * idxStart + 2);
            }
        }
    }
}
```

```
package jobsheet13;

public class BinaryTreeArrayMain04 {
    public static void main(String[] args) {
        BinaryTreeArray04 bta = new BinaryTreeArray04();

        Mahasiswa04 mhs1 = new Mahasiswa04("244160121", "Ali", "A", 3.57);
        Mahasiswa04 mhs2 = new Mahasiswa04("244160185", "Candra", "C", 3.41);
        Mahasiswa04 mhs3 = new Mahasiswa04("244160221", "Badar", "B", 3.75);
        Mahasiswa04 mhs4 = new Mahasiswa04("244160220", "Dewi", "B", 3.35);

        Mahasiswa04 mhs5 = new Mahasiswa04("244160131", "Devi", "A", 3.48);
        Mahasiswa04 mhs6 = new Mahasiswa04("244160205", "Ehsan", "D", 3.61);
        Mahasiswa04 mhs7 = new Mahasiswa04("244160170", "Fizi", "B", 3.86);

        Mahasiswa04[] dataMahasiswas = {mhs1, mhs2, mhs3, mhs4, mhs5, mhs6, mhs7,
null, null, null};
        int idxLast = 6;
        bta.populateData(dataMahasiswas, idxLast);
        System.out.println("\nInorder Traversal Mahasiswa :");
        bta.traverseInOrder(0);
    }
}
```

Outputnya:

```
C:\Praktikum ASD> cmd /C ""C:\Program Files\Java\jdk-21\bin\java.
exe" -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\WINDOW
S 11\AppData\Roaming\Code\User\workspaceStorage\9ab0f952d31733a7c
7e6cca34608067d\redhat.java\jdt_ws\Praktikum ASD_c51dcdc\bin" job
sheet13.BinaryTreeArrayMain04 "

Inorder Traversal Mahasiswa :
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.41
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86
```

14.3.2 Pertanyaan Percobaan

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?
Jawab:Atribut data berfungsi untuk menyimpan array dari node-node pada tree, sedangkan idxLast adalah varibale untuk menyimpan value index terakhir dalam tree.

2. Apakah kegunaan dari method populateData()?
Jawab: Method tersebut berguna sebagai konstruktor yang mengatur kondisi awal dari array data dan isi dari variabel idxLast yang didapat melalui parameter.

3. Apakah kegunaan dari method traverseInOrder()?
Jawab: Berguna untuk menelusuri tree secara dengan metode in order.

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?
Jawab: Posisi left child = 2 * (2) + 1 = 5 dan posisi right child = 2 * (2) + 2 = 6

5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
Jawab: Untuk menetapkan index terakhir dari tree menjadi 6, dibuat 6 karena data yang valid untuk dimasukkan kedalam tree hanya ada 6.

6. Mengapa indeks 2*idxStart+1 dan 2*idxStart+2 digunakan dalam pemanggilan rekursif, dan apa kaitannya dengan struktur pohon biner yang disusun dalam array?
Jawab: Indeks 2*i+1 dan 2*i+2 digunakan karena **merepresentasikan anak kiri dan kanan dari node di indeks i** dalam struktur pohon biner yang disusun dalam array. Hal ini menggantikan penggunaan objek Node dan pointer seperti pada struktur linked-node tree.

14.4 Tugas Praktikum Waktu pengerjaan: 150 menit

1. Buat method di dalam class BinaryTree00 yang akan menambahkan node dengan cara rekursif (addRekursif()).
2. Buat method di dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK paling kecil dan IPK yang paling besar (cariMinIPK() dan cariMaxIPK()) yang ada di dalam binary search tree.
3. Buat method dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK di atas suatu batas tertentu, misal di atas 3.50 (tampilMahasiswaIPKdiAtas(double ipkBatas)) yang ada di dalam binary search tree.
4. Modifikasi class BinaryTreeArray00 di atas, dan tambahkan :
    - method add(Mahasiswa data) untuk memasukan data ke dalam binary tree
    - method traversePreOrder()

```
Node04 addRekursif(Node04 root, Mahasiswa04 mahasiswa) {
        Node04 newNode = new Node04(mahasiswa);
        if (root == null) {
            root = newNode;
        } else if (newNode.mahasiswa.ipk < root.mahasiswa.ipk) {
            root.left = addRekursif(root.left, mahasiswa);
        } else if (newNode.mahasiswa.ipk >= root.mahasiswa.ipk) {
            root.right = addRekursif(root.right, mahasiswa);
        }
        return root;
    }

    public void cariMinIPK() {
        Node04 current = root;
        if (current == null) {
            System.out.println("Tree kosong");
            return;
        }

        while (current.left != null) {
            current = current.left;
        }

        System.out.println("Mahasiswa dengan IPK minimum:");
        current.mahasiswa.tampilInformasi();
    }

    public void cariMaxIPK() {
        Node04 current = root;
        if (current == null) {
            System.out.println("Tree kosong");
            return;
        }

        while (current.right != null) {
            current = current.right;
        }

        System.out.println("Mahasiswa dengan IPK maksimum:");
        current.mahasiswa.tampilInformasi();
    }
```

```
    public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
        tampilMahasiswaDiAtas(root, ipkBatas);
    }

    void tampilMahasiswaDiAtas(Node04 node, double batas) {
        if (node != null) {
            tampilMahasiswaDiAtas(node.left, batas);
            if (node.mahasiswa.ipk > batas) {
                node.mahasiswa.tampilInformasi();
            }
            tampilMahasiswaDiAtas(node.right, batas);
        }
    }

    public void addArrayBased(Mahasiswa04 data) {
        if (idxLast + 1 < dataMahasiswa.length) {
            dataMahasiswa[++idxLast] = data;
        } else {
            System.out.println("Tree penuh");
        }
    }

    public void traversePreOrder(int idxStart) {
        if (idxStart <= idxLast && dataMahasiswa[idxStart] != null) {
            dataMahasiswa[idxStart].tampilInformasi();
            traversePreOrder(2 * idxStart + 1);
            traversePreOrder(2 * idxStart + 2);
        }
    }
```

```
import java.util.Scanner;

public class MahasiswaDemo04 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        MahasiswaBerprestasi04 list = new MahasiswaBerprestasi04();

        for (int i = 0; i < 5; i++) {
            System.out.println("Masukkan Mahasiswa ke-" + (i+1) +":");
            System.out.print("NIM: ");
            String nim = sc.nextLine();
            System.out.print("Nama: ");
            String nama = sc.nextLine();
            System.out.print("Kelas: ");
            String kelas = sc.nextLine();
            System.out.print("IPK: ");
            double ipk = sc.nextDouble();
            System.out.println("---------------------------------");
            sc.nextLine();

            Mahasiswa04 m = new Mahasiswa04(nim, nama, kelas, ipk);
            list.tambah(m);

        }
        System.out.println("\nData mahasiswa sebelum sorting: ");
        list.tampil();

        System.out.println("Data mahasiswa setelah sorting berdasarkan IPK (DESC): ");
        list.bubbleSort();
        list.tampil();

        sc.close();
    }

}
```

Outputnya:

```
xtension-pack-jdk\java\latest\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -
paceStorage\d83555ec4f9c765b284f331a909f075b\redhat.java\jdt_ws\Praktikum05_b1713034\bin" MahasiswaDem
Masukkan Mahasiswa ke-1:
NIM: 123
Nama: rere
Kelas: 2e
IPK: 3,4
--------------------------------
Masukkan Mahasiswa ke-2:
NIM: 124
Nama: tete
Kelas: 2e
IPK: 3,5
--------------------------------
Masukkan Mahasiswa ke-3:
NIM: 126
Nama: yeye
Kelas: 2e
IPK: 3,2
--------------------------------
Masukkan Mahasiswa ke-4:
NIM: 127
Nama: wewe
Kelas: 2e
IPK: 3,1
--------------------------------
Masukkan Mahasiswa ke-5:
NIM: 128
Nama: fefe
Kelas: 2e
IPK: 3,7
--------------------------------
```

```
Data mahasiswa sebelum sorting:
Nama : rere
NIM : 123
Kelas : 2e
ipk : 3.4
------------------------------
Nama : tete
NIM : 124
Kelas : 2e
ipk : 3.5
------------------------------
Nama : yeye
NIM : 126
Kelas : 2e
ipk : 3.2
------------------------------
Nama : wewe
NIM : 127
Kelas : 2e
ipk : 3.1
------------------------------
Nama : fefe
NIM : 128
Kelas : 2e
ipk : 3.7
------------------------------
Data mahasiswa setelah sorting berdasarkan IPK (DESC):
Nama : fefe
NIM : 128
Kelas : 2e
ipk : 3.7
------------------------------
 Data mahasiswa setelah sorting berdasarkan IPK (DESC):
 Nama : fefe
 NIM : 128
 Kelas : 2e
 ipk : 3.7
 ------------------------------
 Nama : tete
 NIM : 124
 Kelas : 2e
 ipk : 3.5
 ------------------------------
 Nama : rere
 NIM : 123
 Kelas : 2e
 ipk : 3.4
 ------------------------------
 Nama : yeye
 NIM : 126
 Kelas : 2e
 ipk : 3.2
 ------------------------------
 Nama : wewe
 NIM : 127
 Kelas : 2e
 ipk : 3.1
 ------------------------------
```

5.3.5 Mengurutkan Data Mahasiswa Berdasarkan IPK (Selection Sort)

```
public class MahasiswaBerprestasi04 {
    Mahasiswa04[] listMhs = new Mahasiswa04[5];
    int idx;

    void tambah (Mahasiswa04 m){
        if (idx < listMhs.length){
            listMhs[idx] = m;
            idx++;
        }else{
            System.out.println("Data Sudah Penuh");
        }
    }

    void tampil(){
        for (Mahasiswa04 m:listMhs){
            m.tampilInformasi();
            System.out.println("------------------------------");
        }
    }

    void bubbleSort(){
        for (int i = 0; i < listMhs.length-1; i++){
            for ( int j = 1; j < listMhs.length-i; j++){
                if (listMhs[j].ipk > listMhs[j-1].ipk){
                    Mahasiswa04 tmp = listMhs[j];
                    listMhs[j] = listMhs[j-1];
                    listMhs[j-1] = tmp;
                }
            }
        }
    }

    void selectionSort(){
        for (int i = 0; i < listMhs.length-1; i++){
            int idxMin = i;
            for (int j = i+1; j < listMhs.length; j++){
                if (listMhs[j].ipk < listMhs[idxMin].ipk){
                    idxMin = j;
                }
            }
            Mahasiswa04 tmp = listMhs[idxMin];
            listMhs[idxMin] = listMhs[i];
            listMhs[i] = tmp;
        }
    }
}
```

```
public class MahasiswaDemo04 {
    public static void main(String[] args) {
    MahasiswaBerprestasi04 list = new MahasiswaBerprestasi04();
    Mahasiswa04 m1 = new Mahasiswa04("123", "Zidan", "2A", 3.2);
    Mahasiswa04 m2 = new Mahasiswa04("124", "Ayu", "2A", 3.5);
    Mahasiswa04 m3 = new Mahasiswa04("125", "Sofi", "2A", 3.1);
    Mahasiswa04 m4 = new Mahasiswa04("126", "Sita", "2A", 3.9);
    Mahasiswa04 m5 = new Mahasiswa04("127", "Miki", "2A", 3.7);

    list.tambah(m1);
    list.tambah(m2);
    list.tambah(m3);
    list.tambah(m4);
    list.tambah(m5);

    System.out.println("Data Mahasiswa Sebelum Sorting: ");
    list.tampil();

    System.out.println("Data Mahasiswa Setelah Sorting Berdasarkan IPK (DESC): ");
    list.bubbleSort();
    list.tampil();

    System.out.println("data yang sudah terurut menggunakan SELECTION SORT (ASC)");
    list.selectionSort();
    list.tampil();
    }

}
```

Outputnya:

```
c:\Praktikum ASD\Praktikum05>
c:\Praktikum ASD\Praktikum05> c: && cd "c:\Praktikum ASD\Praktikum05" && cmd /C ""C:\Users\WINDOWS 11\AppData\Roaming\Code\
xtension-pack-jdk\java\latest\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\WINDOWS
paceStorage\d83555ec4f9c765b284f331a909f075b\redhat.java\jdt_ws\Praktikum05_b1713034\bin" MahasiswaDemo04 "
Data Mahasiswa Sebelum Sorting:
Nama : Zidan
NIM : 123
Kelas : 2A
ipk : 3.2
-----------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
ipk : 3.5
-----------------------------
Nama : Sofi
NIM : 125
Kelas : 2A
ipk : 3.1
-----------------------------
Nama : Sita
NIM : 126
Kelas : 2A
ipk : 3.9
-----------------------------
Nama : Miki
NIM : 127
Kelas : 2A
ipk : 3.7
-----------------------------
```

```
data yang sudah terurut menggunakan SELECTION SORT (ASC)
Nama : Sofi
NIM : 125
Kelas : 2A
ipk : 3.1
-----------------------------
Nama : Zidan
NIM : 123
Kelas : 2A
ipk : 3.2
-----------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
ipk : 3.5
-----------------------------
Nama : Miki
NIM : 127
Kelas : 2A
ipk : 3.7
-----------------------------
Nama : Sita
NIM : 126
Kelas : 2A
ipk : 3.9
-----------------------------
```

Pertanyaan:
Di dalam method selection sort, terdapat baris program seperti di bawah ini:

```
int idxMin=i;
for (int j=i+1; j<listMhs.length; j++){
    if (listMhs[j].ipk<listMhs[idxMin].ipk){
        idxMin=j;
    }
}
```

Untuk apakah proses tersebut, jelaskan!
Jawab:

- ✓ Kode ini berfungsi untuk mencari indeks elemen dengan nilai terkecil dalam array**.**
- ✓ Digunakan dalam Selection Sort untuk menemukan elemen terkecil sebelum dilakukan pertukaran**.**
- ✓ Menjadi dasar dari proses pengurutan Selection Sort, yang bekerja dengan memilih elemen terkecil secara bertahap**.**

5.4 Mengurutkan Data Mahasiswa Berdasarkan IPK Menggunakan Insertion Sort

```
public class MahasiswaBerprestasi04 {
    Mahasiswa04[] listMhs = new Mahasiswa04[5];
    int idx;

    void tambah (Mahasiswa04 m){
        if (idx < listMhs.length){
            listMhs[idx] = m;
            idx++;
        }else{
            System.out.println("Data Sudah Penuh");
        }
    }

    void tampil(){
        for (Mahasiswa04 m:listMhs){
            m.tampilInformasi();
            System.out.println("------------------------------");
        }
    }

    void bubbleSort(){
        for (int i = 0; i < listMhs.length-1; i++){
            for ( int j = 1; j < listMhs.length-i; j++){
                if (listMhs[j].ipk > listMhs[j-1].ipk){
                    Mahasiswa04 tmp = listMhs[j];
                    listMhs[j] = listMhs[j-1];
                    listMhs[j-1] = tmp;
                }
            }
        }
    }

    void selectionSort(){
        for (int i = 0; i < listMhs.length-1; i++){
            int idxMin = i;
            for (int j = i+1; j < listMhs.length; j++){
                if (listMhs[j].ipk < listMhs[idxMin].ipk){
                    idxMin = j;
                }
            }
            Mahasiswa04 tmp = listMhs[idxMin];
            listMhs[idxMin] = listMhs[i];
            listMhs[i] = tmp;
        }
    }

void insertionSort(){
    for (int i = 1; i < listMhs.length; i++){
        Mahasiswa04 temp = listMhs[i];
        int j = i;
        while (j > 0 && listMhs[j-1].ipk > temp.ipk){
            listMhs[j] = listMhs[j-1];
            j--;
        }
        listMhs[j] = temp;
    }
}
}
```

```
import java.util.Scanner;
public class MahasiswaDemo04 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
    MahasiswaBerprestasi04 list = new MahasiswaBerprestasi04();
    Mahasiswa04 m1 = new Mahasiswa04("123", "Zidan", "2A", 3.2);
    Mahasiswa04 m2 = new Mahasiswa04("124", "Ayu", "2A", 3.5);
    Mahasiswa04 m3 = new Mahasiswa04("125", "Sofi", "2A", 3.1);
    Mahasiswa04 m4 = new Mahasiswa04("126", "Sita", "2A", 3.9);
    Mahasiswa04 m5 = new Mahasiswa04("127", "Miki", "2A", 3.7);

    list.tambah(m1);
    list.tambah(m2);
    list.tambah(m3);
    list.tambah(m4);
    list.tambah(m5);

    System.out.println("Data Mahasiswa Sebelum Sorting: ");
    list.tampil();

    System.out.println("Data Mahasiswa Setelah Sorting Berdasarkan IPK (DESC): ");
    list.bubbleSort();
    list.tampil();

    System.out.println("data yang sudah terurut menggunakan SELECTION SORT (ASC)");
    list.selectionSort();
    list.tampil();

    System.out.println("Data yang sudah terurut menggunakan INSERTION SORT (ASC)");
    list.insertionSort();
    list.tampil();
    }

}
```

Outputnya:

```
xtension-pack-jdk\java\latest\bin\java.exe" --ena
paceStorage\d83555ec4f9c765b284f331a909f075b\redH
Data Mahasiswa Sebelum Sorting:
Nama : Zidan
NIM : 123
Kelas : 2A
ipk : 3.2
-----------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
ipk : 3.5
-----------------------------
Nama : Sofi
NIM : 125
Kelas : 2A
ipk : 3.1
-----------------------------
Nama : Sita
NIM : 126
Kelas : 2A
ipk : 3.9
-----------------------------
Nama : Miki
NIM : 127
Kelas : 2A
ipk : 3.7
-----------------------------
```

```
Data yang sudah terurut menggunakan INSERTION SORT (ASC)
Nama : Sofi
NIM : 125
Kelas : 2A
ipk : 3.1
-----------------------------
Nama : Zidan
NIM : 123
Kelas : 2A
ipk : 3.2
-----------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
ipk : 3.5
-----------------------------
Nama : Miki
NIM : 127
Kelas : 2A
ipk : 3.7
-----------------------------
Nama : Sita
NIM : 126
Kelas : 2A
ipk : 3.9
-----------------------------
```

Pertanyaan:
Ubahlah fungsi pada InsertionSort sehingga fungsi ini dapat melaksanakan proses sorting dengan cara descending

```
void insertionSort(){
    for (int i = 1; i < listMhs.length; i++){
        Mahasiswa04 temp = listMhs[i];
        int j = i;
        while (j > 0 && listMhs[j-1].ipk < temp.ipk){
            listMhs[j] = listMhs[j-1];
            j--;
        }
        listMhs[j] = temp;
    }
}
```

Outputnya:

```
xtension-pack-jdk\java\latest\bin\java.exe" --enable-preview -XX:+Show
paceStorage\d83555ec4f9c765b284f331a909f075b\redhat.java\jdt_ws\Prakti
Data Mahasiswa Sebelum Sorting:
Nama : Zidan
NIM : 123
Kelas : 2A
ipk : 3.2
------------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
ipk : 3.5
------------------------------
Nama : Sofi
NIM : 125
Kelas : 2A
ipk : 3.1
------------------------------
Nama : Sita
NIM : 126
Kelas : 2A
ipk : 3.9
------------------------------
Nama : Miki
NIM : 127
Kelas : 2A
ipk : 3.7
------------------------------
Data yang sudah terurut menggunakan INSERTION SORT (ASC)
Nama : Sita
NIM : 126
Kelas : 2A
ipk : 3.9
------------------------------
Nama : Miki
NIM : 127
Kelas : 2A
ipk : 3.7
------------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
ipk : 3.5
------------------------------
Nama : Zidan
NIM : 123
Kelas : 2A
ipk : 3.2
------------------------------
Nama : Sofi
NIM : 125
Kelas : 2A
ipk : 3.1
------------------------------
```

5.5 Latihan Praktikum
Waktu : 45 Menit
Perhatikan class diagram dibawah ini:

Dosen
kode: String nama: String jenisKelamin: Boolean usia: int
Dosen(kd: String, name: String, jk: Boolean, age: int) tampil(): void

DataDosen
dataDosen: Dosen[10] idx: int
tambah(dsn: Dosen): void tampil(): void SortingASC(): void sortingDSC():void insertionSort():void

Berdasarkan class diagram diatas buatlah menu dikelas main dengan pilihan menu:
1. Tambah data digunakan untuk menambahkan data dosen
2. Tampil data digunakan untuk menampilkan data seluruh dosen
3. Sorting ASC digunakan untuk mengurutkan data dosen berdasarkan usia dimulai dari dosen termuda ke dosen tertua menggunakan bublle Sort.
4. Sorting DSC digunakan untuk mengurutkan data dosen berdasarkan usia dimulai dari tertua ke dosen termuda dapat menggunakan algoritma selection sort atau insertion sort.

Jawab:

```
public class dosen04 {
    String kode;
    String nama;
    boolean jenisKelamin;
    int usia;

    dosen04(String kd, String name, boolean jk, int age){
        kode = kd;
        nama = name;
        jenisKelamin = jk;
        usia = age;
    }

    void tampil(){
        System.out.println("Kode : " + kode);
        System.out.println("Nama : " + nama);
        System.out.println("Jenis Kelamin : " + jenisKelamin);
        System.out.println("Usia : " + usia);
    }
}
```

```
public class dataDosen04 {
    dosen04[] dataDosen = new dosen04[10];
    int idx;

    void tambah(dosen04 dsn){
        if (idx < dataDosen.length){
            dataDosen[idx] = dsn;
            idx++;
        }else {
            System.out.println("Data Dosen Sudah Banyak!");
        }
    }
    // Untuk menampilkan Data dosen
    void tampil (){
        if (idx == 0){
            System.out.println("Tidak Ada Data Dosen.");
            return;
        }
        for (int i = 0; i < idx; i++){
            dataDosen[i].tampil();
            System.out.println();
        }
    }
    // Metode untuk mengurutkan data dosen secara ascending (asc)
    void sortingASC(){
        for (int i = 0; i < idx - 1; i++){
            for (int j = 0; j < idx - 1 -i; j++){
                if (dataDosen[j].usia > dataDosen[j+1].usia){
                    dosen04 temp = dataDosen[j];
                    dataDosen[j] = dataDosen[j + 1];
                    dataDosen[j + 1] = temp;
                }
            }
        }
    }
}
```

```
 // Metode untuk mengurutkan data dosen secara descending (dsc)
    void sortingDSC(){
        for (int i = 0; i < idx - 1; i++){
            for (int j = 0; j < idx - 1 - i; j++){
                if (dataDosen[j].usia < dataDosen[j + 1].usia){
                    dosen04 temp = dataDosen[j];
                    dataDosen[j] = dataDosen[j + 1];
                    dataDosen[j + 1] = temp;
                }
            }
        }
    }
}
```

```
import java.util.Scanner;

public class dosenDemo04 {
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        dataDosen04 datadsn = new dataDosen04();
        int pilihan;

        do {
            System.out.println("\nMenu");
            System.out.println("1. Tambah Data Dosen");
            System.out.println("2. Tampilkan Data Dosen");
            System.out.println("3. Sorting ASC (Usia Termuda Ke Tertua)");
            System.out.println("4. Sorting DSC (Usia Tertua ke Termuda)");
            System.out.println("5. Keluar");
            System.out.print("Pilih Menu: ");
            pilihan = sc.nextInt();
            sc.nextLine();

            switch (pilihan) {
                case 1:
                    for (int i = 0; i < 10; i++) {
                        System.out.println("Masukkan data dosen ke-" + (i + 1) + ":");
                        System.out.print("Masukkan kode dosen: ");
                        String kode = sc.nextLine();
                        System.out.print("Masukkan nama dosen: ");
                        String nama = sc.nextLine();
                        System.out.print("Masukkan jenis Kelamin (Perempuan / laki-laki): ");
                        char jk = sc.next().charAt(0);
                        Boolean jenisKelamin = (jk == 'l' || jk == 'p');
                        System.out.print("Masukkan usia dosen: ");
                        int usia = sc.nextInt();
                        sc.nextLine();
                        System.out.println("-------------------------------------------------
");

                        dosen04 dsn = new dosen04(kode, nama, jenisKelamin, usia);
                        datadsn.tambah(dsn);
                    }
                    break;
                case 2:
                    datadsn.tampil();
                    break;
                case 3:
                    datadsn.sortingASC();
                    System.out.println("Data Dosen Telah Diurutkan secara Ascending");
                    datadsn.tampil();
                    break;
                case 4:
```

```
                    datadsn.sortingDSC();
                    System.out.println("Data Dosen Telah Diurutkan secara Descending");
                    datadsn.tampil();
                    break;
                case 5:
                    System.out.println("keluar dari Program");
                    break;
                default:
                    System.out.println("Pilihan tidak Valid. Silahkan Coba Lagi");
            }
        }while (pilihan != 5);
    }
}
```

Outputnya:

```
C:\Praktikum ASD\Praktikum05>
C:\Praktikum ASD\Praktikum05> c: && cd "c:\Praktikum ASD\P
xtension-pack-jdk\java\latest\bin\java.exe" --enable-previ
paceStorage\d83555ec4f9c765b284f331a909f075b\redhat.java\j

Menu
1. Tambah Data Dosen
2. Tampilkan Data Dosen
3. Sorting ASC (Usia Termuda Ke Tertua)
4. Sorting DSC (Usia Tertua ke Termuda)
5. Keluar
Pilih Menu: 1
Masukkan data dosen ke-1:
Masukkan kode dosen: 120
Masukkan nama dosen: pak hendra
Masukkan jenis Kelamin (Perempuan / laki-laki): l
Masukkan usia dosen: 43
-----------------------------------------------
Masukkan data dosen ke-2:
Masukkan kode dosen: 121
Masukkan nama dosen: pak budi
Masukkan jenis Kelamin (Perempuan / laki-laki): l
Masukkan usia dosen: 45
-----------------------------------------------
Masukkan data dosen ke-3:
Masukkan kode dosen: 122
Masukkan nama dosen: pak hapis
Masukkan jenis Kelamin (Perempuan / laki-laki): l
Masukkan usia dosen: 46
-----------------------------------------------
```

```
Masukkan data dosen ke-4:
Masukkan kode dosen: 124
Masukkan nama dosen: pak fauzi
Masukkan jenis Kelamin (Perempuan / laki-laki): l
Masukkan usia dosen: 41
-----------------------------------------------
Masukkan data dosen ke-5:
Masukkan kode dosen: 125
Masukkan nama dosen: pak angga
Masukkan jenis Kelamin (Perempuan / laki-laki): l
Masukkan usia dosen: 47
-----------------------------------------------
Masukkan data dosen ke-6:
Masukkan kode dosen: 123
Masukkan nama dosen: bu kayla
Masukkan jenis Kelamin (Perempuan / laki-laki): p
Masukkan usia dosen: 34
-----------------------------------------------
Masukkan data dosen ke-7:
Masukkan kode dosen: 126
Masukkan nama dosen: bu angel
Masukkan jenis Kelamin (Perempuan / laki-laki): p
Masukkan usia dosen: 32
-----------------------------------------------
Masukkan data dosen ke-8:
Masukkan kode dosen: 127
Masukkan nama dosen: bu vit
Masukkan jenis Kelamin (Perempuan / laki-laki): p
Masukkan usia dosen: 35
-----------------------------------------------
```

```
Masukkan data dosen ke-9:
Masukkan kode dosen: 128
Masukkan nama dosen: bu vivi
Masukkan jenis Kelamin (Perempuan / laki-laki): p
Masukkan usia dosen: 31
-----------------------------------------------
Masukkan data dosen ke-10:
Masukkan kode dosen: 129
Masukkan nama dosen: bu rani
Masukkan jenis Kelamin (Perempuan / laki-laki): p
Masukkan usia dosen: 37
-----------------------------------------------

Menu
1. Tambah Data Dosen
2. Tampilkan Data Dosen
3. Sorting ASC (Usia Termuda Ke Tertua)
4. Sorting DSC (Usia Tertua ke Termuda)
5. Keluar
Pilih Menu: 2
Kode : 120
Nama : pak hendra
Jenis Kelamin : true
Usia : 43

Kode : 121
Nama : pak budi
Jenis Kelamin : true
Usia : 45
```

```
Kode : 122
Nama : pak hapis
Jenis Kelamin : true
Usia : 46

Kode : 124
Nama : pak fauzi
Jenis Kelamin : true
Usia : 41

Kode : 125
Nama : pak angga
Jenis Kelamin : true
Usia : 47

Kode : 123
Nama : bu kayla
Jenis Kelamin : true
Usia : 34

Kode : 126
Nama : bu angel
Jenis Kelamin : true
Usia : 32

Kode : 127
Nama : bu vit
Jenis Kelamin : true
Usia : 35
```

```
Kode : 128
Nama : bu vivi
Jenis Kelamin : true
Usia : 31

Kode : 129
Nama : bu rani
Jenis Kelamin : true
Usia : 37


Menu
1. Tambah Data Dosen
2. Tampilkan Data Dosen
3. Sorting ASC (Usia Termuda Ke Tertua)
4. Sorting DSC (Usia Tertua ke Termuda)
5. Keluar
Pilih Menu: 3
Data Dosen Telah Diurutkan secara Ascending
Kode : 128
Nama : bu vivi
Jenis Kelamin : true
Usia : 31

Kode : 126
Nama : bu angel
Jenis Kelamin : true
Usia : 32
```

```
Kode : 123
Nama : bu kayla
Jenis Kelamin : true
Usia : 34

Kode : 127
Nama : bu vit
Jenis Kelamin : true
Usia : 35

Kode : 129
Nama : bu rani
Jenis Kelamin : true
Usia : 37

Kode : 124
Nama : pak fauzi
Jenis Kelamin : true
Usia : 41

Kode : 120
Nama : pak hendra
Jenis Kelamin : true
Usia : 43

Kode : 121
Nama : pak budi
Jenis Kelamin : true
Usia : 45
```

```
Kode : 122
Nama : pak hapis
Jenis Kelamin : true
Usia : 46

Kode : 125
Nama : pak angga
Jenis Kelamin : true
Usia : 47


Menu
1. Tambah Data Dosen
2. Tampilkan Data Dosen
3. Sorting ASC (Usia Termuda Ke Tertua)
4. Sorting DSC (Usia Tertua ke Termuda)
5. Keluar
Pilih Menu: 4
Data Dosen Telah Diurutkan secara Descending
Kode : 125
Nama : pak angga
Jenis Kelamin : true
Usia : 47

Kode : 122
Nama : pak hapis
Jenis Kelamin : true
Usia : 46
```

```
Kode : 121
Nama : pak budi
Jenis Kelamin : true
Usia : 45

Kode : 120
Nama : pak hendra
Jenis Kelamin : true
Usia : 43

Kode : 124
Nama : pak fauzi
Jenis Kelamin : true
Usia : 41

Kode : 129
Nama : bu rani
Jenis Kelamin : true
Usia : 37

Kode : 127
Nama : bu vit
Jenis Kelamin : true
Usia : 35

Kode : 123
Nama : bu kayla
Jenis Kelamin : true
Usia : 34
```

```
Kode : 127
Nama : bu vit
Jenis Kelamin : true
Usia : 35

Kode : 123
Nama : bu kayla
Jenis Kelamin : true
Usia : 34

Kode : 126
Nama : bu angel
Jenis Kelamin : true
Usia : 32

Kode : 128
Nama : bu vivi
Jenis Kelamin : true
Usia : 31


Menu
1. Tambah Data Dosen
2. Tampilkan Data Dosen
3. Sorting ASC (Usia Termuda Ke Tertua)
4. Sorting DSC (Usia Tertua ke Termuda)
5. Keluar
Pilih Menu: 5
keluar dari Program
```