LAPORAN HASIL PRAKTIKUM
ALGORITMA STRUKTUR DATA
JOBSHEET 7

NAMA : JIRO AMMAR WAFI

NIM : 244107020190

KELAS : 1-E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

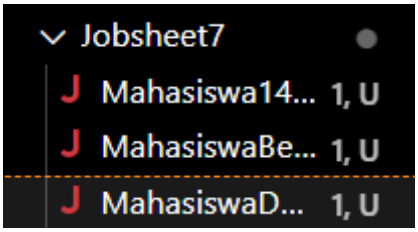
POLINEMA

2025

|| JOBSHEET VII - SEARCHING

A. Searching menggunakan Algoritma Sequential Search

1. Copy 3 class dari folder Jobsheet6 ke folder Jobsheet7:



2. Menambahkan method *sequentialSearching()* pada class MahasiswaBerprestasi, method ini berisi algoritma searching menggunakan teknik sequential:

```
int sequentialSearching(double cari) {
    int posisi = -1;
    for (int j=0; j<listMhs.length; j++){
        if (listMhs[j].ipk == cari){
            posisi = j;
            break;
        }
    }
    return posisi;
}
```

3. Menambahkan method tampilPosisi() pada class MahasiswaBerprestasi yang fungsinya untuk menampilkan pencocokan hasil searching:

```
void tampilPosisi(double x, int pos) {
    if (pos!=-1) {
        System.out.println("Data Mahasiswa dengan IPK: " + x + "
    } else {
        System.out.println("Data " + x + " tidak ditemukan");
    }
}
```

4. Menambahkan method tampilDataSearch() pada class MahasiswaBerprestasi, fungsinya ialah untuk menampilkan data yang sesuai dengan hasil searching:

```
void tampilDataSearch (double x, int pos) {
    if (pos != -1) {
        System.out.println("NIM\t : "+listMhs[pos].nim);
        System.out.println("Nama\t : "+listMhs[pos].nama);
        System.out.println("Kelas\t : "+listMhs[pos].kelas);
        System.out.println("IPK\t : "+ x);
    } else {
        System.out.println("Data Mahasiswa dengan IPK " + x + " tidak
    }
}
```

5. Menambahkan baris kode untuk:
    a. Variabel jumlah mahasiswa yang akan digunakan sebagai titik berhenti loop for:

```
int jumMhs = 5;
```

```
for (int i = 0; i < jumMhs; i++) {
```

    b. Memanggil method yang telah dibuat sebelumnya pada class MahasiswaBerprestasi:

```
// Searching data secara sequential
System.out.println(x:"--------------------------------------");
System.out.println(x:"Data Searching");
System.out.println(x:"--------------------------------------");
System.out.println(x:"Masukkan IPK mahasiswa yang ingin dicari: ")
System.out.print(s:"IPK: "); double cari = in.nextDouble();

System.out.println(x:"Searching secara Sequential");
double posisi = list.sequentialSearching(cari);
int pss = (int) posisi;
list.tampilPosisi(cari, pss);
list.tampilDataSearch(cari, pss);
in.close();
```

6. Verifikasi Kode: sukses

```
>>> Data mahasiswa yang telah di input:
Nama: A
NIM: 111
Kelas: 2
IPK: 3.3
----------------------
Nama: B
NIM: 222
Kelas: 2
IPK: 3.4
----------------------
Nama: C
NIM: 333
Kelas: 3
IPK: 3.5
----------------------
Nama: D
NIM: 444
Kelas: 4
IPK: 3.9
----------------------
Nama: E
NIM: 555
Kelas: 5
IPK: 3.3
----------------------
```

```
--------------------------------------
Data Searching
--------------------------------------
Masukkan IPK mahasiswa yang ingin dicari:
IPK: 3,3
Searching secara Sequential
Data Mahasiswa dengan IPK: 3.3 ditemukan pada indeks 0
NIM      : 111
Nama     : A
Kelas    : 2
IPK      : 3.3
```

|| Pertanyaan A - Sequential Searching

1. Perbedaan diantara kedua method dapat ditemukan pada fungsinya, fungsi method *tampilPosisi()* ialah untuk menampilkan posisi indeks yang sesuai dengan key yang dicari melalui searching. Sedangkan *tampilDataSearch()* akan menampilkan seluruh data yang disimpan yang sesuai dengan indeks tersebut, mengarah pada data yang disimpan melalui array of object.

2. Fungsi *break* di dalam pemilihan tersebut akan langsung memberhentikan looping *for* yang terjadi, apabila ditemukan key yang sesuai. Semisal terdapat 2 data yang memiliki key yang sesuai, key pertama yang ditemukan akan langsung dijadikan sebagai key yang akan ditampilkan, key kedua tidak akan dicari lagi.

B. Searching menggunakan Algoritma Binary Search

1. Method findBinarySearch() pada class MahasiswaBerprestasi, merupakan algoritma untuk menemukan data menggunakan teknik binary searching:

```
int findBinarySearch(double cari, int left, int right){
    int mid;
    if (right>=left){
        mid = (left+right)/2;
        if (cari == listMhs[mid].ipk){
            return (mid);
        } else if (listMhs[mid].ipk > cari){
            return findBinarySearch(cari, left, mid-1);
        } else {
            return findBinarySearch(cari, mid+1, right);
        }
    } return -1;
}
```

2. Memanggil method yang telah dibuat sebelumnya di class MahasiswaDemo, memisahkan antara searching secara sequential dan binary (preferensi saya);

```
// // Searching data secara Sequential
System.out.println(x:"Searching secara Sequential");
double posisi = list.sequentialSearching(cari);
int pss = (int) posisi;
list.tampilPosisi(cari, pss);
list.tampilDataSearch(cari, pss);
```

```
// Searching data secara Binary
System.out.println(x:"Searching secara Binary");
double posisi2 = list.findBinarySearch(cari, left:0, jumMhs-1);
int pss2 = (int) posisi2;
list.tampilPosisi(cari, pss2);
list.tampilDataSearch(cari, pss2);
```

3. Verifikasi Kode:

```
>>> Data mahasiswa yang telah di input:
Nama: Adi
NIM: 111
Kelas: 3
IPK: 3.1
----------------------
Nama: Ila
NIM: 222
Kelas: 3
IPK: 3.2
----------------------
Nama: Lia
NIM: 333
Kelas: 3
IPK: 3.3
----------------------
Nama: Sinta
NIM: 444
Kelas: 3
IPK: 3.5
----------------------
Nama: Anita
NIM: 555
Kelas: 3
IPK: 3.7
----------------------
```

```
--------------------------------------
Data Searching
--------------------------------------
Masukkan IPK mahasiswa yang ingin dicari:
IPK: 3,7
Searching secara Sequential
Data Mahasiswa dengan IPK: 3.7 ditemukan pada indeks 4
NIM      : 555
Nama     : Anita
Kelas    : 3
IPK      : 3.7
--------------------------------------
Searching secara Binary
Data Mahasiswa dengan IPK: 3.7 ditemukan pada indeks 4
NIM      : 555
Nama     : Anita
Kelas    : 3
IPK      : 3.7
```

|| Pertanyaan B - Binary Searching

1. Proses Divide: membagi array menjadi sub array

```
mid = (left+right)/2;
```

2. Proses Conquer: menyelesaikan sub array secara rekursif

```
} else if (listMhs[mid].ipk > cari){
    return findBinarySearch(cari, left, mid-1); // Conquer
} else {
    return findBinarySearch(cari, mid+1, right); // Conquer
}
```

3. Method *findBinarySearch()* tetap dapat berjalan, namun tak dapat menemukan key yang sesuai:

```
-------------------------------------
Data Searching
-------------------------------------
Masukkan IPK mahasiswa yang ingin dicari:
IPK: 2,9
Searching secara Sequential
Data Mahasiswa dengan IPK: 2.9 ditemukan pada indeks 4
NIM      : EEE
Nama     : EEE
Kelas    : EEE
IPK      : 2.9
-------------------------------------
Searching secara Binary
Data 2.9 tidak ditemukan
Data Mahasiswa dengan IPK 2.9 tidak ditemukan
```

Alasannya ialah, Binary Searching bergantung pada bagaimana data-data tersebut diurutkan baik ascending atau descending, karena konsep Divide n Conquer ialah membagi dua sisi (sub-array) yaitu kiri dan kanan, algoritma kesusahan untuk membuat keputusan akibat nilai tengah yang selalu berubah-ubah.

4. Hasilnya:

```
-------------------------------------
Searching secara Binary
Data 3.2 tidak ditemukan
Data Mahasiswa dengan IPK 3.2 tidak ditemukan
```

Yang perlu diubah (>) menjadi (<):

```
} else if (listMhs[mid].ipk < cari){
```

Pada awalnya kode tersebut dikhususkan untuk data yang berurutan secara ascending.

5. Modifikasi: Input dinamis dari keyboard

Class MahasiswaDemo14:

A. Baris input:

```
System.out.print(s:"Masukkan jumlah mahasiswa: ");
int jumMhs = in.nextInt(); in.nextLine();
MahasiswaBerprestasi14 list = new MahasiswaBerprestasi14(jumMhs);
```

B. Perubahan titik berhenti looping for:

```
for (int i = 0; i < jumMhs; i++) {
```

Class MahasiswaBerprestasi14:

C. Menginisialisasi panjang array menggunakan konstruktor berparameter:

```
Mahasiswa14 [] listMhs;
int idx;

MahasiswaBerprestasi14 (int index){
    listMhs = new Mahasiswa14[index];
}
```

|| Latihan Praktikum

Demonstrasi Praktikum:

A. Input statis untuk panjang array:

```
Masukkan banyak data: 3
```

B. Data array of object:

```
>>> Tampilkan Data
Nama Dosen: A
Kode: A
Jenis Kelamin: Laki-laki
Usia: 21
----------------------
Nama Dosen: A
Kode: A
Jenis Kelamin: Perempuan
Usia: 23
----------------------
Nama Dosen: B
Kode: B
Jenis Kelamin: Laki-laki
Usia: 23
----------------------
```

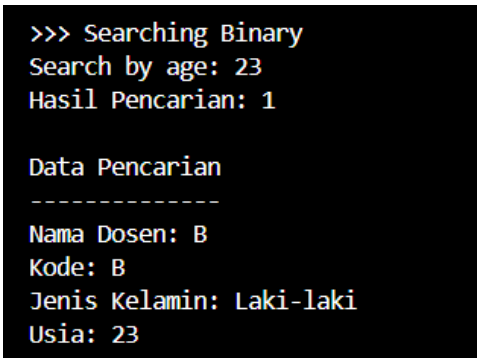
C. Searching Sequential - Hasil searching lebih dari 1:

```
>>> Searching Sequential
Search by name: A
Hasil Pencarian: 2
Terdapat lebih dari 1 hasil yang ditemukan !

Data Pencarian
--------------
Nama Dosen: A
Kode: A
Jenis Kelamin: Laki-laki
Usia: 21
Nama Dosen: A
Kode: A
Jenis Kelamin: Perempuan
Usia: 23
```

D. Searching Sequential - Hasil searching hanya ada 1:

```
>>> Searching Sequential
Search by name: B
Hasil Pencarian: 1

Data Pencarian
--------------
Nama Dosen: B
Kode: B
Jenis Kelamin: Laki-laki
Usia: 23
```

E. Searching Binary - Error
Hasil searching lebih dari 1,namun tak dapat menampilkan semua data searching, kesalahan algoritma yang tak saya temukan.



F. Searching Binary - Hasil searching tepat, namun output yang ditampilkan bercampur dengan output yang tidak sesuai.